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

Committee Draft 04, 15 January 2005

Document identifier: sstc-saml-bindings-2.0-cd-04


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
Tim Alsop, CyberSafe Limited
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
Charles Knouse, Oblix
Steve Anderson, OpenNetwork
Prateek Mishra, Principal Identity
John Linn, RSA Security
Rob Philpott, RSA Security
Jahan Moreh, Sigaba
Anne Anderson, Sun Microsystems
Abstract:
This specification defines protocol bindings for the use of SAML assertions and request-response
messages in communications protocols and frameworks.

Status:
This is a Committee Draft approved by the Security Services Technical Committee on 15
January 2005.

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

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

1 Introduction........................................................................................................................................5
  1.1 Protocol Binding Concepts...........................................................................................................5
  1.2 Notation........................................................................................................................................5
2 Guidelines for Specifying Additional Protocol Bindings..................................................................7
3 Protocol Bindings................................................................................................................................8
  3.1 General Considerations................................................................................................................8
    3.1.1 Use of RelayState.......................................................................................................................8
    3.1.2 Security....................................................................................................................................8
      3.1.2.1 Use of SSL 3.0 or TLS 1.0..................................................................................................8
      3.1.2.2 Data Origin Authentication...............................................................................................8
      3.1.2.3 Message Integrity...............................................................................................................8
      3.1.2.4 Message Confidentiality....................................................................................................9
      3.1.2.5 Security Considerations....................................................................................................9
  3.2 SAML SOAP Binding....................................................................................................................9
    3.2.1 Required Information................................................................................................................9
    3.2.2 Protocol-Independent Aspects of the SAML SOAP Binding......................................................10
      3.2.2.1 Basic Operation................................................................................................................10
      3.2.2.2 SOAP Headers................................................................................................................10
    3.2.3 Use of SOAP over HTTP..........................................................................................................11
      3.2.3.1 HTTP Headers................................................................................................................11
      3.2.3.2 Caching............................................................................................................................11
      3.2.3.3 Error Reporting................................................................................................................11
      3.2.3.4 Metadata Considerations..................................................................................................12
      3.2.3.5 Example SAML Message Exchange Using SOAP over HTTP.................................12
  3.3 Reverse SOAP (PAOS) Binding......................................................................................................13
    3.3.1 Required Information...............................................................................................................13
    3.3.2 Overview................................................................................................................................13
    3.3.3 Message Exchange..................................................................................................................13
      3.3.3.1 HTTP Request; SAML Request in SOAP Response.........................................................14
      3.3.3.2 SAML Response in SOAP Request, HTTP Response.....................................................15
    3.3.4 Caching....................................................................................................................................15
    3.3.5 Security Considerations..........................................................................................................15
      3.3.5.1 Error Reporting................................................................................................................15
      3.3.5.2 Metadata Considerations..................................................................................................15
  3.4 HTTP Redirect Binding................................................................................................................15
    3.4.1 Required Information...............................................................................................................16
    3.4.2 Overview................................................................................................................................16
    3.4.3 RelayState................................................................................................................................16
    3.4.4 Message Encoding..................................................................................................................16
      3.4.4.1 DEFLATE Encoding...........................................................................................................17
    3.4.5 Message Exchange..................................................................................................................18
      3.4.5.1 HTTP and Caching Considerations..................................................................................19
      3.4.5.2 Security Considerations..................................................................................................19
    3.4.6 Error Reporting.....................................................................................................................20
    3.4.7 Metadata Considerations.......................................................................................................20
    3.4.8 Example SAML Message Exchange Using HTTP Redirect.............................................20
3.5 HTTP POST Binding...........................................................................................................21
  3.5.1 Required Information.....................................................................................................21
  3.5.2 Overview.....................................................................................................................21
  3.5.3 RelayState....................................................................................................................22
  3.5.4 Message Encoding.......................................................................................................22
  3.5.5 Message Exchange.......................................................................................................22
    3.5.5.1 HTTP and Caching Considerations.......................................................................23
    3.5.5.2 Security Considerations.......................................................................................24
  3.5.6 Error Reporting............................................................................................................24
  3.5.7 Metadata Considerations..............................................................................................24
  3.5.8 Example SAML Message Exchange Using HTTP POST...........................................24
3.6 HTTP Artifact Binding.......................................................................................................26
  3.6.1 Required Information.....................................................................................................26
  3.6.2 Overview.....................................................................................................................27
  3.6.3 Message Encoding.......................................................................................................27
    3.6.3.1 RelayState..............................................................................................................27
    3.6.3.2 URL Encoding.......................................................................................................27
    3.6.3.3 Form Encoding.......................................................................................................28
  3.6.4 Artifact Format.............................................................................................................28
    3.6.4.1 Required Information............................................................................................28
    3.6.4.2 Format Details........................................................................................................29
  3.6.5 Message Exchange.......................................................................................................29
    3.6.5.1 HTTP and Caching Considerations.......................................................................31
    3.6.5.2 Security Considerations.......................................................................................31
  3.6.6 Error Reporting............................................................................................................32
  3.6.7 Metadata Considerations..............................................................................................32
  3.6.8 Example SAML Message Exchange Using HTTP Artifact........................................32
3.7 SAML URI Binding............................................................................................................35
  3.7.1 Required Information.....................................................................................................35
  3.7.2 Protocol-Independent Aspects of the SAML URI Binding.........................................35
    3.7.2.1 Basic Operation.....................................................................................................35
  3.7.3 Security Considerations...............................................................................................36
  3.7.4 MIME Encapsulation..................................................................................................36
  3.7.5 Use of HTTP URIs........................................................................................................36
    3.7.5.1 URI Syntax.............................................................................................................36
    3.7.5.2 HTTP and Caching Considerations.......................................................................36
    3.7.5.3 Security Considerations.......................................................................................36
    3.7.5.4 Error Reporting....................................................................................................37
    3.7.5.5 Metadata Considerations......................................................................................37
  3.7.6 Example SAML Message Exchange Using an HTTP URI..........................................37
4 References..........................................................................................................................38
Appendix A. Registration of MIME media type application/samlassertion+xml....................40
Appendix B. Acknowledgments..............................................................................................44
Appendix C. Notices...............................................................................................................46
1 Introduction

This document specifies SAML protocol bindings for the use of SAML assertions and request-response messages in communications protocols and frameworks.

The SAML assertions and protocols specification [SAMLCore] defines the SAML assertions and request-response messages themselves, and the SAML profiles specification [SAMLProfile] defines specific usage patterns that reference both [SAMLCore] and bindings defined in this specification or elsewhere. The SAML conformance document [SAMLConform] lists all of the specifications that comprise SAML V2.0.

1.1 Protocol Binding Concepts

Mappings of SAML request-response message exchanges onto standard messaging or communication protocols are called SAML protocol bindings (or just bindings). An instance of mapping SAML request-response message exchanges into a specific communication protocol <FOO> is termed a <FOO> binding for SAML or a SAML <FOO> binding.

For example, a SAML SOAP binding describes how SAML request and response message exchanges are mapped into SOAP message exchanges.

The intent of this specification is to specify a selected set of bindings in sufficient detail to ensure that independently implemented SAML-conforming software can interoperate when using standard messaging or communication protocols.

