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

OASIS Standard, 15 March 2005

Document identifier:
saml-profiles-2.0-os

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

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

SAML V2.0 Contributors:
Conor P. Cahill, AOL
John Hughes, Atos Origin
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rebekah Metz, Booz Allen Hamilton
Rick Randall, Booz Allen Hamilton
Thomas Wisniewski, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Nick Ragouzis, Individual
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Peter C Davis, Neustar
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Paul Madsen, NTT
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems

Table of Contents

Very high level

Compare to sequence diagram

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.

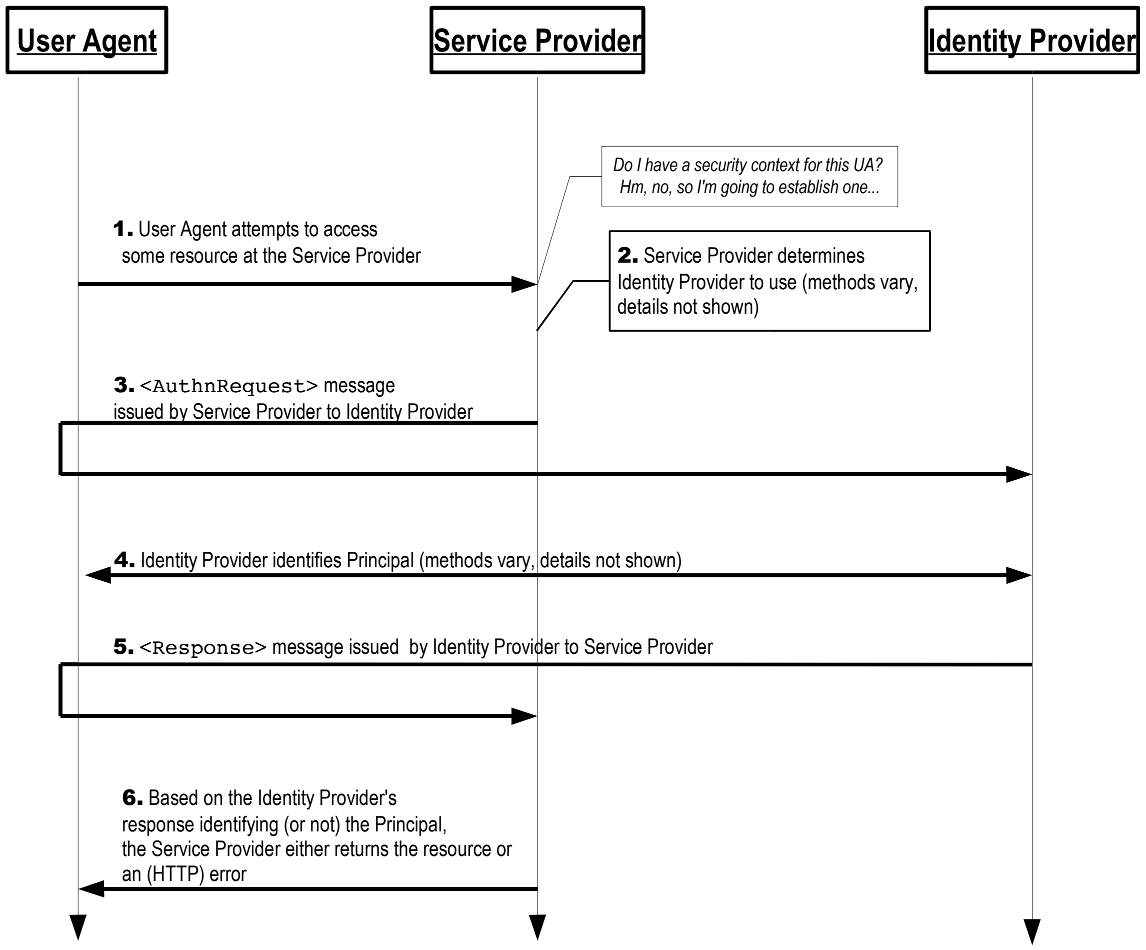

User Agent
Service Provider
Identity Provider
Do I have a security context for this UA? Hm, no, so I'm going to establish one...
1. User Agent attempts to access some resource at the Service Provider
2. Service Provider determines Identity Provider to use (methods vary, details not shown)
3. <AuthnRequest> message issued by Service Provider to Identity Provider
4. Identity Provider identifies Principal (methods vary, details not shown)
5. <Response> message issued by Identity Provider to Service Provider
6. Based on the Identity Provider's response identifying (or not) the Principal, the Service Provider either returns the resource or an (HTTP) error

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

6. Service Provider grants or denies access to Principal

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

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

4.1.3 Profile Description

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

Single Sign-On Service

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

Assertion Consumer Service

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

4.1.3.1 HTTP Request to Service Provider

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

4.1.3.2 Service Provider Determines Identity Provider

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

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

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

The SAML2Int Profile (following)specifies using the redirect binding in[SDP-SP02] & [SDP-IDP02]. Documentation of a well formed AuthnRequest requires multiple references.

The SAML2Int Profile (following) specifies using the redirect binding in[SDP-SP02] & [SDP-IDP02]. The OASIS specification saml-bindings-2.0-os's Redirect binding section also follows.

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

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

The identity provider MUST process the `<AuthnRequest>` message as described in [SAMLCore]. This may constrain the subsequent interactions with the user agent, for example if the `IsPassive` attribute is included. **More critical can be requirements for ForceAuthn and MFA.**

4.1.3.4 Identity Provider Identifies Principal

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

4.1.3.5 Identity Provider Issues <Response> to Service Provider

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

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

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

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

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

4.1.3.6 Service Provider Grants or Denies Access to User Agent

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

4.1.4 Use of Authentication Request Protocol

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

4.1.4.1 <AuthnRequest> Usage

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

SP identifies self to IdP with Issuer

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.

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

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

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

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

4.1.4.2 <Response> Usage

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

- 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.
- The bearer `<SubjectConfirmation>` element described above MUST contain a `<SubjectConfirmationData>` element that contains a `Recipient` attribute containing the service provider's assertion consumer service URL and a `NotOnOrAfter` attribute that limits the window during which the assertion can be delivered. It MAY contain an `Address` attribute limiting the client address from which the assertion can be delivered. It MUST NOT contain a `NotBefore` attribute. If the containing message is in response to an `<AuthnRequest>`, then the `InResponseTo` attribute MUST match the request's `ID`.
- 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>`.
- Other conditions (and other `<Audience>` elements) MAY be included as requested by the service provider or at the discretion of the identity provider. (Of course, all such conditions MUST be understood by and accepted by the service provider in order for the assertion to be considered valid.) The identity provider is NOT obligated to honor the requested set of `<Conditions>` in the `<AuthnRequest>`, if any.

4.1.4.3 <Response> Message Processing Rules

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

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

4.1.4.4 Artifact-Specific <Response> Message Processing Rules

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

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

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

4.1.4.5 POST-Specific Processing Rules

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

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

4.1.5 Unsolicited Responses

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

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

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

4.1.6 Use of Metadata

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

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

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

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

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

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

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

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

4.2 Enhanced Client or Proxy (ECP) Profile

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

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

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

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

4.2.1 Required Information

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

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

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

Description: Given below.

Updates: None.

4.2.2 Profile Overview

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

9 References

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

[Anders] A suggestion on how to implement SAML browser bindings without using "Artifacts". See 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.educause.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*. IETF Internet-Draft, December 2004. See http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-05.txt.

[MSURL] Microsoft technical support article. See 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.

[PAOS] R. Aarts. *Liberty Reverse HTTP Binding for SOAP Specification* Version 1.0. Liberty Alliance Project, 2003. See https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf.

[Rescorla-Sec] E. Rescorla et al. *Guidelines for Writing RFC Text on Security Considerations*. IETF RFC 3552, July 2003. See http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt.

[RFC1738] T. Berners-Lee et al. *Uniform Resource Locators (URL)*. IETF RFC 1738, December 1994. See http://www.ietf.org/rfc/rfc1738.txt.

[RFC1750] D. Eastlake et al. *Randomness Recommendations for Security*. IETF RFC 1750, December 1994. See http://www.ietf.org/rfc/rfc1750.txt.

[RFC1945] T. Berners-Lee et al. *Hypertext Transfer Protocol – HTTP/1.0*. IETF RFC 1945, May 1996. See http://www.ietf.org/rfc/rfc1945.txt.

