SAMLv2: HTTP POST “NoXMLdsig” Binding

Draft, xx June 2006

Document identifier:

draft-hodges-saml-binding-no-xmldsig-00

Location:

Editors:

Jeff Hodges, NeuStar
Scott Cantor, Internet2

Abstract:

This specification defines a SAML HTTP protocol binding, specifically using the HTTP POST method, and not using XML Digital Signature for SAML message and/or SAML assertion data origination authentication. Rather, a “sign the BLOB” technique is employed wherein a conveyed SAML message, along with any content (e.g. SAML assertions) is treated as a simple octet string if it is signed. Security is optional in this binding.

Status:

This is an individually-authored working draft published to the Security Services Technical Committee (SSTC/SAML), and has no official standing.

Committee members should submit comments to the security-services@lists.oasis-open.org list. Non-committee members who wish to comment may do so on the SAML-dev@lists.oasis-open.org mailing list (one must be a list subscriber to post. To subscribe, send mail to mailto:saml-dev-subscribe@lists.oasis-open.org).

Table of Contents

1 Introduction

This specification defines a SAML HTTP protocol binding, specifically using the HTTP POST method, and not using XML Digital Signature for SAML message and/or SAML assertion data origination authentication. Rather, a “sign the BLOB” technique is employed wherein a conveyed SAML message, along with any content (e.g. SAML assertions) is treated as a simple octet string if it is signed. Security is optional in this binding. The next subsection gives a general overview of SAML Protocol Binding concepts.

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 **samlp:RequestAbstractType** and **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].

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

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

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

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

Prefix	XML Namespace	Comments
`saml:`	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore].
`samlp:`	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore].
`SOAP-ENV:`	http://schemas.xmlsoap.org/soap/envelope	This namespace is defined in SOAP V1.1 [SOAP11].

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 HTTP POST Binding - NoXMLdsig

The HTTP POST binding, defined in [SAML20Bind],defines a mechanism by which SAML protocol messages may be transmitted within the base64-encoded content of an HTML form control. When using that binding, SAML protocol messages and/or SAML assertions are signed using [XMLSig], which is an XML-aware, XML-based, invasive digital signature paradigm necessitating canonicalization of the signature target.

This document specifies an alternative HTTP POST binding where the conveyed SAML protocol messages, and their content – i.e. any conveyed SAML assertions – are signed as simple “blobs” (“binary large objects”, aka binary octet strings).

This binding MAY be composed with the HTTP Redirect binding (see Section 3.4 of [SAML20Bind]) and the HTTP Artifact binding (see Section 3.6 of [SAML20Bind]) to transmit request and response messages in a single protocol exchange using two different bindings.

2.0.1 Required Information

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

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

Description: Given below.

Updates: None. Rather, it provides an alternative to the HTTP POST Binding defined in [SAML20Bind]

2.0.2 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.

2.0.3 RelayState

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

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

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

2.0.4 Message Encoding

This section describes how to encode SAML messages, and thus any SAML assertions they may contain, into HTML FORM “controls” [HTML401] (Section 17), thus enabling the SAML messages to be transmitted via the HTTP POST method.

A SAML protocol message is form-encoded by:

1. Applying the base-64 encoding rules to the XML representation of the message. The resulting base64-encoded value MAY be line-wrapped at a reasonable length in accordance with common practice.
2. Encoding the result from the prior step into a “form data set”, in the same fashion as is specified for “successful controls” in [HTML401] (Section 17.13.3), as a form “control value”. The HTML document also MUST adhere to the XHTML specification, [XHTML].
    1 If the message is a SAML request, then the form “control name” used to convey the SAML message itself MUST be `SAMLRequest`.
    2 If the message is a SAML response, then the form “control name” used to convey the SAML message itself MUST be `SAMLResponse`.
    3 Any additional form controls or presentation, other than those noted below for including a signature, MAY be included but MUST NOT be required in order for the recipient to nominally process the message itself.