Unless otherwise specified, a binding should be understood to support the transmission of any SAML protocol message derived from the \texttt{samlp:RequestAbstractType} and \texttt{samlp:StatusResponseType} types. Further, when a binding refers to "SAML requests and responses", it should be understood to mean any protocol messages derived from those types.

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

1.2 Notation

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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>XML Namespace</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>saml:</td>
<td>urn:oasis:names:tc:SAML:2.0:assertion</td>
<td>This is the SAML V2.0 assertion namespace [SAMLCore].</td>
</tr>
<tr>
<td>samlp:</td>
<td>urn:oasis:names:tc:SAML:2.0:protocol</td>
<td>This is the SAML V2.0 protocol namespace [SAMLCore].</td>
</tr>
<tr>
<td>Prefix</td>
<td>XML Namespace</td>
<td>Comments</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ds:</td>
<td><a href="http://www.w3.org/2000/09/xmldsig#">http://www.w3.org/2000/09/xmldsig#</a></td>
<td>This namespace is defined in the XML Signature Syntax and Processing specification [XMLSig] and its governing schema.</td>
</tr>
<tr>
<td>SOAP-ENV:</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope">http://schemas.xmlsoap.org/soap/envelope</a></td>
<td>This namespace is defined in SOAP V1.1 [SOAP11].</td>
</tr>
</tbody>
</table>

This specification uses the following typographical conventions in text: `<ns:Element>`, `XMLAttribute`, `Datatype`, `OtherKeyword`. In some cases, angle brackets are used to indicate non-terminals, rather than XML elements; the intent will be clear from the context.
2 Guidelines for Specifying Additional Protocol Bindings

This specification defines a selected set of protocol bindings, but others will possibly be developed in the future. It is not possible for the OASIS Security Services Technical Committee (SSTC) to standardize all of these additional bindings for two reasons: it has limited resources and it does not own the standardization process for all of the technologies used. This section offers guidelines for third parties who wish to specify additional bindings.

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

Following is a checklist of issues that MUST be addressed by each protocol binding:

1. Specify three pieces of identifying information: a URI that uniquely identifies the protocol binding, postal or electronic contact information for the author, and a reference to previously defined bindings or profiles that the new binding updates or obsoletes.

2. Describe the set of interactions between parties involved in the binding. Any restrictions on applications used by each party and the protocols involved in each interaction must be explicitly called out.

3. Identify the parties involved in each interaction, including how many parties are involved and whether intermediaries may be involved.

4. Specify the method of authentication of parties involved in each interaction, including whether authentication is required and acceptable authentication types.

5. Identify the level of support for message integrity, including the mechanisms used to ensure message integrity.

6. Identify the level of support for confidentiality, including whether a third party may view the contents of SAML messages and assertions, whether the binding requires confidentiality, and the mechanisms recommended for achieving confidentiality.

7. Identify the error states, including the error states at each participant, especially those that receive and process SAML assertions or messages.

8. Identify security considerations, including analysis of threats and description of countermeasures.

9. Identify metadata considerations, such that support for a binding involving a particular communications protocol or used in a particular profile can be advertised in an efficient and interoperable way.
3 Protocol Bindings

The following sections define the protocol bindings that are specified as part of the SAML standard.

3.1 General Considerations

The following sections describe normative characteristics of all protocol bindings defined for SAML.

3.1.1 Use of RelayState

Some bindings define a "RelayState" mechanism for preserving and conveying state information. When such a mechanism is used in conveying a request message as the initial step of a SAML protocol, it places requirements on the selection and use of the binding subsequently used to convey the response. Namely, 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 RelayState data it received with the request into the corresponding RelayState parameter in the response.

3.1.2 Security

Unless stated otherwise, these security statements apply to all bindings. Bindings may also make additional statements about these security features.

3.1.2.1 Use of SSL 3.0 or TLS 1.0

Unless otherwise specified, in any SAML binding's use of SSL 3.0 [SSL3] or TLS 1.0 [RFC2246], servers MUST authenticate to clients using a X.509 v3 certificate. The client MUST establish server identity based on contents of the certificate (typically through examination of the certificate’s subject DN field, subjectAltName attribute, etc.).

3.1.2.2 Data Origin Authentication

Authentication of both the SAML requester and the SAML responder associated with a message is OPTIONAL and depends on the environment of use. Authentication mechanisms available at the SOAP message exchange layer or from the underlying substrate protocol (for example in many bindings the SSL/TLS or HTTP protocol) MAY be utilized to provide data origin authentication.

Transport authentication will not meet end-end origin-authentication requirements in bindings where the SAML protocol message passes through an intermediary – in this case message authentication is recommended.

Note that SAML itself offers mechanisms for parties to authenticate to one another, but in addition SAML may use other authentication mechanisms to provide security for SAML itself.

3.1.2.3 Message Integrity

Message integrity of both SAML requests and SAML responses is OPTIONAL and depends on the environment of use. The security layer in the underlying substrate protocol or a mechanism at the SOAP message exchange layer MAY be used to ensure message integrity.

Transport integrity will not meet end-end integrity requirements in bindings where the SAML protocol message passes through an intermediary – in this case message integrity is recommended.
3.1.2.4 Message Confidentiality

Message confidentiality of both SAML requests and SAML responses is OPTIONAL and depends on the environment of use. The security layer in the underlying substrate protocol or a mechanism at the SOAP message exchange layer MAY be used to ensure message confidentiality.

Transport confidentiality will not meet end-end confidentiality requirements in bindings where the SAML protocol message passes through an intermediary.

3.1.2.5 Security Considerations

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.

IETF RFC 2617 [RFC2617] describes possible attacks in the HTTP environment when basic or message-digest authentication schemes are used.

Special care should be given to the impact of possible caching on security.

3.2 SAML SOAP Binding

SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment [SOAP11]. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation specific semantics. Two major design goals for SOAP are simplicity and extensibility. SOAP attempts to meet these goals by omitting, from the messaging framework, features that are often found in distributed systems. Such features include but are not limited to "reliability", "security", "correlation", "routing", and "Message Exchange Patterns" (MEPs).

A SOAP message is fundamentally a one-way transmission between SOAP nodes from a SOAP sender to a SOAP receiver, possibly routed through one or more SOAP intermediaries. SOAP messages are expected to be combined by applications to implement more complex interaction patterns ranging from request/response to multiple, back-and-forth "conversational" exchanges [SOAP-PRIMER].

SOAP defines an XML message envelope that includes header and body sections, allowing data and control information to be transmitted. SOAP also defines processing rules associated with this envelope and an HTTP binding for SOAP message transmission.

The SAML SOAP binding defines how to use SOAP to send and receive SAML requests and responses. Like SAML, SOAP can be used over multiple underlying transports. This binding has protocol-independent aspects, but also calls out the use of SOAP over HTTP as REQUIRED (mandatory to implement).

3.2.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:bindings:SOAP

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

Description: Given below.

Updates: urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding
3.2.2 Protocol-Independent Aspects of the SAML SOAP Binding

The following sections define aspects of the SAML SOAP binding that are independent of the underlying protocol, such as HTTP, on which the SOAP messages are transported. Note this binding only supports the use of SOAP 1.1.

3.2.2.1 Basic Operation

SOAP 1.1 messages consist of three elements: an envelope, header data, and a message body. SAML request-response protocol elements MUST be enclosed within the SOAP message body.