[RFC2045] N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. IETF RFC 2045, November 1996. See 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. See http://www.ietf.org/rfc/rfc2119.txt.

[RFC2246] T. Dierks. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999. See http://www.ietf.org/rfc/rfc2246.txt.

[RFC2256] M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*. IETF RFC 2256, December 1997. See http://www.ietf.org/rfc/rfc2256.txt.

[RFC2279] F. Yergeau. *UTF-8, a transformation format of ISO 10646*. IETF RFC 2279, January 1998. See http://www.ietf.org/rfc/rfc2279.txt.

[RFC2616] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999. See http://www.ietf.org/rfc/rfc2616.txt.

[RFC2617] J. Franks et al. *HTTP Authentication: Basic and Digest Access Authentication*. IETF RFC 2617, Jujne 1999. See http://www.ietf.org/rfc/rfc2617.txt.

[RFC2798] M. Smith. *Definition of the inetOrgPerson LDAP Object Class*. IETF RFC 2798, April 2000. 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, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLConform] P. Mishra et al. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0.* OASIS SSTC, March 2005. Document ID saml-conformance-2.0-os. 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, March 2005. Document ID saml-core-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLDCE-xsd] S. Cantor et al. SAML DCE PAC attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-dce-2.0. See http://www.oasis-open.org/committees/security/.

[SAMLECP-xsd] S. Cantor et al. SAML ECP profile schema. OASIS SSTC, March 2005. Document ID saml-schema-ecp-2.0. See http://www.oasis-open.org/committees/security/.

[SAMLGloss] J. Hodges et al. *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLX500-xsd] S. Cantor et al. SAML X.500/LDAP attribute profile schema. OASIS SSTC, March 2005. Document ID saml-schema-x500-2.0. See http://www.oasis-open.org/committees/security/.

[SAMLMeta] S. Cantor et al. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os. See http://www.oasis-open.org/committees/security/.

[SAMLReqs] Darren Platt et al. *OASIS Security Services Use Cases and Requirements*. OASIS SSTC, May 2001. Document ID draft-sstc-saml-reqs-01. See 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, March 2005. Document ID saml-sec-consider-2.0-os. 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, March 2005. Document ID 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*. World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-2/.

[SESSION] RL 'Bob' Morgan. *Support of target web server sessions in Shibboleth*. Shibboleth, May 2001. See http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt.

[ShibMarlena] Marlena Erdos et al. *Shibboleth Architecture DRAFT v05*. Shibboleth, May 2002. See 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. See 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*. Shibboleth, April 2001. See 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*. World Wide Web Consortium Recommendation, December 2002. See http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

[**XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web Consortium Recommendation, February 2002. See 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 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

SAML V2.0 Deployment Profile for Federation Interoperability

Version	2.00
Date	2019-12-09
Location	https://kantarainitiative.github.io/SAMLprofiles/saml2int.html
Status	This Group Recommendation Specification was developed and approved by the Federation Interoperability Working Group (FIWG) and later approved by All Member Ballot on 2020-02-26. See the Kantara Initiative Operating Procedures for more information https://kantarainitiative.org/confluence/display/GI/All+Policies?preview=/37750179/104600198/KI%20Operating%20Procedures%20Version%203.0.pdf

Contributors - V2.0

- Keith Wessel (University of Illinois at Urbana-Champaign) - *Editor*
- Scott Cantor (Ohio State University)
- Eric Goodman (University of California, Office of the President)
- Andrew Morgan (Oregon State University)
- Judith Bush (OCLC)
- Thomas Lenggenhager (SWITCH)
- Alex Stuart (JISC)
- Keith Hazelton (Internet2)
- Chris Phillips (CANARIE)
- Nicholas Roy (Internet2)

- Alan Buxey (MyUniDays, LTD.)

Contributors - previous versions

- Andreas Åkre Solberg (UNINETT)
- Eve Maler (ForgeRock)
- Leif Johansson (SUNET)
- Jeff Hodges (Kings Mountain Systems)
- Ian Young (Independent)
- Nate Klingenstein (Signet, Inc.)
- Bob Morgan (University of Washington)

Abstract

This profile specifies behavior and options that deployments of the SAML V2.0 Web Browser SSO profile [SAML2Prof], and related profiles, are required or permitted to rely on. This deployment profile should not be confused with a SAML implementation profile, such as [SAML2Impl]. A deployment profile specifies how software should be configured given a specific set of deployment goals. An implementation profile tells software developers how they must implement their software in order to meet basic requirements.

Copyright Notice

License

Table of Contents

1. Introduction

This revision, the first major rewrite of this material, reflects the input of many experienced implementers and deployers of this technology and best practice developed over 15 years of experience with varied approaches. While it has an emphasis on highly-scaled multi-lateral federation deployments involving thousands of Identity Providers (IdPs) and Service Providers (SPs), most of these requirements are applicable to virtually any significant deployment of SAML SSO.

The requirements specified are in addition to all normative requirements of the underlying Web Browser SSO and Single Logout profiles [SAML2Prof], as modified by the Approved Errata [SAML2Err], and readers are assumed to be familiar with all relevant reference documents. Any such requirements are not repeated here except where deemed necessary to highlight a point of discussion or draw attention to an issue addressed in errata, but remain implied.

Nothing in this profile should be taken to imply that disclosing personally identifiable information, or indeed any information, is required from an IdP with respect to any particular SP. That remains at the discretion of applicable settings, user consent, or other appropriate means in accordance with regulations and policies. However, this profile does obligate IdPs to honor certain key signals from an SP in the area of subject identification and requires successful responses to include specific SAML Attributes [X500SAMLattr] under certain conditions. Failure is always an option.

Note that SAML features that are optional, or lack mandatory processing rules, are assumed to be optional and out of scope of this profile if not otherwise precluded or given specific processing rules.

This profile addresses only behavior specific to the direct participants in the covered profiles. It does not include additional processing requirements that may be important when proxying.

1.1. Notation and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification uses the following typographical conventions in text: `<ns:Element>`, `Attribute`, **Datatype**, `OtherCode`. The normative requirements of this specification are individually labeled with a unique identifier in the following form: **[SDP-EXAMPLE01]**. All information within these requirements should be considered normative unless it is set in *italic* type. Italicized text is non-normative and is intended to provide additional information that may be helpful in implementing the normative requirements.

The abbreviations IdP and SP are used below to refer to Identity Providers and Service Providers in the sense of their usage within the SAML Browser SSO and Single Logout profiles.

Whether explicit or implicit, all the requirements in this document are meant to apply to deployments of SAML profiles and may involve explicit support for requirements by SAML-implementing software and/or supplemental support via application code. Deployments of a Service Provider may refer to both stand-alone implementations of SAML, libraries integrated with an application, or any combination of the two. It is difficult to define a clear boundary between a Service Provider and the application/service it represents, and unnecessary to do so for the purposes of this document.

1.1.1. References to SAML 2.0 specification

When referring to elements from the SAML 2.0 core specification [SAML2Core], the following syntax is used:

- `<samlp:ProtocolElement>` - for elements from the SAML 2.0 Protocol namespace.
- `<saml:AssertionElement>` - for elements from the SAML 2.0 Assertion namespace.

When referring to elements from the SAML 2.0 metadata specification [SAML2Meta], the following syntax is used:

- `<md:MetadataElement>`

... Skipping some of the profile ...

3. Service Provider Requirements

This section provides requirements specific to SPs, in addition to the Common Requirements above.

3.1. Web Browser SSO

[SDP-SP01]

SPs MUST support the Web Browser SSO profile [SAML2Prof], as updated by the Approved Errata [SAML2Err], with behavior, capabilities, and options consistent with the additional constraints specified in this section.

3.1.1. Requests

Binding

[SDP-SP02] *Potentially replace with a FedCM binding?*

The HTTP-Redirect binding [SAML2Bind] MUST be used for the transmission of `<samlp:AuthnRequest>` messages.

[SDP-SP03]

Requests MUST NOT be issued inside an HTML frame or via any mechanism that would require the use of third-party cookies by the IdP to establish or recover a session with the User Agent. This will typically imply that requests will involve a full-frame redirect, in order that the top level window origin be associated with the IdP.

