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

OASIS Draft, 2 January 2004

Document identifier:
sstc-saml-bindings-2.0-draft-02

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

Editors:
Frederick Hirsch, Nokia Mobile Phones

Contributors:
TBD

Abstract:

This specification defines protocol bindings and profiles for the use of SAML assertions and request-response messages in communications protocols and frameworks.

Status:

This is a Draft.

Committee members should submit comments and potential errata to the security-services@lists.oasis-open.org list. Others should submit them to the security-services-comment@lists.oasis-open.org list (to post, you must subscribe; to subscribe, send a message to security-services-comment-request@lists.oasis-open.org with "subscribe" in the body) or use other OASIS-supported means of submitting comments. The committee will publish vetted errata on the Security Services TC web page (http://www.oasis-open.org/committees/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 document specifies protocol bindings and profiles for the use of SAML assertions and request-response messages in communications protocols and frameworks.

A separate specification [SAMLCore] defines the SAML assertions and request-response messages themselves.

1.1 Protocol Binding and Profile Concepts

Mappings from SAML request-response message exchanges into 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 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.

Sets of rules describing how to embed SAML assertions into and extract them from a framework or protocol are called *profiles of SAML.* A profile describes how SAML assertions are embedded in or combined with other objects (for example, files of various types, or protocol data units of communication protocols) by an originating party, communicated from the originating site to a destination site, and subsequently processed at the destination. A particular set of rules for embedding SAML assertions into and extracting them from a specific class of <FOO> objects is termed a *<FOO> profile of SAML*.

For example, a SOAP profile of SAML describes how SAML assertions can be added to SOAP messages, how SOAP headers are affected by SAML assertions, and how SAML-related error states should be reflected in SOAP messages.

The intent of this specification is to specify a selected set of bindings and profiles in sufficient detail to ensure that independently implemented products will interoperate.

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

1.2 Notation

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: Non-normative notes and explanations appear like this.

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:

- The prefix saml: stands for the SAML assertion namespace [SAMLCore].
- The prefix samlp: stands for the SAML request-response protocol namespace [SAMLCore].
- The prefix ds: stands for the W3C XML Signature namespace, http://www.w3.org/2000/09/xmldsig# [XMLSig].
- The prefix SOAP-ENV: stands for the SOAP 1.1 namespace, http://schemas.xmlsoap.org/soap/envelope [SOAP1.1].

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

2 Specification of Additional Protocol Bindings and Profiles

This specification defines a selected set of protocol bindings and profiles, but others will possibly be developed in the future. It is not possible for the OASIS Security Services Technical Committee to standardize all of these additional bindings and profiles for two reasons: it has limited resources and it does not own the standardization process for all of the technologies used. The following sections offer guidelines for specifying bindings and profiles and a process framework for describing and registering them.

2.1 Guidelines for Specifying Protocol Bindings and Profiles

This section provides a checklist of issues that MUST be addressed by each protocol binding and profile.

1. Describe the set of interactions between parties involved in the binding or profile. Any restrictions on applications used by each party and the protocols involved in each interaction must be explicitly called out.
2. Identify the parties involved in each interaction, including how many parties are involved and whether intermediaries may be involved.
3. Specify the method of authentication of parties involved in each interaction, including whether authentication is required and acceptable authentication types.
4. Identify the level of support for message integrity, including the mechanisms used to ensure message integrity.
5. Identify the level of support for confidentiality, including whether a third party may view the contents of SAML messages and assertions, whether the binding or profile requires confidentiality, and the mechanisms recommended for achieving confidentiality.
6. Identify the error states, including the error states at each participant, especially those that receive and process SAML assertions or messages.
7. Identify security considerations, including analysis of threats and description of countermeasures.
8. Identify SAML confirmation method identifiers defined and/or utilized by the binding or profile.

2.2 Process Framework for Describing and Registering Protocol Bindings and Profiles

For any new protocol binding or profile to be interoperable, it needs to be openly specified. The OASIS Security Services Technical Committee will maintain a registry and repository of submitted bindings and profiles titled "Additional Bindings and Profiles" at the SAML website [SAMLWeb] in order to keep the SAML community informed. The committee will also provide instructions for submission of bindings and profiles by OASIS members.

When a profile or protocol binding is registered, the following information MUST be supplied:

1. Identification: Specify a URI that uniquely identifies this protocol binding or profile.
2. Contact information: Specify the postal or electronic contact information for the author of the protocol binding or profile.
3. Description: Provide a text description of the protocol binding or profile. The description SHOULD follow the guidelines described in Section 2.1.

4. Updates: Provide references to previously registered protocol bindings or profiles that the current entry improves or obsoletes.