SOAP 1.1 also defines an optional data encoding system. This system is not used within the SAML SOAP binding. This means that SAML messages can be transported using SOAP without re-encoding from the "standard" SAML schema to one based on the SOAP encoding.

The system model used for SAML conversations over SOAP is a simple request-response model.

1. A system entity acting as a SAML requester transmits a SAML request element within the body of a SOAP message to a system entity acting as a SAML responder. The SAML requester MUST NOT include more than one SAML request per SOAP message or include any additional XML elements in the SOAP body.

2. The SAML responder MUST return either a SAML response element within the body of another SOAP message or generate a SOAP fault. The SAML responder MUST NOT include more than one SAML response per SOAP message or include any additional XML elements in the SOAP body. If a SAML responder cannot, for some reason, process a SAML request, it MUST generate a SOAP fault. SOAP fault codes MUST NOT be sent for errors within the SAML problem domain, for example, inability to find an extension schema or as a signal that the subject is not authorized to access a resource in an authorization query. (SOAP 1.1 faults and fault codes are discussed in [SOAP11] §4.1.)

On receiving a SAML response in a SOAP message, the SAML requester MUST NOT send a fault code or other error messages to the SAML responder. Since the format for the message interchange is a simple request-response pattern, adding additional items such as error conditions would needlessly complicate the protocol.

[SOAP11] references an early draft of the XML Schema specification including an obsolete namespace. SAML requesters SHOULD generate SOAP documents referencing only the final XML schema namespace. SAML responders MUST be able to process both the XML schema namespace used in [SOAP11] as well as the final XML schema namespace.

3.2.2.2 SOAP Headers

A SAML requester in a SAML conversation over SOAP MAY add arbitrary headers to the SOAP message. This binding does not define any additional SOAP headers.

Note: The reason other headers need to be allowed is that some SOAP software and libraries might add headers to a SOAP message that are out of the control of the SAML-aware process. Also, some headers might be needed for underlying protocols that require routing of messages or by message security mechanisms.

A SAML responder MUST NOT require any headers in the SOAP message in order to process the SAML message correctly itself, but MAY require additional headers that address underlying routing or message security requirements.

Note: The rationale is that requiring extra headers will cause fragmentation of the SAML standard and will hurt interoperability.
3.2.3 Use of SOAP over HTTP

A SAML processor that claims conformance to the SAML SOAP binding MUST implement SAML over SOAP over HTTP. This section describes certain specifics of using SOAP over HTTP, including HTTP headers, caching, and error reporting.

The HTTP binding for SOAP is described in [SOAP11] §6.0. It requires the use of a SOAPAction header as part of a SOAP HTTP request. A SAML responder MUST NOT depend on the value of this header. A SAML requester MAY set the value of the SOAPAction header as follows:

\[
\text{http://www.oasis-open.org/committees/security}
\]

3.2.3.1 HTTP Headers

A SAML requester in a SAML conversation over SOAP over HTTP MAY add arbitrary headers to the HTTP request. This binding does not define any additional HTTP headers.

**Note:** The reason other headers need to be allowed is that some HTTP software and libraries might add headers to an HTTP message that are out of the control of the SAML-aware process. Also, some headers might be needed for underlying protocols that require routing of messages or by message security mechanisms.

A SAML responder MUST NOT require any headers in the HTTP request to correctly process the SAML message itself, but MAY require additional headers that address underlying routing or message security requirements.

**Note:** The rationale is that requiring extra headers will cause fragmentation of the SAML standard and will hurt interoperability.

3.2.3.2 Caching

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

When using HTTP 1.1, requesters 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 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.

3.2.3.3 Error Reporting

A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD return a "403 Forbidden" response. In this case, the content of the HTTP body is not significant.

As described in [SOAP11] § 6.2, in the case of a SOAP error while processing a SOAP request, the SOAP HTTP server MUST return a "500 Internal Server Error" response and include a SOAP message in the response with a SOAP <SOAP-ENV:fault> element. This type of error SHOULD be returned for SOAP-related errors detected before control is passed to the SAML processor, or when the SOAP processor reports an internal error (for example, the SOAP XML namespace is incorrect, the SAML
schema cannot be located, the SAML processor throws an exception, and so on).

In the case of a SAML processing error, the SOAP HTTP server MUST respond with "200 OK" and include a SAML-specified <samlp:Status> element in the SAML response within the SOAP body. Note that the <samlp:Status> element does not appear by itself in the SOAP body, but only within a SAML response of some sort.

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

3.2.3.4 Metadata Considerations

Support for the SOAP binding SHOULD be reflected by indicating either a URL endpoint at which requests contained in SOAP messages for a particular protocol or profile are to be sent, or alternatively with a WSDL port/endpoint definition.

3.2.3.5 Example SAML Message Exchange Using SOAP over HTTP

Following is an example of a query that asks for an assertion containing an attribute statement from a SAML attribute authority.

```
POST /SamlService HTTP/1.1
Host: www.example.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   <SOAP-ENV:Body
     <samlp:AttributeQuery xmlns:samlp:="…" xmlns:saml="…” xmlns:ds="…” ID="_6c3a4f8b9c2d" Version="2.0"
     IssueInstant="2004-03-27T08:41:00Z"
     <ds:Signature> … </ds:Signature>
     <saml:Subject>
     …
     </saml:Subject>
     </samlp:AttributeQuery>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Following is an example of the corresponding response, which supplies an assertion containing the attribute statement as requested.

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   <SOAP-ENV:Body
     <samlp:Response xmlns:samlp="…" xmlns:saml="…” xmlns:ds="…” ID="_6c3a4f8b9c2d" Version="2.0"
     IssueInstant="2004-03-27T08:42:00Z"
     <saml:Issuer>https://www.example.com/SAML</saml:Issuer>
     <ds:Signature> … </ds:Signature>
     <Status>
     <StatusCode Value="…” />
     </Status>
     <saml:Assertion>
      <saml:Subject>
      …
      </saml:Subject>
      <saml:AttributeStatement>
      …
      </saml:AttributeStatement>
      </saml:Assertion>
```
3.3 Reverse SOAP (PAOS) Binding

This binding leverages the Reverse HTTP Binding for SOAP specification [PAOS]. Implementers MUST comply with the general processing rules specified in [PAOS] in addition to those specified in this document. In case of conflict, [PAOS] is normative.

3.3.1 Required Information


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

Description: Given below.

Updates: None.

3.3.2 Overview

The reverse SOAP binding is a mechanism by which an HTTP requester can advertise the ability to act as a SOAP responder or a SOAP intermediary to a SAML requester. The HTTP requester is able to support a pattern where a SAML request is sent to it in a SOAP envelope in an HTTP response from the SAML requester, and the HTTP requester responds with a SAML response in a SOAP envelope in a subsequent HTTP request. This message exchange pattern supports the use case defined in the ECP SSO profile (described in the SAML profiles specification [SAMLProfile]), in which the HTTP requester is an intermediary in an authentication exchange.

3.3.3 Message Exchange

The PAOS binding includes two component message exchange patterns:

1. The HTTP requester sends an HTTP request to a SAML requester. The SAML requester responds with an HTTP response containing a SOAP envelope containing a SAML request message.