Request Content

[SDP-SP04]

The `<samlp:AuthnRequest>` message MUST either omit the `<samlp:NameIDPolicy>` element (RECOMMENDED), or the element MUST contain an `AllowCreate` attribute of "true" and MUST NOT contain a `Format` attribute.

[SDP-SP05]

The message SHOULD contain an `AssertionConsumerServiceURL` attribute and MUST NOT contain an `AssertionConsumerServiceIndex` attribute (i.e., the desired endpoint MUST be the default, or identified via the `AssertionConsumerServiceURL` attribute).

[SDP-SP06]

The `AssertionConsumerServiceURL` value, if present, MUST match an endpoint location expressed in the SP's metadata exactly, without requiring URL canonicalization/normalization.

As an example, the SP cannot specify URLs that include a port number (e.g., https://sp.example.com:443/acs) in its requests unless it also includes that port number in the URLs specified in its metadata, and vice versa.

Authentication Contexts

[SDP-SP07] **See https://wiki.refeds.org/display/ASS/How+to+request+authentication+contexts for details on requesting multifactor authentication.**

An SP that does not require a specific `<saml:AuthnContextClassRef>` value MUST NOT include a `<samlp:RequestedAuthnContext>` element in its requests.

An SP that requires specific `<saml:AuthnContextClassRef>` values MUST specify the allowable values in a `<samlp:RequestedAuthnContext>` element in its requests, with the `Comparison` attribute set to `exact`.

An SP should not request a `<saml:AuthnContextClassRef>` *value in the absence of a shared understanding between itself and the IdP regarding its definition.*

3.1.2. Responses

Binding

[SDP-SP08]

SPs MUST support the HTTP-POST binding for the receipt of `<samlp:Response>` messages. Support for other bindings is OPTIONAL.

[SDP-SP09]

The endpoint(s) at which an SP supports receipt of `<samlp:Response>` messages MUST be protected by TLS/SSL.

XML Encryption

[SDP-SP10]

SPs MUST support decryption of `<saml:EncryptedAssertion>` elements. Support for other encrypted constructs is OPTIONAL.

Error Handling

[SDP-SP11]

SPs MUST gracefully handle error responses containing `<samlp:StatusCode>` other than `urn:oasis:names:tc:SAML:2.0:status:Success`.

[SDP-SP12]

If a successful authentication response lacks sufficient or appropriate SAML Attributes (including subject identifiers) for successful SP operation, the SP MUST display a meaningful status message to the user. This message MUST direct the user to appropriate support resources offered by the SP or, alternatively, to the `errorURL` attribute in an IdP's metadata.

There are many reasons an SP may be unable or choose not to provide service to a user based on an given authentication response. IdPs failing to release the necessary SAML Attributes is the most prevalent interoperability issue encountered in larger, general purpose federations, which is why this scenario is singled out here.

3.1.3. Subject Identification

NameID Formats

[SDP-SP13]

SPs MUST NOT require the presence of a `<saml:NameID>` element.

Use of `<saml:NameID>` *elements in this profile is restricted to their role in the Single Logout profile, and not for long term identification of subjects. Standardized SAML Attributes are used instead, as described below.*

Subject Identifiers

[SDP-SP14]

If an SP requires persistent tracking/identification of its users (as most do), then it MUST support one or both of the SAML Attributes defined by [SAML2SubjId] for this purpose.

SPs MAY support legacy or historical `<saml:NameID>` and `<saml:Attribute>` identifier content for compatibility reasons but MUST NOT require their use.

If an SP requires coordination and/or correlation of user activity between itself and other SPs, then the SAML Attribute named `urn:oasis:names:tc:SAML:attribute:subject-id` *is appropriate. Otherwise the SAML Attribute named* `urn:oasis:names:tc:SAML:attribute:pairwise-id` *can be used.*

Subject Identifier Requirements Signaling

[SDP-SP15]

An SP MUST represent its identifier requirements in its SAML metadata, consistent with the Requirements Signaling mechanism defined in [SAML2SubjId].

Identifier Scoping

[SDP-SP16]

SPs MUST prevent unintended identifier collisions in the values asserted by different IdPs, and the required identifier types, per [SAML2SubjId], are "scoped" via a DNS-like syntax to help fulfill this requirement.

[SDP-SP17]

SPs MUST associate identifier scopes with IdPs such that only authorized IdPs may assert identifiers with particular scopes for particular purposes.

It is RECOMMENDED that the `<shibmd:Scope>` metadata extension defined in [SAML2SubjId] be supported for this purpose. SPs MAY ignore any such extension elements whose `regexp` attribute is `true` or `1`. SPs MUST NOT rely on this extension unless the metadata is verifiably obtained from a third party that is trusted to supply it.

In the event that this extension cannot be used, then SPs MUST apply policy established in some other manner.

Note that scopes and IdPs do not necessarily have a 1:1 relationship; it may well be legitimate for multiple IdPs to assert a given scope, or for an IdP to assert identifiers in multiple scopes, but the rules for this should be explicit and enforced.

Displayable Identifiers

The required identifier types above are opaque, unknown to users in most cases, and unsuitable for display.

[SDP-SP18]

SPs requiring the display of identifiers to users, the identification of other users via searching, selection, etc., and similar use cases SHOULD rely on additional suitable SAML Attributes such as:

- `urn:oid:0.9.2342.19200300.100.1.3` (mail)
- `urn:oid:2.16.840.1.113730.3.1.241` (displayName)
- `urn:oid:2.5.4.42` (givenName)
- `urn:oid:2.5.4.4` (sn)

Note that most standardized SAML Attributes of this sort tend to be defined as multi-valued.

3.1.4. Attribute Value Constraints

[SDP-SP19]

When consuming SAML Attributes with standardized definitions in external specifications, SPs MUST NOT impose constraints beyond the definitions of those attributes.

For example, the definition of the `mail` *attribute (in SAML,* `urn:oid:0.9.2342.19200300.100.1.3` *) explicitly allows for multiple values, so an SP that consumes it for some purpose must necessarily allow for that possibility.*

3.1.5. Usability

Silo-oriented, multi-tenant approaches to federated application deployment create an inherent friction with the intended design of the web, user behavior and experience, and the needs of collaboration inherent in many applications. SSO, when integrated poorly, can negatively impact usability, and the following sections, while not strictly matters of SAML interoperability, have a significant effect on the perception of the system as a whole and on the successful adoption of SSO, regardless of the protocol.

The web inherently operates on the basis of *addressability* of resources; that is, users expect to be able to access a piece of information or an application function directly, without regard for their identity, current level of access, or what is convenient for an application developer to support. This leads naturally to the ability to create bookmarks to what matters to them, and users will consistently route around attempts to force them through proxies, portals, and other artificial access paths.

At a high level, these issues fall under the term `deep linking`.

For a wide range of applications in the collaborative space, this notion is not merely convenient, but utterly essential, because such applications presume the sharing of resources with peers between organizations.

For the purposes of the following requirements, we will refer to applications that rely on the exposure of resource URLs that may be shared between users from multiple organizations as "collaborative" applications, even if their purpose may not specifically align with that term.

Support for Multiple IdPs

[SDP-SP20] *@ SAML Profile §4.1.3.2*

SPs MUST allow for the possibility that any given request requiring authentication may be potentially satisfied by more than one IdP. That is, any scenario in which a piece of content, policy, configuration, or decision on the part of an application is bound to an IdP MUST be constructed in a fashion such that more than one IdP may be so bound.

This requirement flows from both the inherent requirements of collaborative applications described above, and from the simple reality that enterprises vary in their structure. Some organizations rely on more than one IdP due to administrative boundaries, but frequently contract for or access services as a single body. Thus, any presumed mapping between a contract or set of access policies and a single SAML IdP is too constraining. This constraint imposes a need for complex proxying of SSO by many organizations and SPs are cautioned to avoid it.

Deep Linking

[SDP-SP21]

Applications SHOULD, and collaborative applications MUST, support deep linking. Deep linking implies maintaining support for such links across the boundary of a Web Browser SSO profile interaction involving any IdP necessary to complete the login process.

It should be possible to request a resource and (authorization permitting) have it supplied as the result of a successful Web Browser SSO profile exchange.

Deep linking implies support for SP-initiated SSO, i.e., the direct generation of authentication request messages in response to unauthenticated or insufficiently-authenticated access attempts to an application as a whole, or to specific protected content. Deep linking may co-exist with support for unsolicited responses (so-called IdP-initiated SSO), but precludes its requirement.