SAML messages, and any SAML assertions conveyed by said messages in this binding, MUST NOT be signed using [XMLSig].

> If they are so signed before being processed as defined herein, their XML digital signatures MUST be removed as described in [SAML20Bind] section 3.4.4.1 step 1.

Rather, if a SAML message is to be signed – which this binding leaves as a decision of the implementor and/or deployer – it MUST be signed using the technique given below in section 2.0.5. The resultant signature value is conveyed in a form control value named `MsgSig`, and the signature algorithm is conveyed in a form control value named `SigAlg`, These form control values are included in the form data set constructed in step 2 above.

If the message is signed, the `Destination` XML attribute in the root SAML element of the SAML 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.

If a “RelayState” value is to accompany the SAML protocol message, it MUST be in a form control named `RelayState`, and included in the form data set constructed in step 2 above, and also included in any signed content if the message is signed.

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". The enctype attribute specifies the form content type and MUST be `application/x-www-form-urlencoded`.

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.

2.0.5 Signature

To construct a signature of a SAML message conveyed by this binding:

1. The signature algorithm used MUST be identified by a URI, specified according to [XMLSig] or whatever specification governs the algorithm. 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

2. A string consisting of the concatenation of the `RelayState` (if present), `SigAlg`, and `SAMLRequest` (or `SAMLResponse`) values (as appropriate), as defined in section 2.0.4 above, is constructed in one of the following ways (each individually ordered as shown):

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

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

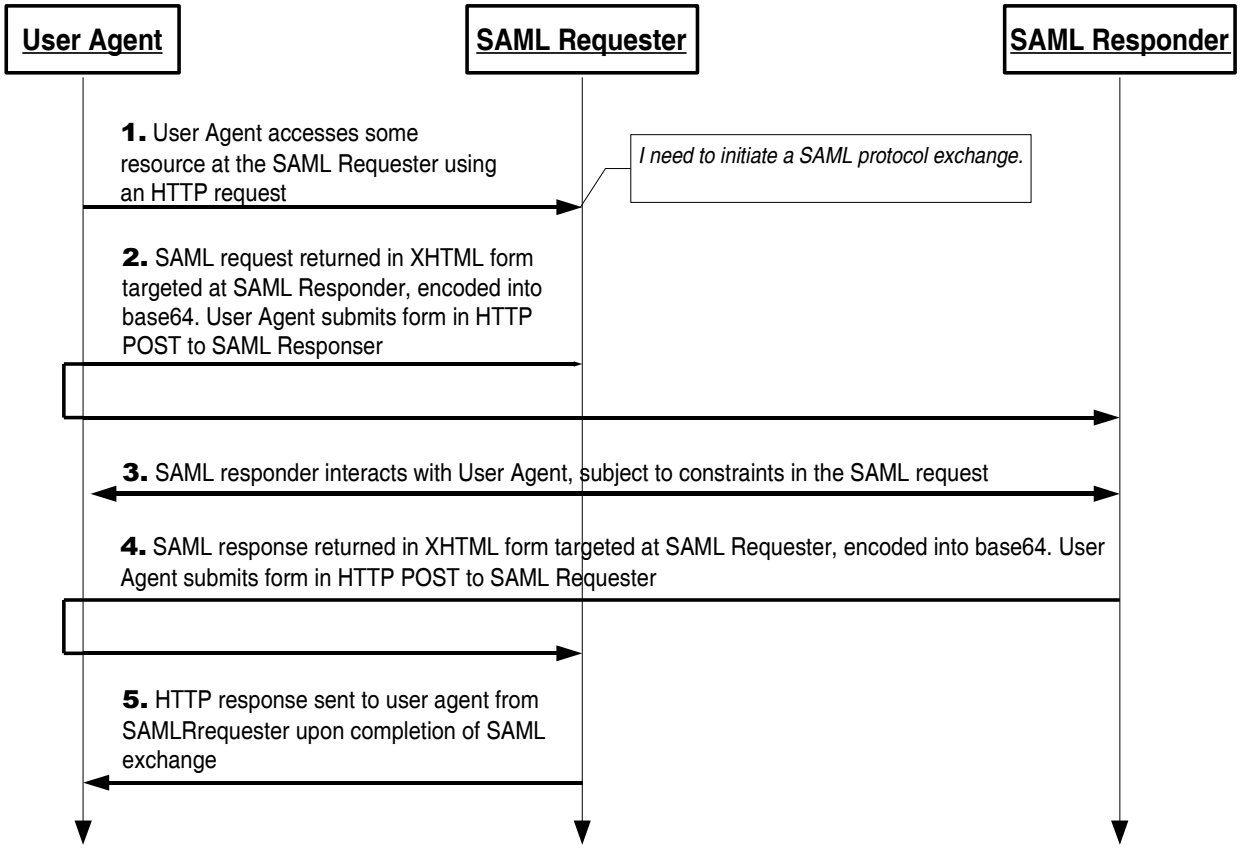
3. The resultant octet string is fed into the signature algorithm.
4. The value yielded by the signature algorithm is base64 encoded (see RFC 2045 [RFC2045]) with any whitespace removed, and used as the value for the MsgSig form control as discussed in section 2.0.4, above.