2. Subsequently, the HTTP requester sends an HTTP request to the original SAML requester containing a SOAP envelope containing a SAML response message. The SAML requester responds with an HTTP response, possibly in response to the original service request in step 1.

The ECP profile uses the PAOS binding to provide authentication of the client to the service provider before the service is provided. This occurs in the following steps, illustrated in Figure A:

1. Client requests service using HTTP request.

2. Service Provider responds with a SAML authentication request. This is sent using a SOAP request, carried in the HTTP response.

3. The Client returns a SOAP response carrying a SAML authentication response. This is sent using a new HTTP request.

4. Assuming service provider authentication and authorization is successful the service provider may respond to the original service request in the HTTP response.
The HTTP requester advertises the ability to handle this reverse SOAP binding in its HTTP requests using the HTTP headers defined by the PAOS specification. Specifically:

- The HTTP Accept Header field MUST indicate an ability to accept the "text/vnd.paos+xml" content type.
- The HTTP PAOS Header field MUST be present and specify the PAOS version with "urn:liberty:paos:2003-08" at a minimum.

Additional PAOS headers such as the service value MAY be specified by profiles that use the PAOS binding. The HTTP requester MAY add arbitrary headers to the HTTP request.

Note that this binding does not define a RelayState mechanism. Specific profiles that make use of this binding must therefore define such a mechanism, if needed. The use of a SOAP header is suggested for this purpose.

The following sections provide more detail on the two steps of the message exchange.

### 3.3.3.1 HTTP Request, SAML Request in SOAP Response

In response to an arbitrary HTTP request, the HTTP responder MAY return a SAML request message using this binding by returning a SOAP 1.1 envelope in the HTTP response containing a single SAML request message in the SOAP body, with no additional body content. The SOAP envelope MAY contain arbitrary SOAP headers defined by PAOS, SAML profiles, or additional specifications.

Note that while the SAML request message is delivered to the HTTP requester, the actual intended...
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


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) 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 insure 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**


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 insure 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.

1. User Agent accesses some resource at the SAML Requester using an HTTP request
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 are exchanged using the HTTP Redirect binding.

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

```xml
<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: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
```
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%2Bh8k912TaDUGFUp7EbZQ6rxKb3mKcbvNjBOX%2FVxaxQ9tYecu0vH1v3nzkg1Z%2B1Af7YSf%2B8s8jhaqxB8d1Bu2gXkjkjcx1OpVEdoPPalo8vB8n3VI70eqttTlbJbbJBCOqc7a8j9XTBH9Vy
QhgyRbrTlxEl4Yo61oUQAqopShYZHIDQgks41ltAve2PqsAgNokr0as4zsc5521L1iJrTXI
BjVBr4wvcJ877jibcXZkmaRUxtk7CU7gqcB5mu8pKvdvgh4
2Ben9iDIMA3CXTtOse5BzFsdh%2FfnDK%2FEqg69Ye%2BUnvGl%2F8CfbQnBs%
2FQS3z4QLW9aT1oB1wa0j%2FGoYab9%2FV34Dw5k779IBAAA%
3D6RelayState=0043bfc1bc45110dael7004005b13a2b&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.
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.

### 3.5.3 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 insure 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.

### 3.5.4 Message Encoding

Messages are encoded for use with this binding by encoding the XML into an HTML form control and are transmitted using the HTTP POST method. A SAML protocol message is form-encoded by applying the base-64 encoding rules to the XML representation of the message and placing the result in a hidden form control within a form as defined by [HTML401] §17. The HTML document MUST adhere to the XHTML specification, [XHTML]. The base64-encoded value MAY be line-wrapped at a reasonable length in accordance with common practice.

If the message is a SAML request, then the form control MUST be named `SAMLRequest`. If the message is a SAML response, then the form control MUST be named `SAMLResponse`. Any additional form controls or presentation MAY be included but MUST NOT be required in order for the recipient to process the message.

If a “RelayState” value is to accompany the SAML protocol message, it MUST be placed in an additional hidden form control named `RelayState` within the same form with the SAML message.

The `action` attribute of the form MUST be the recipient's HTTP endpoint for the protocol or profile using this binding to which the SAML message is to be delivered. The `method` attribute MUST be "POST".

Any technique supported by the user agent MAY be used to cause the submission of the form, and any form content necessary to support this MAY be included, such as submit controls and client-side scripting commands. However, the recipient MUST be able to process the message without regard for the mechanism by which the form submission is initiated.

Note that any form control values included MUST be transformed so as to be safe to include in the XHTML document. This includes transforming characters such as quotes into HTML entities, etc.

### 3.5.5 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 responder are assumed to be HTTP responders. See the following diagram illustrating the messages exchanged.
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 an HTTP request from the user agent by returning a SAML request. The request is returned in an XHTML document containing the form and content defined in Section 3.5.4. The user agent delivers the SAML request by issuing an HTTP POST 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.

### 3.5.5.1 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.

### 3.5.5.2 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 or confidentiality protection and must authenticate the messages received instead. SAML provides for a signature on protocol messages for authentication and integrity for such cases. Form-encoded messages MAY be signed before the base64 encoding is applied.

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.

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.

Note also that there is no mechanism defined to protect the integrity of the relationship between the SAML protocol message and the "RelayState" value, if any. That is, an attacker can potentially recombine a pair of valid HTTP responses by switching the "RelayState" values associated with each SAML protocol message. The individual "RelayState" and SAML message values can be integrity protected, but not the combination. As a result, the producer and consumer of "RelayState" information MUST take care not to associate sensitive state information with the "RelayState" value without taking additional precautions (such as based on the information in the SAML message).

### 3.5.6 Error Reporting

A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD return a 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].

### 3.5.7 Metadata Considerations

Support for the HTTP POST 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.

### 3.5.8 Example SAML Message Exchange Using HTTP POST

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

First, here are the actual SAML protocol messages being exchanged:
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 SAML request message. The SAMLRequest parameter value is actually derived from the request message above.

HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/html; charset=iso-8859-1

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<body onload="document.forms[0].submit()">

<noscript>
<p>
<strong>Note:</strong> Since your browser does not support JavaScript, you must press the Continue button once to proceed.
</p>
</noscript>