[SDP-SP22]

It is RECOMMENDED that SPs support the preservation of POST bodies across a successful Web Browser SSO profile exchange, subject to size limitations dictated by policy or implementation constraints.

Discovery

Deep linking also implies support for some form of IdP "discovery", the process by which an SP establishes which IdP to use on behalf of a subject. Use of IdP-initiated SSO is a common workaround for supporting discovery, but cannot be required when deep linking is supported, in addition to having other drawbacks.

A common means of discovery is the mapping of resource/application URL (typically virtual host, sometimes path) to a specific IdP. This is strongly discouraged, and is disallowed for collaborative applications, since it makes the sharing of URLs between users from multiple organizations at best inconvenient, and in some cases, impossible.

[SDP-SP23] *@ SAML Profile §4.1.3.2*

SPs that support deep linking MUST support some form of Identity Provider discovery that accomodates all, or at least the vast majority, of their user base. Support for caching mechanisms such as cookies or other persistence solutions is encouraged.

3.2. Single Logout

[SDP-SP24]

SPs MAY support the Single Logout profile [SAML2Prof], as updated by the Approved Errata [SAML2Err]. The following requirements apply in the case of such support.

3.2.1. Requests

Binding

[SDP-SP25]

The HTTP-Redirect binding [SAML2Bind] MUST be used for the transmission of `<samlp:LogoutRequest>` messages.

[SDP-SP26]

SPs MUST support the HTTP-Redirect [SAML2Bind] binding for the receipt of `<samlp:LogoutRequest>` messages, in the event that inbound `<samlp:LogoutRequest>` messages are supported.

[SDP-SP27]

Requests MUST NOT be issued inside an HTML frame or via any mechanism that would require the use of third-party cookies by the IdP to establish or recover a session with the User Agent. This will typically imply that requests must involve a full-frame redirect, in order that the top level window origin be associated with the IdP.

The full-frame requirement is also necessary to ensure that full control of the user interface is released to the IdP.

Request Content

[SDP-SP28]

Requests MUST be signed (via a signature created in accordance with the HTTP-Redirect binding [SAML2Bind]).

[SDP-SP29]

The `<saml:NameID>` element included in `<samlp:LogoutRequest>` messages MUST exactly match the corresponding element received from the IdP, including its element content and all XML attributes included therein.

[SDP-SP30]

The `<saml:NameID>` element in `<samlp:LogoutRequest>` messages MUST NOT be encrypted.

The normative requirement for the use of transient identifiers is intended to obviate the need for XML Encryption.

3.2.2. Responses

Binding

[SDP-SP31]

The HTTP-Redirect binding [SAML2Bind] MUST be used for the transmission of `<samlp:LogoutResponse>` messages.

[SDP-SP32]

SPs MUST support the HTTP-Redirect [SAML2Bind] binding for the receipt of `<samlp:LogoutResponse>` messages, in the event that they do not include the `<aslo:Asynchronous>` extension [SAML2ASLO] in all of their requests.

Response Content

[SDP-SP33]

Responses MUST be signed (via a signature created in accordance with the HTTP-Redirect binding [SAML2Bind]).

3.2.3. Behavioral Requirements

[SDP-SP34]

SPs MUST terminate a subject’s local session before issuing a `<samlp:LogoutRequest>` message to the IdP.

This ensures the safest possible result for subjects in the event that logout fails for some reason, as it often will.

[SDP-SP35]

SPs MUST NOT issue a `<samlp:LogoutRequest>` message as the result of an idle activity timeout.

Timeout of a single application/service must not trigger logout of an SSO session because this imposes a single service's requirements on an entire IdP deployment. Applications with sensitive requirements should consider other mechanisms, such as the `ForceAuthn` *attribute, to achieve their goals.*

3.2.4. Logout and Virtual Hosting

[SDP-SP36]

An SP that maintains distinct sessions across multiple virtual hosts SHOULD identify itself by means of a distinct entityID (with associated metadata) for each virtual host.

A single entity can have only one well-defined `<SingleLogoutService>` *endpoint per binding. Cookies are typically host-based and logout cannot typically be implemented easily across virtual hosts. Unlike during SSO, a* `<samlp:LogoutRequest>` *message cannot specify a particular response endpoint, so this scenario is generally not viable.*

3.3. Metadata and Trust Management

3.3.1. Support for Multiple Keys

The ability to perform seamless key migration depends upon proper support for consuming and/or leveraging multiple keys at the same time.

[SDP-SP37]

SP deployments MUST support multiple signing certificates in IdP metadata and MUST support validation of XML signatures using a key from any of them.

[SDP-SP38]

SP deployments MUST be able to support multiple decryption keys and MUST be able to decrypt `<saml:EncryptedAssertion>` elements encrypted with any configured key.

3.3.2. Metadata Content

[SDP-SP39]

By virtue of this profile's requirements, an SP's metadata MUST contain:

- an `<md:SPSSODescriptor>` role element containing
  - at least one `<md:AssertionConsumerService>` endpoint element

  - at least one `<md:KeyDescriptor>` element whose `use` attribute is omitted or set to `encryption`
  - an `<md:Extensions>` element at the role level containing
    - an `<mdui:UIInfo>` extension element containing the child elements `<mdui:DisplayName>`, `<mdui:Logo>`, and `<mdui:PrivacyStatementURL>`
    - an `<mdattr:EntityAttributes>` extension element for signaling Subject Identifier requirements with previously prescribed content
- an `<md:ContactPerson>` element with a `contactType` of `technical` and an `<md:EmailAddress>` element

If the SP supports the Single Logout profile, then its metadata MUST contain (within its `<md:SPSSODescriptor>` role element):

- at least one `<md:KeyDescriptor>` element whose `use` attribute is omitted or set to `signing`
- at least one `<md:SingleLogoutService>` endpoint element (this MAY be omitted if the SP solely issues `<samlp:LogoutRequest>` messages containing the `<aslo:Asynchronous>` extension [SAML2ASLO])

4. Identity Provider Requirements

This section provides requirements specific to IdPs, in addition to the Common Requirements above.

4.1. Web Browser SSO

[SDP-IDP01]

IdPs MUST support the Web Browser SSO profile [SAML2Prof], as updated by the Approved Errata [SAML2Err], with behavior, capabilities, and options consistent with the additional constraints specified in this section.

4.1.1. Requests

Binding

[SDP-IDP02]

IdPs MUST support the HTTP-Redirect binding [SAML2Bind] for the receipt of `<samlp:AuthnRequest>` messages.

[SDP-IDP03]

The endpoint(s) at which an IdP supports receipt of `<samlp:AuthnRequest>` messages MUST be protected by TLS/SSL.

Signing

[SDP-IDP04]

IdPs MUST support unsigned requests generally but MUST reject unsigned requests in the event that an SP's metadata includes an `AuthnRequestsSigned` attribute set to `true` or `1`.

[SDP-IDP05]

If a request is signed, IdPs MUST successfully verify the signature or fail the request. An IdP MAY handle a signature verification failure locally rather than via an error response to the SP.

Endpoint Selection/Verification

[SDP-IDP06]

IdPs MUST verify the `AssertionConsumerServiceURL` supplied in an SP's `<samlp:AuthnRequest>` (if any) against the `<md:AssertionConsumerService>` elements in the SP's metadata. In the absence of such a value, the default endpoint from the SP's metadata MUST be used for the response.

When verifying the `AssertionConsumerServiceURL`, it is RECOMMENDED that the IdP perform a case-sensitive string comparison between the requested value and the values found in the SP's metadata. It is OPTIONAL to apply any form of URL canonicalization.

The Web Browser SSO profile [SAML2Prof] notes that validation of the response endpoint is required but does not mandate a specific approach, primarily due to metadata being an optional portion of the original standard. The above is the most common and interoperable approach to meeting this requirement.

Forced Re-Authentication

[SDP-IDP07]

IdPs MUST ensure that any response to a `<samlp:AuthnRequest>` that contains the attribute `ForceAuthn` set to `true` or `1` results in an authentication challenge that requires proof that the subject is present. If this condition is met, the IdP MUST also reflect this by setting the value of the `AuthnInstant` value in the assertion it returns to a fresh value.

If an IdP cannot prove subject presence, then it MUST fail the request and SHOULD respond to the SP with a SAML error status.

