SAMLv2.0 HTTP POST “SimpleSign” Binding

Draft 02, 5 January 2007

Document identifier:
draft-sstc-saml-binding-simplesign-02

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

Technical Committee:
OASIS Security Services TC

Chairs:
Hal Lockhart, BEA Systems, Inc.
Prateek Mishra, Oracle Corporation

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 data origination authentication. Rather, a “sign the BLOB” technique is employed wherein a conveyed SAML message is treated as a simple octet string if it is signed. Conveyed SAML assertions may be individually signed using XMLdsig. Security is optional in this binding.

Status:
This is a **Working Draft** and the text may change before completion.

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.

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

This specification defines a SAML HTTP protocol binding, specifically using the HTTP POST method, and which specifically does not use XML Digital Signature[XMLSig] for SAML message data origination authentication. Rather, a “sign the BLOB” technique is employed wherein a conveyed SAML message, along with any content (e.g. SAML assertion(s)), is treated as a simple octet string if it is signed. Additionally, it is out of the scope of this specification whether or not conveyed SAML assertions are authenticated via XML Digital Signature. Security is optional in this binding.

The next subsection gives a general overview of SAML Protocol Binding concepts, followed by notation and namespace declarations. The binding itself is defined in Section 2.

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 the given binding in sufficient detail to ensure that independently implemented SAML-conforming software can interoperate when using standard messaging or communication protocols.

Unless otherwise specified, this binding should be understood to support the transmission of any SAML protocol message derived from the **samlp:RequestAbstractType** and **samlp:StatusResponseType** types. Further, when this 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].

Prefix	XML Namespace	Comments
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-SimpleSign

The HTTP POST binding, defined in [SAMLBind],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-based binding where the conveyed SAML protocol messages – including their content, i.e. any conveyed SAML assertions – are signed as simple “BLOBs” (“Binary Large Objects”, aka binary octet strings).

Note that this binding defines the conveyance of an <u>individual</u> SAML request <u>or</u> response message via HTTP POST.Thus this binding MAY be composed with the HTTP Redirect binding (see Section 3.4 of [SAMLBind]) or the HTTP Artifact binding (see Section 3.6 of [SAMLBind] to transmit request and response messages in an overall SAML protocol exchange, the definition of which is termed a “SAML Profile” [SAMLProf], using two different bindings.

2.1 Required Information

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

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

Description: Given below.

Updates: None. Rather, it provides an <u>alternative</u> to the HTTP POST Binding defined in [SAMLBind]

2.2 Overview

The HTTP POST-SimpleSign binding is intended for cases in which the SAML requester or responder need to communicate using an HTTP user agent (as defined in HTTP 1.1 [RFC2616] as an intermediary, and when data origination authentication and integrity protection of the SAML message is not required, or when a lighter-weight signature mechanism (as compared to [XMLSig] is appropriate. 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 does not require such capabilities—it assumes nothing apart from the capabilities of a common web browser.

2.3 Relay State

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, either via a digital signature (see section 2.5) or by some independent means.

If a SAML request message is accompanied by RelayState data, then the SAML responder MUST return its SAML protocol response message 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 message.

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.

2.4 Message Encoding and Conveyance

This section describes how to encode a SAML protocol message, and thus any SAML assertion(s) it may contain, into HTML FORM "control(s)" [HTML401] (Section 17), thus enabling the SAML protocol message to be conveyed 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].
    a. If the SAML protocol message is a SAML request, then the form "control name" used to convey the SAML protocol message itself MUST be `SAMLRequest`.
    b. If the SAML protocol message is a SAML response, then the form "control name" used to convey the SAML protocol message itself MUST be `SAMLResponse`.
    c. 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 SAML protocol message itself.