<form action="https://ServiceProvider.com/SAML/SLO/Browser" method="post">
<div>
<input type="hidden" name="RelayState" value="0043bfc1bc45110dae17004005b13a2b"/>
<input type="hidden" name="SAMLRequest" value="PHNhbwxwOxKxv1c3QgeGlsbnM6c2FtbaHA9InVybipvYXNpczpuyW11
czp0YzpTU1MojjUaMDpwcm90b2NvbCIgeGlsbnM9InVybipvYXNpczpuyW11czp0
YzpTU1MojjUaMDphc3NlcnRpbi24iDQogICAqSUQ9IyMqYjdjMzg4ZVJvMzZmY2Zd
MsijJmhjZDI50YNGE4iPlY3c3NlZULuc3Rhbnp9i1WMDQtmDExmF1MTk6MDA6
NDIaJiaJWZVMjEaW9uPSY1lyfAfp06iclAIc2DxQc3NlZUIaXH0cHM6Ly9JZGVudGl0
eVBtb3P2CyVymVWv9TU1MjPc9j3c3NlZUIyZDhWJ0RgBoQt3tXyQ9
InVybipvYXNpczpuyW11czp0YzptTU1MojjUaMDpuYW1laWQtZm9ybWF0dNBlc3p0
3c3lnpqPqA6WVEwNM1uLWFkODIlTmEwZCIhNTU2LTAYMzlwM3yYjwvTmFt
ZULicpq6iclAIc2DxQc3Rhbnp9i1WMDQtmDExmF1MTk6MDA6
NDIaJiaJWZVMjEaW9uPSY1lyfAfp06iclAIc2DxQc3NlZUIaXH0cHM6Ly9JZGVudGl0
eVBtb3P2CyVymVWv9TU1MjPc9j3c3NlZUIyZDhWJ0RgBoQt3tXyQ9
InVybipvYXNpczpuyW11czp0YzptTU1MojjUaMDpuYW1laWQtZm9ybWF0dNBlc3p0
3c3lnpqPqA6WVEwNM1uLWFkODIlTmEwZCIhNTU2LTAYMzlwM3yYjwvTmFt
ZULicpq6iclAIc2DxQc3Rhbnp9i1WMDQtmDExmF1MTk6MDA6
NDIaJiaJWZVMjEaW9uPSY1lyfAfp06iclAIc2DxQc3NlZUIaXH0cHM6Ly9JZGVudGl0
"/>
</div>

</form>
After any unspecified interactions may have taken place, the SAML responder returns the HTTP response below containing the SAML response message. Again, the SAMLResponse parameter value is actually derived from the response message above.

```
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/html; charset=iso-8859-1
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<body onload="document.forms[0].submit()">
  <noscript>
    <p>
      <strong>Note:</strong> Since your browser does not support JavaScript,
you must press the Continue button once to proceed.
    </p>
  </noscript>
  <form action="https://IdentityProvider.com/SAML/SLO/Response"
    method="post">
    <div>
      <input type="hidden" name="RelayState"
        value="0043bfc1bc45110dae17004005b13a2b"/>
      <input type="hidden" name="SAMLResponse"
        value="PHNhbWxwOkxvZ291dFJlc3BvbnNlIHHtbG5zOnNhbwxwPJSJcm46b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaXM6bmFtZXJm6bmFtZXM6b2FzaX`
```

3.6 HTTP Artifact Binding

In the HTTP Artifact binding, the SAML request, the SAML response, or both are transmitted by reference using a small stand-in called an artifact. A separate, synchronous binding, such as the SAML SOAP binding, is used to exchange the artifact for the actual protocol message using the artifact resolution protocol defined in the SAML assertions and protocols specification [SAMLCore].

This binding MAY be composed with the HTTP Redirect binding (see Section 3.4) and the HTTP POST binding (see Section 3.5) to transmit request and response messages in a single protocol exchange using two different bindings.

3.6.1 Required Information

**Identification:** urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact
3.6.2 Overview

The HTTP Artifact binding is intended for cases in which the SAML requester and responder need to communicate using an HTTP user agent as an intermediary, but the intermediary's limitations preclude or discourage the transmission of an entire message (or message exchange) through it. This may be for technical reasons or because of a reluctance to expose the message content to the intermediary (and if the use of encryption is not practical).

Note that because of the need to subsequently resolve the artifact using another synchronous binding, such as SOAP, a direct communication path must exist between the SAML message sender and recipient in the reverse direction of the artifact's transmission (the receiver of the message and artifact must be able to send a <samlp:ArtifactResolve> request back to the artifact issuer). The artifact issuer must also maintain state while the artifact is pending, which has implications for load-balanced environments.

3.6.3 Message Encoding

There are two methods of encoding an artifact for use with this binding. One is to encode the artifact into a URL parameter and the other is to place the artifact in an HTML form control. When URL encoding is used, the HTTP GET method is used to deliver the message, while POST is used with form encoding. All endpoints that support this binding MUST support both techniques.

3.6.3.1 RelayState

RelayState data MAY be included with a SAML artifact 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 insure that the value has not been tampered with by using a checksum, a pseudo-random value, or similar means.

If an artifact that represents a SAML request 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 artifact into the corresponding RelayState parameter in the response.

If no such value is included with an artifact representing a SAML request, 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.

3.6.3.2 URL Encoding

To encode an artifact into a URL, the artifact value is URL-encoded and placed in a query string parameter named SAMLart.

If a “RelayState” value is to accompany the SAML artifact, it MUST be URL-encoded and placed in an additional query string parameter named RelayState.
3.6.3.3 Form Encoding

A SAML artifact is form-encoded by placing it in a hidden form control within a form as defined by [HTML401], chapter 17. The HTML document MUST adhere to the XHTML specification, [XHTML]. The form control MUST be named SAMLart. Any additional form controls or presentation MAY be included but MUST NOT be required in order for the recipient to process the artifact.

If a “RelayState” value is to accompany the SAML artifact, it MUST be placed in an additional hidden form control named RelayState, within the same form with the SAML message.

The action attribute of the form MUST be the recipient's HTTP endpoint for the protocol or profile using this binding to which the artifact is to be delivered. The method attribute MUST be set to "POST".

Any technique supported by the user agent MAY be used to cause the submission of the form, and any form content necessary to support this MAY be included, such as submit controls and client-side scripting commands. However, the recipient MUST be able to process the artifact without regard for the mechanism by which the form submission is initiated.

Note that any form control values included MUST be transformed so as to be safe to include in the XHTML document. This includes transforming characters such as quotes into HTML entities, etc.

3.6.4 Artifact Format

With respect to this binding, an artifact is a short, opaque string. Different types can be defined and used without affecting the binding. The important characteristics are the ability of an artifact receiver to identify the issuer of the artifact, resistance to tampering and forgery, uniqueness, and compactness.

The general format of any artifact includes a mandatory two-byte artifact type code and a two-byte index value identifying a specific endpoint of the artifact resolution service of the issuer, as follows:

```
SAML_artifact := B64(TypeCode EndpointIndex RemainingArtifact)
TypeCode := Byte1Byte2
EndpointIndex := Byte1Byte2
```

The notation B64(TypeCode EndpointIndex RemainingArtifact) stands for the application of the base64 [RFC2045] transformation to the catenation of the TypeCode, EndpointIndex, and RemainingArtifact.

The following practices are RECOMMENDED for the creation of SAML artifacts:

- Each issuer is assigned an identifying URI, also known as the issuer's entity (or provider) ID. See Section 8.3.6 of [SAMLCore] for a discussion of this kind of identifier.
- The issuer constructs the SourceID component of the artifact by taking the SHA-1 hash of the identification URL. The hash value is NOT encoded into hexadecimal.
- The MessageHandle value is constructed from a cryptographically strong random or pseudorandom number sequence [RFC1750] generated by the issuer. The sequence consists of values of at least 16 bytes in size. These values should be padded as needed to a total length of 20 bytes.

The following describes the single artifact type defined by SAML V2.0.

3.6.4.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:artifact-04

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

Description: Given below.
3.6.4.2 Format Details

SAML V2.0 defines an artifact type of type code 0x0004. This artifact type is defined as follows:

\[
\begin{array}{l}
\text{TypeCode} := 0x0004 \\
\text{RemainingArtifact} := \text{SourceID} \; \text{MessageHandle} \\
\text{SourceID} := 20\text{-byte_sequence} \\
\text{MessageHandle} := 20\text{-byte_sequence}
\end{array}
\]

SourceID is a 20-byte sequence used by the artifact receiver to determine artifact issuer identity and the set of possible resolution endpoints.