Due to the potential for confusion over more frequent authentication challenges, the IdP may wish to indicate when this feature is being used on the login user interface it presents to the user.

4.1.2. Responses

Binding

[SDP-IDP08]

IdPs MUST support the HTTP-POST binding [SAML2Bind] for the transmission of `<samlp:Response>` messages.

Response Content

[SDP-IDP09]

Successful responses MUST be directly signed using a `<ds:Signature>` element within the `<samlp:Response>` element. Error responses MAY be unsigned.

[SDP-IDP10]

Successful responses MUST contain exactly one SAML assertion. The assertion MUST contain exactly one `<saml:AuthnStatement>` element and MUST contain zero or one `<saml:AttributeStatement>` elements. The assertion within the response MAY be directly signed.

[SDP-IDP11]

In the event the HTTP-POST binding [SAML2Bind] is used, assertions MUST be encrypted and transmitted via a `<saml:EncryptedAssertion>` element. Information intended for the consumption of the SP MUST NOT be further encrypted via `<saml:EncryptedID>` or `<saml:EncryptedAttribute>` constructs.

While encryption is viewed in some quarters as onerous or unnecessary, interoperability is enhanced by uniformity. Moreover, a spate of recent vulnerabilities across the industry would have been almost entirely mitigated by its use, demonstrating that it is no longer acceptable to view it as an optional part of front-channel delivery of assertions, if it ever was.

4.1.3. Subject Identifiers

[SDP-IDP12]

Assertions MUST contain a `<saml:NameID>` element with the `urn:oasis:names:tc:SAML:2.0:nameid-format:transient` Format, as defined in [SAML2Core], for the purposes of logout.

[SDP-IDP13]

IdPs MUST support one or both of the SAML Attributes defined by [SAML2SubjId] for non-transient identification of subjects. Support for both is RECOMMENDED.

[SDP-IDP14]

IdPs MUST enumerate the scope(s) of the subject identifiers they support in their metadata by means of the `<shibmd:Scope>` extension element, as defined in [SAML2SubjId]. They MUST NOT contain a regular expression (i.e., each element's `regexp` attribute MUST be set to `false` or `0`).

The element(s) may be positioned as an extension of either the `<md:EntityDescriptor>` or `<md:IDPSSODescriptor>` as deemed appropriate.

Note that while common, it is not a requirement for the scope(s) to be contained within the IdP's entityID, nor for it to bear any relationship to other data asserted by the IdP, such as email addresses.

Subject Identifier Requirements Signaling

[SDP-IDP15]

IdPs MUST support the metadata-based identifier requirement signaling mechanism defined in [SAML2SubjId].

The purpose of this requirement is to provide a level of confidence to a signaling SP that a compliant IdP which fails to do as instructed is unwilling or unable to fulfill the requirements rather than merely oblivious to them.

[SDP-IDP16]

If an IdP cannot or will not satisfy the requirements of an SP in this respect, then it MAY return an assertion without the data it is unable to provide or return an error as it sees fit.

[SDP-IDP17]

In the absence of any signaling by an SP, an IdP MAY supply either, both, or neither of the SAML Attributes defined in [SAML2SubjId], or return an error as it sees fit.

4.1.4. Attributes

[SDP-IDP18]

`<saml:Attribute>` elements MUST contain a NameFormat of `urn:oasis:names:tc:SAML:2.0:attrname-format:uri` .

This requirement ensures unique, non-conflicting naming of SAML Attributes even in cases involving custom requirements for which no standard SAML Attributes may exist.

[SDP-IDP19]

It is RECOMMENDED that the content of each `<saml:AttributeValue>` element be limited to a single child text node (i.e., a simple string value).

Note that this refers to `<saml:AttributeValue>` *elements, not* `<saml:Attribute>` *elements, and refers to the form of each individual value. It discourages the use of complex XML content models within the value of a SAML Attribute.*

[SDP-IDP20]

Multiple values of a `<saml:Attribute>` MUST be expressed as individual `<saml:AttributeValue>` elements rather than embedded in a delimited form within a single `<saml:AttributeValue>` element.

4.2. Single Logout

[SDP-IDP21]

IdPs MUST support the Single Logout profile [SAML2Prof], as updated by the Approved Errata [SAML2Err], with behavior, capabilities, and options consistent with the additional constraints specified in this section.

The term "IdP session" is used to refer to the ongoing state between the IdP and its clients allowing for SSO. Support for logout implies supporting termination of a subject's IdP session in response to receiving a `<samlp:LogoutRequest>` *or upon some administrative signal.*

[SDP-IDP22]

IdPs MAY allow a subject the option to maintain their IdP session rather than unilaterally terminating it.

[SDP-IDP23]

IdPs MAY support the propagation of logout signaling to SPs.

4.2.1. Requests

Binding

[SDP-IDP24]

The HTTP-Redirect binding [SAML2Bind] MUST be used for the transmission of

`<samlp:LogoutRequest>` messages, in the event that propagation is supported.

[SDP-IDP25]

IdPs MUST support the HTTP-Redirect [SAML2Bind] binding for the receipt of `<samlp:LogoutRequest>` messages.

4.2.2. Request Content

[SDP-IDP26]

Requests MUST be signed (via a signature created in accordance with the HTTP-Redirect binding [SAML2Bind]).

[SDP-IDP27]

The `<saml:NameID>` element in `<samlp:LogoutRequest>` messages MUST NOT be encrypted.

The normative requirement for the use of transient identifiers is intended to obviate the need for XML Encryption.

4.2.3. Responses

Binding

[SDP-IDP28]

The HTTP-Redirect binding [SAML2Bind] MUST be used for the transmission of `<samlp:LogoutResponse>` messages.

[SDP-IDP29]

IdPs MUST support the HTTP-Redirect [SAML2Bind] binding for the receipt of `<samlp:LogoutResponse>` messages, in the event that `<samlp:LogoutRequest>` propagation is supported.

Response Content

[SDP-IDP30]

Responses MUST be signed (via a signature created in accordance with the HTTP-Redirect binding [SAML2Bind]).

[SDP-IDP31]

The `<samlp:StatusCode>` in the response issued by the IdP MUST reflect whether the IdP session was successfully terminated.

4.3. Metadata and Trust Management

4.3.1. Support for Multiple Keys

The ability to perform seamless key migration depends upon proper support for consuming and/or leveraging multiple keys at the same time.

[SDP-IDP32]

IdP deployments MUST support multiple signing certificates in SP metadata and MUST support validation of signatures using a key from any of them.

4.3.2. Metadata Content

[SDP-IDP33]

By virtue of this profile's requirements, an IdP's metadata MUST contain:

- an `<md:IDPSSODescriptor>` role element containing
  - at least one `<md:SingleSignOnService>` endpoint element
  - at least one `<md:SingleLogoutService>` endpoint element
  - at least one `<md:KeyDescriptor>` element whose `use` attribute is omitted or set to `signing`
  - an `errorURL` attribute
  - an `<md:Extensions>` element at the role level containing
    - an `<mdui:UIInfo>` extension element containing the child elements `<mdui:DisplayName>` and `<mdui:Logo>`
    - at least one `<shibmd:Scope>` element
      - alternately, the `<shibmd:Scope>` element(s) MAY instead reside in an `<md:Extensions>` element at the root (`<md:EntityDescriptor>`) level
- an `<md:ContactPerson>` element with a `contactType` of `technical` and an `<md:EmailAddress>` element