3 Protocol Bindings

The following sections define SAML protocol bindings sanctioned by the OASIS Security Services Technical Committee. Only one binding, the SAML SOAP binding, is currently defined.

3.1 SAML SOAP Binding

SOAP (Simple Object Access Protocol) 1.1 [SOAP1.1] is a specification for RPC-like interactions and message communications using XML and HTTP. It has three main parts. One is a message format that uses an envelope and body metaphor to wrap XML data for transmission between parties. The second is a restricted definition of XML data for making strict RPC-like calls through SOAP, without using a predefined XML schema. Finally, it provides a binding for SOAP messages to HTTP and extended HTTP.

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).

3.1.1 Required Information

Identification: urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding

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

Description: Given below.

Updates: None.

3.1.2 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.

3.1.2.1 Basic Operation

SOAP 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 `<Response>` element within the body of another SOAP message or a SOAP fault code. 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 return a SOAP fault code. 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 [SOAP1.1] §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.

[SOAP1.1] 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 [SOAP1.1] as well as the final XML schema namespace.

3.1.2.2 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.

A SAML responder MUST NOT require any headers for the SOAP message.

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

3.1.2.3 Authentication

Authentication of both the SAML requester and the SAML responder is OPTIONAL and depends on the environment of use. Authentication protocols available from the underlying substrate protocol MAY be utilized to provide authentication. Section 3.1.3.2 describes authentication in the SOAP over HTTP environment.

3.1.2.4 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 MAY be used to ensure message integrity. Section 3.1.3.3 describes support for message integrity in the SOAP over HTTP environment.

3.1.2.5 Confidentiality

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 MAY be used to ensure message confidentiality. Section 3.1.3.4 describes support for confidentiality in the SOAP over HTTP environment.

3.1.3 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, error reporting, authentication, message integrity, and confidentiality.

The HTTP binding for SOAP is described in [SOAP1.1] §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 `SOAPAction` header as follows:

```
http://www.oasis-open.org/committees/security
```

3.1.3.1 HTTP Headers

HTTP proxies MUST NOT cache responses carrying SAML assertions.

Both of the following conditions apply when using HTTP 1.1:

1. If the value of the `Cache-Control` header field is **not** set to `no-store`, then the SAML responder MUST NOT include the `Cache-Control` header field in the response.
2. If the `Expires` response header field is **not** disabled by a `Cache-Control` header field with a value of `no-store`, then the `Expires` field SHOULD NOT be included.

There are no other restrictions on HTTP headers.

3.1.3.2 Authentication

The SAML requester and responder MUST implement the following authentication methods:

1. No client or server authentication.
2. HTTP basic client authentication [RFC2617] with and without SSL 3.0 or TLS 1.0.
3. HTTP over SSL 3.0 or TLS 1.0 (see Section 6) server authentication with a server-side certificate.
4. HTTP over SSL 3.0 or TLS 1.0 mutual authentication with both server-side and a client-side certificate.

If a SAML responder uses SSL 3.0 or TLS 1.0, it MUST use a server-side certificate.

3.1.3.3 Message Integrity

When message integrity needs to be guaranteed, SAML responders MUST use HTTP over SSL 3.0 or TLS 1.0 (see Section 6) with a server-side certificate.

3.1.3.4 Message Confidentiality

When message confidentiality is required, SAML responders MUST use HTTP over SSL 3.0 or TLS 1.0 (see Section 6) with a server-side certificate.

3.1.3.5 Security Considerations

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

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

3.1.3.6 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 [SOAP1.1] § 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 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 `<Status>` element as the only child of a SAML `<Response>` element within the SOAP body.