2.0.6 Signature Verification

To verify a received signed SAML message conveyed by this binding, the receiver MUST extract the form control values for the `RelayState` (if present), `SigAlg`, and `SAMLRequest` (or `SAMLResponse`) values (as appropriate) from the received HTTP message. Then the receiver reconstructs the string as described in section 2.0.5 step 2, above. The signature value conveyed in the MsgSig control value is then checked against this string per the signature algorithm given by the SigAlg control value. Error handling and generated messages as a result of the signature not verifying are implementation-dependent.

2.0.7 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 2.0.4, above. 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 it 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.

2.0.7.1 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.

2.0.7.2 Security Considerations

The presence of the user agent intermediary means that the requester and responder cannot rely on the transport layer for endpoint-to-endpoint (i.e. SAML Requester to/from SAML Responder) authentication, integrity or confidentiality protection. Instead, this binding defines a means for signing the conveyed SAML messages and optional `RelayState`.in order to provide endpoint-to-endpoint integrity protection and data origin authentication.

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.

2.0.8 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].

2.0.9 Metadata Considerations

@@TODO: any fixups needed here?

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.

2.0.10 Example SAML Message Exchange Using HTTP POST - NoXMLdsig

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

@@TODO: update below examples as necessary

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

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

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

The initial HTTP request from the user agent in step 1 is not defined by this binding. To initiate the logout protocol exchange, the SAML requester returns the following HTTP response, containing a 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="PHNhbWxwOkxvZ291dFJlcXVlc3QgeG1sbnM6c2FtbHA9InVybjpvYXNpczpuYW1l
czp0YzpTQU1MOjIuMDpwcm90b2NvbCIgeG1sbnM9InVybjpvYXNpczpuYW1lczp0
YzpTQU1MOjIuMDphc3NlcnRpb24iDQogICAgSUQ9ImQyYjdjMzg4Y2VjMzZmYTdj
MzljMjhmZDI5ODY0NGE4IiBJc3N1ZUluc3RhbnQ9IjIwMDQtMDEtMjFUMTk6MDA6
NDlaIiBWZXJzaW9uPSIyLjAiPg0KICAgIDxJc3N1ZXI+aHR0cHM6Ly9JZGVudGl0
eVByb3ZpZGVyLmNvbS9TQU1MPC9Jc3N1ZXI+DQogICAgPE5hbWVJRCBGb3JtYXQ9
InVybjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0OnBlcnNp
c3RlbnQiPjAwNWEwNmUwLWFkODItMTEwZC1hNTU2LTAwNDAwNWIxM2EyYjwvTmFt
ZUlEPg0KICAgIDxzYW1scDpTZXNzaW9uSW5kZXg+MTwvc2FtbHA6U2Vzc2lvbklu
ZGV4Pg0KPC9zYW1scDpMb2dvdXRSZXF1ZXN0Pg=="/>
</div>
<noscript>
<div>
<input type="submit" value="Continue"/>
</div>
</noscript>
</form>
</body>
</html>
```

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="PHNhbWxwOkxvZ291dFJlc3BvbnNlIHhtbG5zOnNhbWxwPSJ1cm46b2FzaXM6bmFt
ZXM6dGM6U0FNTDoyLjA6cHJvdG9jb2wiIHhtbG5zPSJ1cm46b2FzaXM6bmFtZXM6
dGM6U0FNTDoyLjA6YXNzZXJ0aW9uIg0KICAgIElEPSJiMDczMGQyMWI2MjgxMTBk