5. References

5.1. Normative

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

- [RFC8174] IETF RFC 8174, Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words, May 2017. http://www.ietf.org/rfc/rfc8174.txt
- [RFC4051] IETF RFC 4051, Additional XML Security Uniform Resource Identifiers, April 2005. https://www.ietf.org/rfc/rfc4051.txt
- [SAML2Core] OASIS Standard, Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0 /saml-core-2.0-os.pdf
- [SAML2Bind] OASIS Standard, Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf
- [SAML2Prof] OASIS Standard, Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
- [SAML2Meta] OASIS Standard, Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf
- [X500SAMLattr] OASIS Committee Specification, SAML V2.0 X.500/LDAP Attribute Profile, March 2008. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-attribute-x500-cs-01.pdf
- [SAML2MDIOP] OASIS Standard, SAML V2.0 Metadata Interoperability Profile Version 1.0, October 2019. http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-iop-os.pdf
- [SAML2Err] OASIS Approved Errata, SAML Version 2.0 Errata 05, May 2012. http://docs.oasis-open.org/security/saml/v2.0/sstc-saml-approved-errata-2.0.pdf
- [XMLEnc] D. Eastlake et al. XML Encryption Syntax and Processing. W3C Recommendation, April 2013. https://www.w3.org/TR/xmlenc-core1/
- [XMLSig] D. Eastlake et al. XML-Signature Syntax and Processing, Version 1.1. W3C Recommendation, April 2013. https://www.w3.org/TR/xmldsig-core1/
- [SAML2SubjId] OASIS Committee Specification, SAMLV2.0 Subject Identifier Attributes Profile Version 1.0, January 2019. https://docs.oasis-open.org/security/saml-subject-id-attr/v1.0 /cs01/saml-subject-id-attr-v1.0-cs01.pdf
- [SAML2ASLO] OASIS Committee Specification, SAML V2.0 Asynchronous Single Logout Profile Extension Version 1.0, November 2012. http://docs.oasis-open.org/security/saml/Post2.0/saml-async-slo/v1.0/cs01/saml-async-slo-v1.0-cs01.pdf
- [MetaUI] OASIS Standard, SAML V2.0 Metadata Extensions for Login and Discovery User

Interface Version 1.0, October 2019. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-metadata-ui/v1.0/os/sstc-saml-metadata-ui-v1.0-os.pdf

- [MetaAttr] OASIS Committee Specification, SAML V2.0 Metadata Extension for Entity Attributes Version 1.0, August 2009. http://docs.oasis-open.org/security/saml/Post2.0/sstc-metadata-attr-cs-01.pdf
- [SAML2Impl] Kantara Initiative Profile, SAML V2.0 Implementation Profile for Federation Interoperability Version 1.0, April 2018. https://kantarainitiative.github.io/SAMLprofiles/fedinterop.html

5.2. Non-Normative

- [XMLEncBreak] Jager and Somorovsky, How to Break XML Encryption, October 2011. http://www.nds.rub.de/media/nds/veroeffentlichungen/2011/10/22/HowToBreakXMLenc.pdf

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

OASIS Standard, 15 March 2005

Document identifier:
saml-bindings-2.0-os

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

Editors:
Scott Cantor, Internet2
Frederick Hirsch, Nokia
John Kemp, Nokia
Rob Philpott, RSA Security
Eve Maler, Sun Microsystems

SAML V2.0 Contributors:
Conor P. Cahill, AOL
John Hughes, Atos Origin
Hal Lockhart, BEA Systems
Michael Beach, Boeing
Rebekah Metz, Booz Allen Hamilton
Rick Randall, Booz Allen Hamilton
Thomas Wisniewski, Entrust
Irving Reid, Hewlett-Packard
Paula Austel, IBM
Maryann Hondo, IBM
Michael McIntosh, IBM
Tony Nadalin, IBM
Nick Ragouzis, Individual
Scott Cantor, Internet2
RL 'Bob' Morgan, Internet2
Peter C Davis, Neustar
Jeff Hodges, Neustar
Frederick Hirsch, Nokia
John Kemp, Nokia
Paul Madsen, NTT
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems
Eve Maler, Sun Microsystems
Ron Monzillo, Sun Microsystems

Table of Contents

recipient MAY be another system entity, with the HTTP requester acting as an intermediary, as defined by specific profiles.

3.3.3.2 SAML Response in SOAP Request, HTTP Response

When the HTTP requester delivers a SAML response message to the intended recipient using the PAOS binding, it places it as the only element in the SOAP body in a SOAP envelope in an HTTP request. The HTTP requester may or may not be the originator of the SAML response. The SOAP envelope MAY contain arbitrary SOAP headers defined by PAOS, SAML profiles, or additional specifications. The SAML exchange is considered complete and the HTTP response is unspecified by this binding.

Profiles MAY define additional constraints on the HTTP content of non-SOAP responses during the exchanges covered by this binding.

3.3.4 Caching

HTTP proxies should not cache SAML protocol messages. To ensure this, the following rules SHOULD be followed.

When using HTTP 1.1, requesters sending SAML protocol messages SHOULD:

- Include a `Cache-Control` header field set to "`no-cache, no-store`".
- Include a `Pragma` header field set to "`no-cache`".

When using HTTP 1.1, responders returning SAML protocol messages SHOULD:

- Include a `Cache-Control` header field set to "`no-cache, no-store, must-revalidate, private`".
- Include a `Pragma` header field set to "`no-cache`".
- NOT include a Validator, such as a `Last-Modified` or ETag header.

3.3.5 Security Considerations

The HTTP requester in the PAOS binding may act as a SOAP intermediary and when it does, transport layer security for origin authentication, integrity and confidentiality may not meet end-end security requirements. In this case security at the SOAP message layer is recommended.

3.3.5.1 Error Reporting

Standard HTTP and SOAP error conventions MUST be observed. Errors that occur during SAML processing MUST NOT be signaled at the HTTP or SOAP layer and MUST be handled using SAML response messages with an error `<samlp:Status>` element.

3.3.5.2 Metadata Considerations

Support for the PAOS binding SHOULD be reflected by indicating a URL endpoint at which HTTP requests and/or SAML protocol messages contained in SOAP envelopes for a particular protocol or profile are to be sent. Either a single endpoint or distinct request and response endpoints MAY be supplied.

3.4 HTTP Redirect Binding

The HTTP Redirect binding defines a mechanism by which SAML protocol messages can be transmitted within URL parameters. Permissible URL length is theoretically infinite, but unpredictably limited in practice. Therefore, specialized encodings are needed to carry XML messages on a URL, and larger or more complex message content can be sent using the HTTP POST or Artifact bindings.

This binding MAY be composed with the HTTP POST binding (see Section 3.5) and the HTTP Artifact binding (see Section 3.6) to transmit request and response messages in a single protocol exchange using two different bindings.

This binding involves the use of a message encoding. While the definition of this binding includes the definition of one particular message encoding, others MAY be defined and used.

3.4.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect

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

Description: Given below.

Updates: None.

3.4.2 Overview

The HTTP Redirect binding is intended for cases in which the SAML requester and responder need to communicate using an HTTP user agent (as defined in HTTP 1.1 [RFC2616]) as an intermediary. This may be necessary, for example, if the communicating parties do not share a direct path of communication. It may also be needed if the responder requires an interaction with the user agent in order to fulfill the request, such as when the user agent must authenticate to it.

Note that some HTTP user agents may have the capacity to play a more active role in the protocol exchange and may support other bindings that use HTTP, such as the SOAP and Reverse SOAP bindings. This binding assumes nothing apart from the capabilities of a common web browser.

3.4.3 RelayState

RelayState data MAY be included with a SAML protocol message transmitted with this binding. The value MUST NOT exceed 80 bytes in length and SHOULD be integrity protected by the entity creating the message independent of any other protections that may or may not exist during message transmission. Signing is not realistic given the space limitation, but because the value is exposed to third-party tampering, the entity SHOULD ensure that the value has not been tampered with by using a checksum, a pseudo-random value, or similar means.

If a SAML request message is accompanied by RelayState data, then the SAML responder MUST return its SAML protocol response using a binding that also supports a RelayState mechanism, and it MUST place the exact data it received with the request into the corresponding RelayState parameter in the response.

If no such value is included with a SAML request message, or if the SAML response message is being generated without a corresponding request, then the SAML responder MAY include RelayState data to be interpreted by the recipient based on the use of a profile or prior agreement between the parties.

3.4.4 Message Encoding

Messages are encoded for use with this binding using a URL encoding technique, and transmitted using the HTTP GET method. There are many possible ways to encode XML into a URL, depending on the constraints in effect. This specification defines one such method without precluding others. Binding endpoints SHOULD indicate which encodings they support using metadata, when appropriate. Particular encodings MUST be uniquely identified with a URI when defined. It is not a requirement that all possible SAML messages be encodable with a particular set of rules, but the rules MUST clearly indicate which messages or content can or cannot be so encoded.

A URL encoding MUST place the message entirely within the URL query string, and MUST reserve the rest of the URL for the endpoint of the message recipient.

A query string parameter named `SAMLEncoding` is reserved to identify the encoding mechanism used. If this parameter is omitted, then the value is assumed to be `urn:oasis:names:tc:SAML:2.0:bindings:URL-Encoding:DEFLATE`.