SAML protocol messages, and any SAML assertions contained within the SAML protocol messages, MAY be signed using [XMLSig], and if so, any such signatures MUST remain intact. Additionally, SAML protocol messages MAY be signed using the technique given below in section 2.5. This technique is referred to as the "SimpleSign technique". The SimpleSign signature value is conveyed in a form control value named `Signature`, 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 SAML protocol message is signed using SimpleSign, 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 SAML protocol message has been received. Also, the signer's certificate or other keying information MAY be included in a form control named `KeyInfo`. This form control, if present, MUST contain a base-64 encoded `<ds:KeyInfo>` element [XMLSig] (base-64 encoding is done as in step 1, above).

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 protocol 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`.

All of the above form attributes and form controls, to which values are assigned per the above discussion, comprise the form data set. The form data set is then encoded into an HTTP response `message-body` as a `<FORM>` element. The HTTP response message is then sent to the user agent.

Any technique supported by the user agent MAY be used to cause the submission of the form (to cause it to be conveyed to the SAML protocol message recipient), 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. [HTML401][XHTML]

2.5 SimpleSign 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 raw, unencoded XML making up the SAML protocol message (NOT the base64-encoded version), the `RelayState` value (if present), and the `SigAlg` value, 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 [RFC2045]), and used as the value for the `Signature` form control as discussed in section 2.4, above.

Note that this is subtly different from the signature approach defined by the HTTP-Redirect binding [SAMLBind]. Experimentation shows that many web browsers alter linefeeds when submitting form controls that span multiple lines. Since base64-encoded data often wraps, it is not possible to guarantee that the values submitted will match what the original signer produced, resulting in verification failures. Using the raw XML content as a component of the octet string addresses this issue.

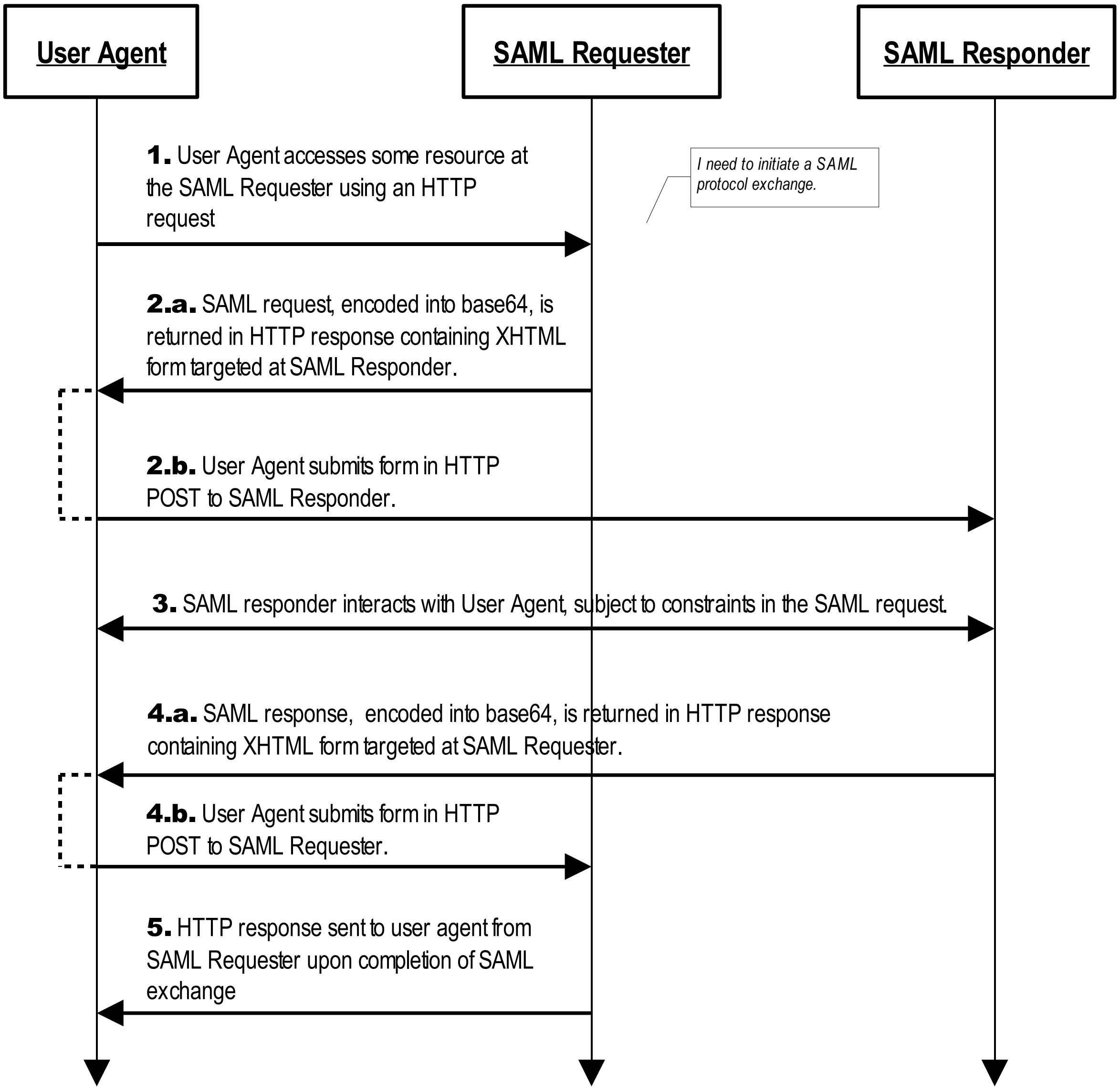
The original XML MUST be concatenated with the other information as shown above without regard for any embedded whitespace, even if the result spans multiple lines. The specific whitespace characters present will be safely encoded in base64 and then recovered by the relying party for use in verifying the signature.