OGI3ZTAwNDAwNWIxM2EyYiIgSW5SZXNwb25zZVRvPSJkMmI3YzM4OGNlYzM2ZmE3
YzM5YzI4ZmQyOTg2NDRhOCINCiAgICBJc3N1ZUluc3RhbnQ9IjIwMDQtMDEtMjFU
MTk6MDA6NDlaIiBWZXJzaW9uPSIyLjAiPg0KICAgIDxJc3N1ZXI+aHR0cHM6Ly9T
ZXJ2aWNlUHJvdmlkZXIuY29tL1NBTUw8L0lzc3Vlcj4NCiAgICA8c2FtbHA6U3Rh
dHVzPg0KICAgICAgICA8c2FtbHA6U3RhdHVzQ29kZSBWYWx1ZT0idXJuOm9hc2lz
Om5hbWVzOnRjOlNBTUw6Mi4wOnN0YXR1czpTdWNjZXNzIi8+DQogICAgPC9zYW1s
cDpTdGF0dXM+DQo8L3NhbWxwOlJlc3BvbnNlPg=="/>
</div>
<noscript>
<div>
<input type="submit" value="Continue"/>
</div>
</noscript>
</form>
</body>
</html>
```

3 References

@@TODO: clean up here

[HTML401] D. Raggett et al. *HTML 4.01 Specification.* World Wide Web Consortium Recommendation, December 1999. See http://www.w3.org/TR/html4.

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

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

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

[RFC2045] N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, IETF RFC 2045, November 1996. See http://www.ietf.org/rfc/rfc2045.txt.

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

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

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

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

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

[SAML11Bind] E. Maler et al. *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-1.1. See http://www.oasis-open.org/committees/security/.

[SAML20Bind] S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os. See http://www.oasis-open.org/committees/security/.

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

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

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

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

[SAMLmime] application/saml+xml Media Type Registration, IETF Internet-Draft, http://www.ietf.org/internet-drafts/draft-hodges-saml-mediatype-01.txt.

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

[SAMLSecure] F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-sec-consider-2.0-os. See http://www.oasis-open.org/committees/security/.

[SOAP11] D. Box et al. *Simple Object Access Protocol (SOAP) 1.1*. World Wide Web Consortium Note, May 2000. See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[SOAP-PRIMER] N. Mitra. *SOAP Version 1.2 Part 0: Primer*. World Wide Web Consortium Recommendation, June 2003. See http://www.w3.org/TR/soap12-part0/,

[SSL3] A. Frier et al. *The SSL 3.0 Protocol*. Netscape Communications Corp, November 1996.

[SSTCWeb] OASIS Security Services Technical Committee website, http://www.oasis-open.org/committees/security.

[XHTML] *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*. World Wide Web Consortium Recommendation, August 2002. See http://www.w3.org/TR/xhtml1/.

[XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web Consortium Recommendation, February 2002. See http://www.w3.org/TR/xmldsig-core/.

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:

- TBD

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

- TBD

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.