All endpoints that support this binding MUST support the DEFLATE encoding described in the following sub-section.

3.4.4.1 DEFLATE Encoding

Identification: urn:oasis:names:tc:SAML:2.0:bindings:URL-Encoding:DEFLATE

SAML protocol messages can be encoded into a URL via the DEFLATE compression method (see [RFC1951]). In such an encoding, the following procedure should be applied to the original SAML protocol message's XML serialization:

1. Any signature on the SAML protocol message, including the `<ds:Signature>` XML element itself, MUST be removed. Note that if the content of the message includes another signature, such as a signed SAML assertion, this embedded signature is not removed. However, the length of such a message after encoding essentially precludes using this mechanism. Thus SAML protocol messages that contain signed content SHOULD NOT be encoded using this mechanism.
2. The DEFLATE compression mechanism, as specified in [RFC1951] is then applied to the entire remaining XML content of the original SAML protocol message.
3. The compressed data is subsequently base64-encoded according to the rules specified in IETF RFC 2045 [RFC2045]. Linefeeds or other whitespace MUST be removed from the result.
4. The base-64 encoded data is then URL-encoded, and added to the URL as a query string parameter which MUST be named `SAMLRequest` (if the message is a SAML request) or `SAMLResponse` (if the message is a SAML response).
5. If RelayState data is to accompany the SAML protocol message, it MUST be URL-encoded and placed in an additional query string parameter named `RelayState`.
6. If the original SAML protocol message was signed using an XML digital signature, a new signature covering the encoded data as specified above MUST be attached using the rules stated below.

XML digital signatures are not directly URL-encoded according to the above rules, due to space concerns. If the underlying SAML protocol message is signed with an XML signature [XMLSig], the URL-encoded form of the message MUST be signed as follows:

1. The signature algorithm identifier MUST be included as an additional query string parameter, named `SigAlg`. The value of this parameter MUST be a URI that identifies the algorithm used to sign the URL-encoded SAML protocol message, specified according to [XMLSig] or whatever specification governs the algorithm.
2. To construct the signature, a string consisting of the concatenation of the `RelayState` (if present), `SigAlg`, and `SAMLRequest` (or `SAMLResponse`) query string parameters (each one URL-encoded) is constructed in one of the following ways (ordered as below):

```
SAMLRequest=value&RelayState=value&SigAlg=value
SAMLResponse=value&RelayState=value&SigAlg=value
```

3. The resulting string of bytes is the octet string to be fed into the signature algorithm. Any other content in the original query string is not included and not signed.
4. The signature value MUST be encoded using the base64 encoding (see RFC 2045 [RFC2045]) with any whitespace removed, and included as a query string parameter named `Signature`. Note that some characters in the base64-encoded signature value may themselves require URL-encoding before being added.

5. The following signature algorithms (see [XMLSig]) and their URI representations MUST be supported with this encoding mechanism:
    - DSAwithSHA1 – http://www.w3.org/2000/09/xmldsig#dsa-sha1
    - RSAwithSHA1 – http://www.w3.org/2000/09/xmldsig#rsa-sha1

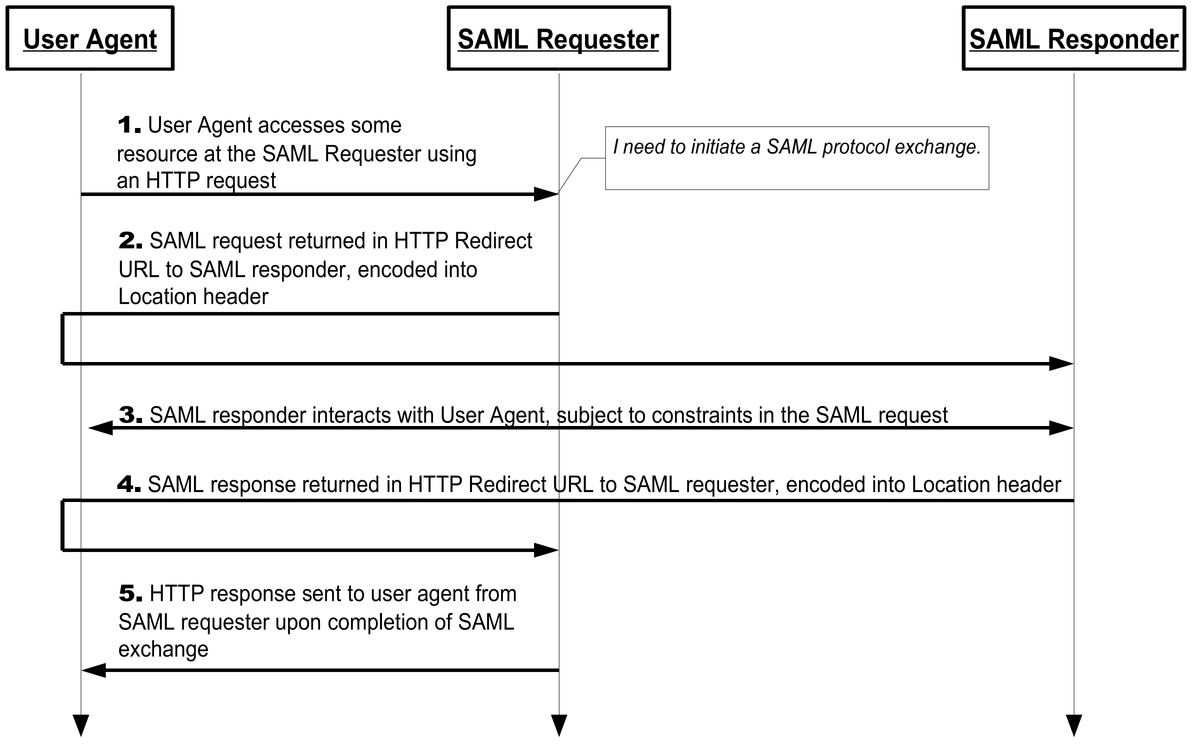
Note that when verifying signatures, the order of the query string parameters on the resulting URL to be verified is not prescribed by this binding. The parameters may appear in any order. Before verifying a signature, if any, the relying party MUST ensure that the parameter values to be verified are ordered as required by the signing rules above.

Further, note that URL-encoding is not canonical; that is, there are multiple legal encodings for a given value. The relying party MUST therefore perform the verification step using the original URL-encoded values it received on the query string. It is not sufficient to re-encode the parameters after they have been processed by software because the resulting encoding may not match the signer's encoding.

Finally, note that if there is no `RelayState` value, the entire parameter should be omitted from the signature computation (and not included as an empty parameter name).

3.4.5 Message Exchange

The system model used for SAML conversations via this binding is a request-response model, but these messages are sent to the user agent in an HTTP response and delivered to the message recipient in an HTTP request. The HTTP interactions before, between, and after these exchanges take place is unspecified. Both the SAML requester and the SAML responder are assumed to be HTTP responders. See the following sequence diagram illustrating the messages exchanged.


User Agent
SAML Requester
SAML Responder
1. User Agent accesses some resource at the SAML Requester using an HTTP request
I need to initiate a SAML protocol exchange.
2. SAML request returned in HTTP Redirect URL to SAML responder, encoded into Location header
3. SAML responder interacts with User Agent, subject to constraints in the SAML request
4. SAML response returned in HTTP Redirect URL to SAML requester, encoded into Location header
5. HTTP response sent to user agent from SAML requester upon completion of SAML exchange

1. Initially, the user agent makes an arbitrary HTTP request to a system entity. In the course of processing the request, the system entity decides to initiate a SAML protocol exchange.
2. The system entity acting as a SAML requester responds to the HTTP request from the user agent in step 1 by returning a SAML request. The SAML request is returned encoded into the HTTP

response's Location header, and the HTTP status MUST be either 303 or 302. The SAML requester MAY include additional presentation and content in the HTTP response to facilitate the user agent's transmission of the message, as defined in HTTP 1.1 [RFC2616]. The user agent delivers the SAML request by issuing an HTTP GET request to the SAML responder.