2.6 SimpleSign Signature Verification

To verify a received SAML protocol message, which was signed using SimpleSign and 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 2.5 step 2, above. The signature value conveyed in the `Signature` control value is then checked against this string per the signature algorithm given by the `SigAlg` control value, and using (as appropriate, see [XMLSig]) the keying material obtained via the `<ds:KeyInfo>` conveyed in the `KeyInfo` control value (if present). Error handling and generated messages as a result of the signature not verifying are implementation-dependent.

2.7 Message Exchange

The system model used for SAML conversations via this binding is a request-response model. However, a SAML request message is sent to the user agent via an HTTP response message, and subsequently delivered to the SAML responder via an HTTP request message issued by the user agent. Any HTTP interactions before, between, and after the foregoing 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. Note that although the diagram illustrates both the SAML request and the SAML response being conveyed via the HTTP POST-SimpleSign binding, one or the other of the SAML request or the SAML response could be conveyed via a different SAML HTTP-based binding.



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. (a) 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 2.4, above. (b) 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>` [SAMLCore].
4. Eventually the responder SHOULD (a) return a SAML response to the user agent to be (b) returned to the SAML requester. The SAML response is returned in the same fashion as described for the SAML request in step 2, if this or a similar binding is employed for this step. Otherwise, details may vary.

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

2.7.1 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.

2.7.2 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. This binding defines the SimpleSign approach as a means for signing the conveyed SAML protocol 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 on-the-wire confidentiality is necessary, SSL 3.0 [SSL3]or TLS 1.0 [RFC2246] SHOULD be used to protect the overall HTTP messages, and the conveyed SAML protocol messages, 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:** Cryptographically-based security is entirely OPTIONAL in this binding. If no security mechanisms are employed, then there is essentially no runtime assurance as to the identity of any of the communicating entities.

If the SAML protocol messages are signed (using the SimpleSign approach or [XMLSig]) then 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.

Note also that the SimpleSign technique, if employed, binds the RelayState value (if present) to the SAML protocol message, unlike the [XMLSig]-based technique of the HTTP POST binding [SAMLBind]. Thus, if a SAML protocol message is not signed using SimpleSign, but is signed using the [XMLSig]-based technique, then the caveats with respect to any conveyed RelayState value, presented in section 3.5.5.2 of [SAMLBind], should be taken into account.