It is assumed that the destination site will maintain a table of SourceID values as well as one or more indexed URL endpoints (or addresses) for the corresponding SAML responder. The SAML metadata specification [SAMLMeta] MAY be used for this purpose. On receiving the SAML artifact, the receiver determines if the SourceID belongs to a known artifact issuer and obtains the location of the SAML responder using the EndpointIndex before sending a SAML <samlp:ArtifactResolve> message to it.

Any two artifact issuers with a common receiver MUST use distinct SourceID values. Construction of MessageHandle values is governed by the principle that they SHOULD have no predictable relationship to the contents of the referenced message at the issuing site and it MUST be infeasible to construct or guess the value of a valid, outstanding message handle.

3.6.5 Message Exchange

The system model used for SAML conversations by means of this binding is a request-response model in which an artifact reference takes the place of the actual message content, and the artifact reference is 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 responder are assumed to be HTTP responders.

Additionally, it is assumed that on receipt of an artifact by way of the user agent, the recipient invokes a separate, direct exchange with the artifact issuer using the Artifact Resolution Protocol defined in [SAMLCore]. This exchange MUST use a binding that does not use the HTTP user agent as an intermediary, such as the SOAP binding. On the successful acquisition of a SAML protocol message, the artifact is discarded and the processing of the primary SAML protocol exchange resumes (or ends, if the message is a response).

Issuing and delivering an artifact, along with the subsequent resolution step, constitutes half of the overall SAML protocol exchange. This binding can be used to deliver either or both halves of a SAML protocol exchange. A binding composable with it, such as the HTTP Redirect (see Section 3.4) or POST (see Section 3.5) binding, MAY be used to carry the other half of the exchange. The following sequence assumes that the artifact binding is used for both halves. See the diagram below illustrating the messages exchanged.
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 an HTTP request from the user agent by returning an artifact representing a SAML request.

   • If URL-encoded, the artifact 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 accesses some resource at the SAML Requester using an HTTP request.

3. `<ArtifactResolve>` message sent by SAML Responder directly to SAML Requester.

4. `<ArtifactResponse>` message returned by SAML Requester containing original SAML request message inside.

5. SAML Responder interacts with User Agent, subject to constraints in the SAML request.

6. SAML artifact returned in HTTP Redirect URL encoded into Location header or XHTML form control targeted at SAML Requester.

7. `<ArtifactResolve>` message sent by SAML Requester directly to SAML Responder.

8. `<ArtifactResponse>` message returned by SAML Responder containing original SAML response message inside.

9. HTTP response sent to user agent from SAML requester upon completion of SAML exchange.
agent delivers the artifact by issuing an HTTP GET request to the SAML responder.

- If form-encoded, then the artifact is returned in an XHTML document containing the form and content defined in Section 3.6.3.3. The user agent delivers the artifact by issuing an HTTP POST request to the SAML responder.

3. The SAML responder determines the SAML requester by examining the artifact (the exact process depends on the type of artifact), and issues a `<samlp:ArtifactResolve>` request containing the artifact to the SAML requester using a direct SAML binding, temporarily reversing roles.

4. Assuming the necessary conditions are met, the SAML requester returns a `<samlp:ArtifactResponse>` containing the original SAML request message it wishes the SAML responder to process.

5. In general, the SAML responder MAY respond to the SAML request by immediately returning a SAML artifact 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>`).

6. Eventually the responder SHOULD return a SAML artifact to the user agent to be returned to the SAML requester. The SAML response artifact is returned in the same fashion as described for the SAML request artifact in step 2. The SAML requester determines the SAML responder by examining the artifact, and issues a `<samlp:ArtifactResolve>` request containing the artifact to the SAML responder using a direct SAML binding, as in step 3.

7. Assuming the necessary conditions are met, the SAML responder returns a `<samlp:ArtifactResponse>` containing the SAML response message it wishes the requester to process, as in step 4.

8. Upon receiving the SAML response, the SAML requester returns an arbitrary HTTP response to the user agent.

### 3.6.5.1 HTTP and Caching Considerations

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

When returning SAML artifacts 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.

### 3.6.5.2 Security Considerations

This binding uses a combination of indirect transmission of a message reference followed by a direct exchange to return the actual message. As a result, the message reference (artifact) need not itself be authenticated or integrity protected, but the callback request/response exchange that returns the actual message MAY be mutually authenticated and integrity protected, depending on the environment of use.

If the actual SAML protocol message is intended for a specific recipient, then the artifact's issuer MUST authenticate the sender of the subsequent `<samlp:ArtifactResolve>` message before returning the actual message.

The transmission of an artifact to and from the user agent SHOULD be protected with confidentiality; SSL 3.0 [SSL3] or TLS 1.0 [RFC2246] SHOULD be used. The callback request/response exchange that returns the actual message MAY be protected, depending on the environment of use.
In general, this binding relies on the artifact as a hard-to-forge short-term reference and applies other
security measures to the callback request/response that returns the actual message. All artifacts MUST
have a single-use semantic enforced by the artifact issuer.

Furthermore, it is RECOMMENDED that artifact receivers also enforce a single-use semantic on the
artifact values they receive, to prevent an attacker from interfering with the resolution of an artifact by a
user agent and then resubmitting it to the artifact receiver. If an attempt to resolve an artifact does not
complete successfully, the artifact SHOULD be placed into a blocked artifact list for a period of time that
exceeds a reasonable acceptance period during which the artifact issuer would resolve the artifact.

Note also that there is no mechanism defined to protect the integrity of the relationship between the
artifact and the "RelayState" value, if any. That is, an attacker can potentially recombine a pair of valid
HTTP responses by switching the "RelayState" values associated with each artifact. As a result, the
producer/consumer of "RelayState" information MUST take care not to associate sensitive state
information with the "RelayState" value without taking additional precautions (such as based on the
information in the SAML protocol message retrieved via artifact).

### 3.6.6 Error Reporting

A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD
return a response message with a second-level `<samlp:StatusCode>` value of

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.

If the issuer of an artifact receives a `<samlp:ArtifactResolve>` message that it can understand, it
MUST return a `<samlp:ArtifactResponse>` with a `<samlp:StatusCode>` value of
urn:oasis:names:tc:SAML:2.0:status:Success, even if it does not return the corresponding
message (for example because the artifact requester is not authorized to receive the message or the
artifact is no longer valid).

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

### 3.6.7 Metadata Considerations

Support for the HTTP Artifact 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. One or more indexed endpoints for processing
<samlp:ArtifactResolve> messages SHOULD also be described.

### 3.6.8 Example SAML Message Exchange Using HTTP Artifact

In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair are exchanged using the
HTTP Artifact binding, with the artifact resolution taking place using the SOAP binding bound to HTTP.

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

```xml
<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>
```
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 SAML artifact.