3. In general, the SAML responder MAY respond to the SAML request by immediately returning a SAML response or MAY return arbitrary content to facilitate subsequent interaction with the user agent necessary to fulfill the request. Specific protocols and profiles may include mechanisms to indicate the requester's level of willingness to permit this kind of interaction (for example, the `IsPassive` attribute in `<samlp:AuthnRequest>`) .
4. Eventually the responder SHOULD return a SAML response to the user agent to be returned to the SAML requester. The SAML response is returned in the same fashion as described for the SAML request in step 2.
5. Upon receiving the SAML response, the SAML requester returns an arbitrary HTTP response to the user agent.

3.4.5.1 HTTP and Caching Considerations

HTTP proxies and the user agent intermediary should not cache SAML protocol messages. To ensure this, the following rules SHOULD be followed.

When returning SAML protocol messages using HTTP 1.1, HTTP responders SHOULD:

- Include a `Cache-Control` header field set to "`no-cache, no-store`".
- Include a `Pragma` header field set to "`no-cache`".

There are no other restrictions on the use of HTTP headers.

3.4.5.2 Security Considerations

The presence of the user agent intermediary means that the requester and responder cannot rely on the transport layer for end-end authentication, integrity and confidentiality. URL-encoded messages MAY be signed to provide origin authentication and integrity if the encoding method specifies a means for signing.

If the message is signed, the `Destination` XML attribute in the root SAML element of the protocol message MUST contain the URL to which the sender has instructed the user agent to deliver the message. The recipient MUST then verify that the value matches the location at which the message has been received.

This binding SHOULD NOT be used if the content of the request or response should not be exposed to the user agent intermediary. Otherwise, confidentiality of both SAML requests and SAML responses is OPTIONAL and depends on the environment of use. If confidentiality is necessary, SSL 3.0 [SSL3] or TLS 1.0 [RFC2246] SHOULD be used to protect the message in transit between the user agent and the SAML requester and responder.

Note also that URL-encoded messages may be exposed in a variety of HTTP logs as well as the HTTP "Referer" header.

Before deployment, each combination of authentication, message integrity, and confidentiality mechanisms SHOULD be analyzed for vulnerability in the context of the specific protocol exchange, and the deployment environment. See specific protocol processing rules in [SAMLCore], and the SAML security considerations document [SAMLSecure] for a detailed discussion.

In general, this binding relies on message-level authentication and integrity protection via signing and does not support confidentiality of messages from the user agent intermediary.

3.4.6 Error Reporting

A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD return a SAML response message with a second-level `<samlp:StatusCode>` value of `urn:oasis:names:tc:SAML:2.0:status:RequestDenied`.

HTTP interactions during the message exchange MUST NOT use HTTP error status codes to indicate failures in SAML processing, since the user agent is not a full party to the SAML protocol exchange.

For more information about SAML status codes, see the SAML assertions and protocols specification [SAMLCore].

3.4.7 Metadata Considerations

Support for the HTTP Redirect binding SHOULD be reflected by indicating URL endpoints at which requests and responses for a particular protocol or profile should be sent. Either a single endpoint or distinct request and response endpoints MAY be supplied.

3.4.8 Example SAML Message Exchange Using HTTP Redirect

In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair is exchanged using the HTTP Redirect binding.

First, here are the actual SAML protocol messages being exchanged:

```
<samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="d2b7c388cec36fa7c39c28fd298644a8" IssueInstant="2004-01-
21T19:00:49Z" Version="2.0">
    <Issuer>https://IdentityProvider.com/SAML</Issuer>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">005a06e0-ad82-110d-a556-004005b13a2b</NameID>
    <samlp:SessionIndex>1</samlp:SessionIndex>
</samlp:LogoutRequest>
```

```
<samlp:LogoutResponse xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="b0730d21b628110d8b7e004005b13a2b"
InResponseTo="d2b7c388cec36fa7c39c28fd298644a8"
    IssueInstant="2004-01-21T19:00:49Z" Version="2.0">
    <Issuer>https://ServiceProvider.com/SAML</Issuer>
    <samlp:Status>
        <samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
</samlp:LogoutResponse>
```

The initial HTTP request from the user agent in step 1 is not defined by this binding. To initiate the logout protocol exchange, the SAML requester returns the following HTTP response, containing a signed SAML request message. The `SAMLRequest` parameter value is actually derived from the request message above. The signature portion is only illustrative and not the result of an actual computation. Note that the line feeds in the HTTP `Location` header below are an artifact of the document, and there are no line feeds in the actual header value.

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:00:49 GMT
```

```
Location:
https://ServiceProvider.com/SAML/SLO/Browser?SAMLRequest=fVFdS8MwFH0f7D%
2BUvGdNsq62oSsIQyhMESc%2B%2BJYlmRbWpObeyvz3puv2IMjyFM7HPedyK1DdsZdb%2F%
2BEHfLFfgwVMTt3RgTwzazIEJ72CFqRTnQWJWu7uH7dSLJjsg0ev%2FZFMlttiBWADtt6R%
2BSyJr9msiRH7O70sCm31Mj%2Bo%2BC%
2B1KA5GlEWeZaogSQMw2MYBKodrIhjLKONU8FdeSsZkVr6T5M0GiHMjvWCknqZXZ2OoPxF7kG
naGOuwxZ%2Fn4L9bY8NC%
2By4du1XpRXnxPcXizSZ58KFTeHujEWkNPZylsh9bAMYYUjO2Uiy3jCpTCMo5M1StVjmN9SO1
50sl9lU6RV2Dp0vsLIy7NM7YU82r9B90PrvCf85W%2FwL8zSVQzAEAAA%3D%
3D&RelayState=0043bfc1bc45110dae17004005b13a2b&SigAlg=http%3A%2F%
2Fwww.w3.org%2F200%2F09%2Fxmldsig%23rsa-
sha1&Signature=NOTAREALSIGNATUREBUTTHEREALONEWOULDGOHERE
Content-Type: text/html; charset=iso-8859-1
```

After any unspecified interactions may have taken place, the SAML responder returns the HTTP response below containing the signed SAML response message. Again, the `SAMLResponse` parameter value is actually derived from the response message above. The signature portion is only illustrative and not the result of an actual computation.

```
HTTP/1.1 302 Object Moved
Date: 21 Jan 2004 07:00:49 GMT
Location:
https://IdentityProvider.com/SAML/SLO/Response?SAMLResponse=fVFNa4QwEL0X%
2Bh8k912TaDUGFUp7EbZQ6rKH3mKcbQVNJBOX%2FvxaXQ9tYec0vHlv3nzkqIZ%2BlAf7YSf%
2FBjhagxB8Db1BuZQKMjkjrcIOpVEDoPRa1o8vB8n3VI7OeqttT1bJbbJCBOc7a8j9XTBH9Vy
QhqYRbTlrEi4Yo61oUqA0pvShYZHiDQkqs411tAVpeZPqSAgNOkrOas4zzcW55ZlI4liJrTXi
BJVBr4wvCJ877ijbcXZkmaRUxtk7CU7gcB5mLu8pKVddvghd%
2Ben9iDIMa3CXTsOrs5euBbfXdgh%2F9snDK%2FEqW69Ye%2BUnvGL%2F8CfbQnBS%
2FQS3z4QLW9aT1oBIws0j%2FGOyAb9%2FV34Dw5k779IBAAA%
3D&RelayState=0043bfc1bc45110dae17004005b13a2b&SigAlg=http%3A%2F%
2Fwww.w3.org%2F200%2F09%2Fxmldsig%23rsa-
sha1&Signature=NOTAREALSIGNATUREBUTTHEREALONEWOULDGOHERE
Content-Type: text/html; charset=iso-8859-1
```

3.5 HTTP POST Binding

The HTTP POST binding defines a mechanism by which SAML protocol messages may be transmitted within the base64-encoded content of an HTML form control.

This binding MAY be composed with the HTTP Redirect binding (see Section 3.4) and the HTTP Artifact binding (see Section 3.6) to transmit request and response messages in a single protocol exchange using two different bindings.

3.5.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST

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

Description: Given below.

Updates: Effectively replaces the binding aspects of the Browser/POST profile in SAML V1.1 [SAML11Bind].

3.5.2 Overview

The HTTP POST binding is intended for cases in which the SAML requester and responder need to communicate using an HTTP user agent (as defined in HTTP 1.1 [RFC2616]) as an intermediary. This may be necessary, for example, if the communicating parties do not share a direct path of communication. It may also be needed if the responder requires an interaction with the user agent in order to fulfill the request, such as when the user agent must authenticate to it.