2.8 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]

2.9 Metadata Considerations

Support for the HTTP POST-SimpleSign 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 [SAMLMeta]. The identification URI given in section 2.1 is used as the value for the `Binding` attribute of any endpoint elements.

2.10 Note to Implementors

SAML protocol message recipients can distinguish between HTTP-SAML messages constructed via this specification's HTTP POST-SimpleSign binding and ones constructed via the HTTP POST binding [SAMLBind] by examining received HTTP messages for an XHTML form field with a name attribute value of `Signature`. If this is present, then the message MUST be processed in accordance with this specification. If not present, then the HTTP message MAY be processed in accordance with the HTTP POST binding specification.

2.11 Example

In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair is exchanged using the HTTP POST–SimpleSign binding. The messages are signed as described in section 2.5, above. If the messages were unsigned, they would be the same as shown below, except that the hidden form controls named `Signature` and `SigAlg` would be missing.

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="http://ServiceProvider.com/SAML/SLO/Browser" method="post">
<div>
<input type="hidden" name="RelayState"
value="0043bfc1bc45110dae17004005b13a2b"/>
<input type="hidden" name="SAMLRequest"
value="PHNhbWxwOkxvZ291dFJlcXVlc3QgeG1sbnM6c2FtbHA9InVybjpvYXNpczpuYW1l
czp0YzpTQU1MOjIuMDpwcm90b2NvbCIgeG1sbnM9InVybjpvYXNpczpuYW1lczp0
YzpTQU1MOjIuMDphc3NlcnRpb24iCiAgICBJRD0iZDJiN2MzODhjZWMzNmZhN2Mz
OWMyOGZkMjk4NjQ0YTgiIElzc3VlSW5zdGFudD0iMjAwNC0wMS0yMVQxOTowMDo0
OVoiIFZlcnNpb249IjIuMCI+CiAgICA8SXNzdWVyPmh0dHBzOi8vSWRlbnRpdHlQ
cm92aWRlci5jb20vU0FNTDwvSXNzdWVyPgogICAgPE5hbWVJRCBGb3JtYXQ9InVy
bjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0OnBlcnNpc3Rl
bnQiPjAwNWEwNmUwLWFkODItMTEwZC1hNTU2LTAwNDAwNWIxM2EyYjwvTmFtZUlE
PgogICAgPHNhbWxwOlNlc3Npb25JbmRleD4xPC9zYW1scDpTZXNzaW9uSW5kZXg+
Cjwvc2FtbHA6TG9nb3V0UmVxdWVzdD4K"/>
<input type="hidden" name="Signature"
value="J4if7CCeHVfn4H6hMZN5fijOjQIyZ/laoFUZWz4LCRN3J82UeoyYvAiTDoQOUZHT
RJNU1lWGub1pW4QR9MH5bwfLEa8XDivA118dR0Q7YN5L/U5rmbxnGlQ9pV0jT44c
RNeqtbLW0YF4plfcqg7E5iOSljE3QLkiaAdkAec2a4HwPFkn/JP7wO11Mc6kU8ML
CBbZAa3+94ZvVwHBEdyCdU+1yEvf+JGxTw66BwI2ugmPfxvoJdsOOAWwS3KhAFhL
LSPXnhb3nd/ovKNNV/khZYwqsFTFNTMA+0JraKsZiCRtEZzEPXaP9KilrjPIIvRV
xDQhETj96flk5zMkEM3ruw=="/>
<input type="hidden" name="SigAlg"
value="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
</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="PHNhbWxwOkxvZ291dFJlcXVlc3QgeG1sbnM6c2FtbHA9InVybjpvYXNpczpuYW1l
czp0YzpTQU1MOjIuMDpwcm90b2NvbCIgeG1sbnM9InVybjpvYXNpczpuYW1lczp0