Note that the line feeds in the HTTP Location header below are a result of document formatting, 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?SAMLart=AAQAADWNEw5VT47wc04zX%iEzMmFqGknDfws2ZtgSgkNbsW1cmVR0buzU%
X%2F1Mf1MqGknDfws2ZtgSgkNbsW1cmVR0buzU%
3DRelayState=0043bcf1bc45110dea17004005b13a2b
Content-Type: text/html; charset=iso-8859-1

The SAML responder then resolves the artifact it received into the actual SAML request using the Artifact Resolution protocol and the SOAP binding in steps 3 and 4, as follows:

Step 3:
POST /SAML/Artifact/Resolve HTTP/1.1
Host: IdentityProvider.com
Content-Type: text/xml
Content-Length: mnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<samlp:ArtifactResolve
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
ID="_6c3a4f8b9c2d" Version="2.0"
IssueInstant="2004-01-21T19:00:49Z">
<Issuer>
https://ServiceProvider.com/SAML
</Issuer>
<Artifact>
AAQAADWNEw5VT47wc04zX/iEzMmFqGknDfws2ZtgSgkNbsW1cmVR0buzU%
</Artifact>
</samlp:ArtifactResolve>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Step 4:
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:00:49 GMT
Content-Type: text/xml
Content-Length: mnnn
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<samlp:ArtifactResponse
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
ID="_FQvGknDfws2ZtgSgkNbsW1cmVR0buzU%
InResponseTo="_6c3a4f8b9c2d"
IssueInstant="2004-01-21T19:00:49Z">
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
After any unspecified interactions may have taken place, the SAML responder returns a second SAML artifact in its HTTP response in step 6:

The SAML responder then resolves the artifact it received into the actual SAML request using the Artifact Resolution protocol and the SOAP binding in steps 7 and 8, as follows:

**Step 7:**

```plaintext
POST /SAML/Artifact/Resolve HTTP/1.1
Host: ServiceProvider.com
Content-Type: text/xml
Content-Length: nnn

<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
   ID="_ec36fa7c39" Version="2.0"
   IssueInstant="2004-01-21T19:05:49Z">
  <Issuer>https://IdentityProvider.com/SAML</Issuer>
  <Artifact>AAQAAFGIZXx5+QaBaE5qYurHWJ01nAgLaqrfn1DH1ggbFUGmlSGFTyQifc</Artifact>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Step 8:**

```plaintext
HTTP/1.1 200 OK
Date: 21 Jan 2004 07:05:49 GMT
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
   ID="_FQvGknDfws2Z" InResponseTo="_ec36fa7c39"
   IssueInstant="2004-01-21T19:05:49Z">
  <Issuer>https://ServiceProvider.com/SAML</Issuer>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
3.7 SAML URI Binding

URIs are a protocol-independent means of referring to a resource. This binding is not a general SAML request/response binding, but rather supports the encapsulation of a \(<\text{samlp:AssertionIDRequest}>\) message with a single \(<\text{saml:AssertionIDRef}>\) into the resolution of a URI. The result of a successful request is a SAML \(<\text{saml:Assertion}>\) element (but not a complete SAML response).

Like SOAP, URI resolution can occur over multiple underlying transports. This binding has transport-independent aspects, but also calls out the use of HTTP with SSL 3.0 [SSL3] or TLS 1.0 [RFC2246] as REQUIRED (mandatory to implement).

3.7.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:bindings:URI

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

Description: Given below.

Updates: None

3.7.2 Protocol-Independent Aspects of the SAML URI Binding

The following sections define aspects of the SAML URI binding that are independent of the underlying transport protocol of the URI resolution process.

3.7.2.1 Basic Operation

A SAML URI reference identifies a specific SAML assertion. The result of resolving the URI MUST be a message containing the assertion, or a transport-specific error. The specific format of the message depends on the underlying transport protocol. If the transport protocol permits the returned content to be described, such as HTTP 1.1 [RFC2616], then the assertion MAY be encoded in whatever format is permitted. If not, the assertion MUST be returned in a form which can be unambiguously interpreted as or transformed into an XML serialization of the assertion.

It MUST be the case that if the same URI reference is resolved in the future, then either the same SAML assertion, or an error, is returned. That is, the reference MAY be persistent but MUST consistently reference the same assertion, if any.
3.7.3 Security Considerations

Indirect use of a SAML assertion presents dangers if the binding of the reference to the result is not secure. The particular threats and their severity depend on the use to which the assertion is being put. In general, the result of resolving a URI reference to a SAML assertion SHOULD only be trusted if the requester can be certain of the identity of the responder and that the contents have not been modified in transit.

It is often not sufficient that the assertion itself be signed, because URI references are by their nature somewhat opaque to the requester. The requester SHOULD have independent means to ensure that the assertion returned is actually the one that is represented by the URI; this is accomplished by both authenticating the responder and relying on the integrity of the response.

3.7.4 MIME Encapsulation

For resolution protocols that support MIME as a content description and packaging mechanism, the resulting assertion SHOULD be returned as a MIME entity of type application/samlassertion+xml, as defined by [SAMLmime].

3.7.5 Use of HTTP URIs

A SAML authority that claims conformance to the SAML URI binding MUST implement support for HTTP. This section describes certain specifics of using HTTP URIs, including URI syntax, HTTP headers, and error reporting.

3.7.5.1 URI Syntax

In general, there are no restrictions on the permissible syntax of a SAML URI reference as long as the SAML authority responsible for the reference creates the message containing it. However, authorities MUST support a URL endpoint at which an HTTP request can be sent with a single query string parameter named ID. There MUST be no query string in the endpoint URL itself independent of this parameter.

For example, if the documented endpoint at an authority is "https://saml.example.edu/assertions", a request for an assertion with an ID of abcde can be sent to:

https://saml.example.edu/assertions?ID=abcde

Note that the use of wildcards is not allowed for such ID queries.

3.7.5.2 HTTP and Caching Considerations

HTTP proxies MUST NOT cache SAML assertions. To ensure this, the following rules SHOULD be followed.

When returning SAML assertions 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".

3.7.5.3 Security Considerations

RFC 2617 [RFC2617] describes possible attacks in the HTTP environment when basic or message-digest authentication schemes are used.

Use of SSL 3.0 [SSL3] or TLS 1.0 [RFC2246] is STRONGLY RECOMMENDED as a means of authentication, integrity protection, and confidentiality.
3.7.4 Error Reporting

As an HTTP protocol exchange, the appropriate HTTP status code SHOULD be used to indicate the result of a request. For example, a SAML responder that refuses to perform a message exchange with the SAML requester SHOULD return a "403 Forbidden" response. If the assertion specified is unknown to the responder, then a "404 Not Found" response SHOULD be returned. In these cases, the content of the HTTP body is not significant.

3.7.5 Metadata Considerations

Support for the URI binding over HTTP SHOULD be reflected by indicating a URL endpoint at which requests for arbitrary assertions are to be sent.

3.7.6 Example SAML Message Exchange Using an HTTP URI

Following is an example of a request for an assertion.

```
GET /SamlService?ID=abcde HTTP/1.1
Host: www.example.com
```

Following is an example of the corresponding response, which supplies the requested assertion.

```
HTTP/1.1 200 OK
Content-Type: application/samlassertion+xml
Cache-Control: no-cache, no-store
Pragma: no-cache
Content-Length: nnnn

<saml:Assertion ID="abcde" ...>
...
</saml:Assertion>
```
4 References


XHTML 1.0 The Extensible HyperText Markup Language (Second Edition), http://www.w3.org/TR/xhtml1/.