YzpTQU1MOjIuMDphc3NlcnRpb24iCiAgICBJRD0iZDJiN2MzODhjZWMzNmZhN2Mz
OWMyOGZkMjk4NjQ0YTgiIElzc3VlSW5zdGFudD0iMjAwNC0wMS0yMVQxOTowMDo0
OVoiIFZlcnNpb249IjIuMCI+CiAgICA8SXNzdWVyPmh0dHBzOi8vSWRlbnRpdHlQ
cm92aWRlci5jb20vU0FNTDwvSXNzdWVyPgogICAgPE5hbWVJRCBGb3JtYXQ9InVy
bjpvYXNpczpuYW1lczp0YzpTQU1MOjIuMDpuYW1laWQtZm9ybWF0OnBlcnNpc3Rl
bnQiPjAwNWEwNmUwLWFkODItMTEwZC1hNTU2LTAwNDAwNWIxM2EyYjwvTmFtZUlE
PgogICAgPHNhbWxwOlNlc3Npb25JbmRleD4xPC9zYW1scDpTZXNzaW9uSW5kZXg+
Cjwvc2FtbHA6TG9nb3V0UmVxdWVzdD4K"/>
<input type="hidden" name="Signature"
value="DCDqAwIDqSwyXGvG2cYvNjmj7P1kt0+kbCfRjq9gGTrN4KKPxvQl5EsFrWRkMOdx
xuwPldWPKvfgX6rt+pKwLgCt1TqRj+71y+VdGS8ORsBeEIURRn9wSu+pKsWiHexw
KnIe65bjONbg2db44QOWZlDe76fLi05Psy/7HZTQuMoDRFYSR//VyNGHQmf9Sxi6
mkmrYMXPOyZAUfNhX4eLaXFfwCHt0yRrEcm/PAEDDa7uqe8Uo5ilstgXDWDodWdk
Szk8ZS1irjFkvtxH7FJlm9ADtlW/SoX92jGjMIrdQwCyArI6o8KTiDp/cjDjHZGi
XLx2WvS7GEibA7Qd+5hSBQ=="/>
<input type="hidden" name="SigAlg"
value="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
</div>
<noscript>
<div>
<input type="submit" value="Continue"/>
</div>
</noscript>
</form>
</body>
</html>
```

3 References

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

[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.

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

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

[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/.

[SAMLProf] 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/.

[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 A. Acknowledgments

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

- Christopher Laskowski, Booz Allen Hamilton
- Rebekah Metz, Booz Allen Hamilton
- Hal Lockhart, BEA Systems, Inc.
- Steve Anderson, BMC Software
- Sharon Boeyen, Entrust
- Thomas Wisniewski, Entrust
- Carolina Canales-Valenzuela, Ericsson
- Dana Kaufman, Forum Systems, Inc.
- Ashish Patel, France Telecom
- Greg Whitehead, Hewlett-Packard
- Guy Denton, IBM
- Heather Hinton, IBM
- Anthony Nadalin, IBM
- Eric Tiffany, IEEE Industry Standards and Technology Org (IEEE-ISTO)
- Scott Cantor, Internet2
- Bob Morgan, Internet2
- Tom Scavo, National Center for Supercomputing Applications (NCSA)
- Peter Davis, Neustar, Inc.
- Jeff Hodges, Neustar, Inc.
- Frederick Hirsch, Nokia Corporation
- Abbie Barbir, Nortel Networks Limited
- Paul Madsen, NTT Corporation
- Ari Kermaier, Oracle Corporation
- Prateek Mishra, Oracle Corporation
- Brian Campbell, Ping Identity Corporation
- Rob Philpott, RSA Security
- Jahan Moreh, Sigaba Corp.
- Bhavna Bhatnagar, Sun Microsystems
- Eve Maler, Sun Microsystems
- Emily Xu, Sun Microsystems
- David Staggs, Veterans Health Administration

Appendix B. Notices