Appendix A. Registration of MIME media type application/samlassertion+xml

Introduction

The IESG has approved a request to register the application/samlassertion+xml MIME media type in the standards tree. This media type is a product of the Organization for the Advancement of Structured Information Systems (OASIS).

The IESG contact persons are Ted Hardie and Scott Hollenbeck.

MIME media type name

application

MIME subtype name

samlassertion+xml

Required parameters

None

Optional parameters

charset

Same as charset parameter of application/xml [RFC3023].

Encoding considerations

Same as for application/xml [RFC3023].

Security considerations

Per their specification, samlassertion+xml-typed objects do not contain executable content. However, SAML assertions are XML-based objects [XML]. As such, they have all of the general security considerations presented in section 10 of [RFC3023], as well as additional ones, since they are explicit security objects. For example, samlassertion+xml-typed objects will often contain data that may identify or pertain to a natural person, and may be used as a basis for sessions and access control decisions.

To counter potential issues, samlassertion+xml-typed objects contain data that should be signed appropriately by the sender. Any such signature must be verified by the recipient of the data - both as a valid signature, and as being the signature of the sender. Issuers of samlassertion+xml-typed objects containing SAMLv2 assertions may also encrypt all, or portions of, the assertions (see [SAMLv2Core]).

In addition, SAML profiles and protocol bindings specify use of secure channels as appropriate.

[SAMLv2.0] incorporates various privacy-protection techniques in its design. For example: opaque handles, specific to interactions between specific system entities, may be assigned to subjects. The handles are mappable to wider-context identifiers (e.g. email addresses, account identifiers, etc) by only the specific parties.
For a more detailed discussion of SAML security considerations and specific security-related
design techniques, please refer to the SAML specifications listed in the below bibliography. The
specifications containing security-specific information have been explicitly listed for each version
of SAML.

Interoperability considerations
SAML assertions are explicitly versioned. Relying parties should ensure that they observe
assertion version information and behave accordingly. See chapters on SAML Versioning in
[SAMLv1Core], [SAMLv11Core], or [SAMLv2Core], as appropriate.

Published specification
[SAMLv2Bind] explicitly specifies use of the application/samlassertion+xml MIME media
type. However, it is conceivable that non-SAMLv2 assertions (i.e., SAMLv1 and/or SAMLv1.1)
might in practice be conveyed using SAMLv2 bindings.

Applications which use this media type
Potentially any application implementing SAML, as well as those applications implementing
specifications based on SAML, e.g. those available from the Liberty Alliance [LAP].

Additional information
Magic number(s)
In general, the same as for application/xml [RFC3023]. In particular, the XML root element of the
returned object will be <saml:Assertion>, where "saml" maps to a version-specific SAML
assertion namespace, as defined by the appropriate SAML "core" specification (see bibliography).
In the case of SAMLv2.0, the root element of the returned object may be either
<saml:Assertion> or <saml:EncryptedAssertion>, where "saml" maps to the SAMLv2.0
assertion namespace URI:

urn:oasis:names:tc:SAML:2.0:assertion

File extension(s)
None

Macintosh File Type Code(s)
None

Person & email address to contact for further information
This registration is made on behalf of the OASIS Security Services Technical Committee (SSTC)
Please refer to the SSTC website for current information on committee chairperson(s) and their
contact addresses: http://www.oasis-open.org/committees/security/. Committee members should
submit comments and potential errata to the security-services@lists.oasis-open.org list. Others
should submit them by filling out the web form located at http://www.oasis-
open.org/committees/comments/form.php?wg_abbrev=security.

Additionally, the SAML developer community email distribution list, saml-dev@lists.oasis-
open.org, may be employed to discuss usage of the application/samlassertion+xml MIME media
type. The "saml-dev" mailing list is publicly archived here: http://lists.oasis-
open.org/archives/saml-dev/. To post to the "saml-dev" mailing list, one must subscribe to it. To
subscribe, send a message with the single word "subscribe" in the message body, to: saml-dev-request@lists.oasis-open.org.

**Intended usage**

COMMON

**Author/Change controller**

The SAML specification sets are a work product of the OASIS Security Services Technical Committee (SSTC). OASIS and the SSTC have change control over the SAML specification sets.

**Bibliography**

[LAP] "Liberty Alliance Project". See http://www.projectliberty.org/

[OASIS] "Organization for the Advancement of Structured Information Systems".
See http://www.oasis-open.org/


open.org/committees/security/


Appendix B. Acknowledgments

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

- Conor Cahill, AOL
- John Hughes, Atos Origin
- Hal Lockhart, BEA Systems
- Mike Beach, Boeing
- Rebekah Metz, Booz Allen Hamilton
- Rick Randall, Booz Allen Hamilton
- Ronald Jacobson, Computer Associates
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems
- Irving Reid, Hewlett-Packard
- Paula Austel, IBM
- Michael McIntosh, IBM
- Anthony Nadalin, IBM
- Nick Ragouzis, Individual
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Peter Davis, Neustar
- Jeff Hodges, Neustar
- Frederick Hirsch, Nokia
- Senthil Sengodan, Nokia
- Abbie Barbir, Nortel Networks
- Scott Kiester, Novell
- Cameron Morris, Novell
- Paul Madsen, NTT
- Steve Anderson, OpenNetwork
- Ari Kermaier, Oracle
- Vamsi Motukuru, Oracle
- Darren Platt, Ping Identity
- Prateek Mishra, Principal Identity
- Jim Lien, RSA Security
- John Linn, RSA Security
- Rob Philpott, RSA Security
- Dipak Chopra, SAP
- Jahan Moreh, Sigaba
- Bhavna Bhatnagar, Sun Microsystems
- Eve Maler, Sun Microsystems
- Ronald Monzillo, Sun Microsystems
- Emily Xu, Sun Microsystems
- Greg Whitehead, Trustgenix
The editors also would like to acknowledge the following people for their contributions to previous versions of the OASIS Security Assertions Markup Language Standard:

- Stephen Farrell, Baltimore Technologies
- David Orchard, BEA Systems
- Krishna Sankar, Cisco Systems
- Zahid Ahmed, CommerceOne
- Carlisle Adams, Entrust
- Tim Moses, Entrust
- Nigel Edwards, Hewlett-Packard
- Joe Pato, Hewlett-Packard
- Bob Blakley, IBM
- Marlena Erdos, IBM
- Marc Chanliau, Netegrity
- Chris McLaren, Netegrity
- Lynne Rosenthal, NIST
- Mark Skall, NIST
- Simon Godik, Overxeer
- Charles Norwood, SAIC
- Evan Prodromou, Securant
- Robert Griffin, RSA Security (former editor)
- Sai Allarvarpu, Sun Microsystems
- Chris Ferris, Sun Microsystems
- Mike Myers, Traceroute Security
- Phillip Hallam-Baker, VeriSign (former editor)
- James Vanderbeek, Vodafone
- Mark O’Neill, Vordel
- Tony Palmer, Vordel

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

- Thomas Gross, IBM
- Birgit Pfizmann, IBM
Appendix C. Notices

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

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

Copyright © OASIS Open 2005. All Rights Reserved.

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

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

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