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

3.1.3.7 Example SAML Message Exchange Using SOAP over HTTP

Following is an example of a request that asks for an assertion containing an authentication statement from a SAML authentication 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:Request xmlns:samlp:="…" xmlns:saml="…" xmlns:ds="…">
            <ds:Signature> … </ds:Signature>
            <samlp:AuthenticationQuery>
            …
            </samlp:AuthenticationQuery>
        </samlp:Request>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <samlp:Response xmlns:samlp="…" xmlns:saml="…" xmlns:ds="…">
            <Status>
              <StatusCodevalue="samlp:Success"/>
            </Status>
        <ds:Signature> … </ds:Signature>
        <saml:Assertion>
            <saml:AuthenticationStatement>
            …
            </saml:AuthenticationStatement>
        </saml:Assertion>
        </samlp:Response>
    </SOAP-Env:Body>
</SOAP-ENV:Envelope>
```

4 Profiles

The following sections define profiles of SAML that are sanctioned by the OASIS Security Services Technical Committee.

Two web browser-based profiles are defined to support single sign-on (SSO), supporting Scenario 1-1 of the SAML requirements document [SAMLReqs]:

- The browser/artifact profile of SAML
- The browser/POST profile of SAML

For each type of profile, a section describing the threat model and relevant countermeasures is also included.

Some additional profiles that have been published outside the Security Services Technical Committee are:

- The OASIS Web Services Security Technical Committee has produced a draft “SAML token profile” of the WSS specification [WSS-SAML], which describes how to use SAML assertions to secure a web service message.
- The Liberty Alliance Project [Liberty] has produced a set of profiles for its extended version of SAML.

4.1 Web Browser SSO Profiles of SAML

In the scenario supported by the web browser SSO profiles, a web user authenticates to a *source site*. The web user then uses a secured resource at a destination site, without directly authenticating to the *destination site*.

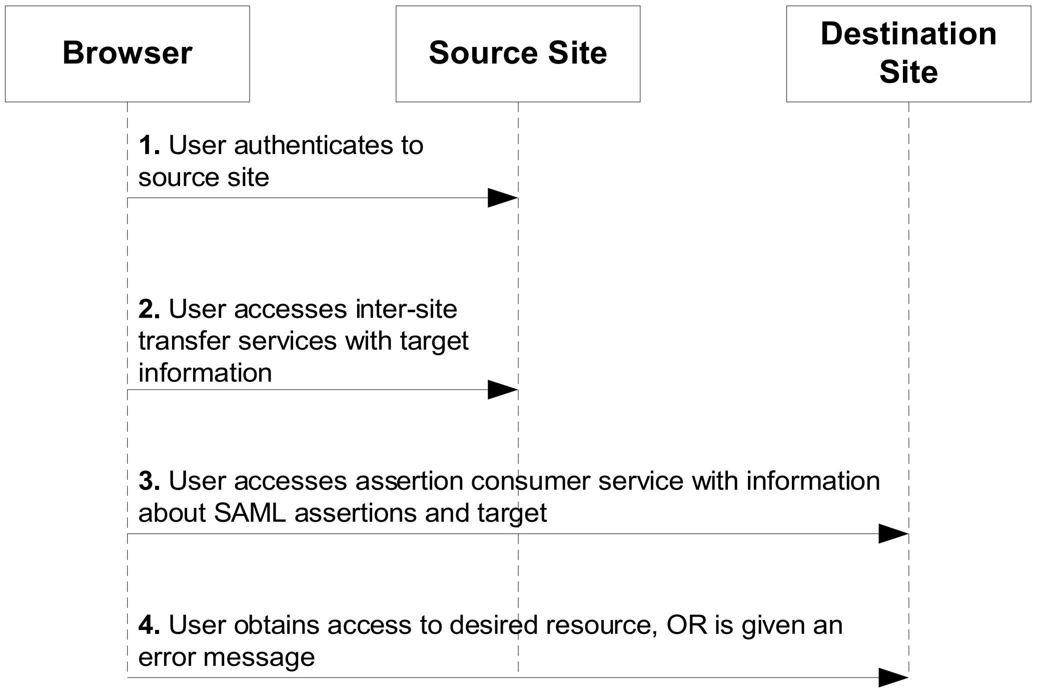
The following assumptions are made about this scenario for the purposes of these profiles:

- user is using a standard commercial browser and has authenticated to a source site by some means outside the scope of SAML.
- The source site has some form of security engine in place that can track locally authenticated users [WEBSSO]. Typically, this takes the form of a session that might be represented by an encrypted cookie or an encoded URL or by the use of some other technology [SESSION]. This is a substantial requirement but one that is met by a large class of security engines.

At some point, the user attempts to access a *target* resource available from the destination site, and subsequently, through one or more steps (for example, redirection), arrives at an *inter-site transfer service* (which may be associated with one or more URIs) at the source site. Starting from this point, the web browser SSO profiles describe a canonical sequence of HTTP exchanges that transfer the user browser to an *assertion consumer service* at the destination site. Information about the SAML assertions provided by the source site and associated with the user, and the desired target, is conveyed from the source to the destination site by the protocol exchange.

The assertion consumer service at the destination site can examine both the assertions and the target information and determine whether to allow access to the target resource, thereby achieving web SSO for authenticated users originating from a source site. Often, the destination site also utilizes a security engine that will create and maintain a session, possibly utilizing information contained in the source site assertions, for the user at the destination site.

The following figure illustrates this basic template for achieving SSO.


Browser
Source Site
Destination Site
1. User authenticates to source site
2. User accesses inter-site transfer services with target information
3. User accesses assertion consumer service with information about SAML assertions and target
4. User obtains access to desired resource, OR is given an error message

Two HTTP-based techniques are used in the web browser SSO profiles for conveying information from one site to another via a standard commercial browser.

- **SAML artifact:** A SAML artifact of "small" bounded size is carried to the destination site as part of a URL query string such that, when the artifact is later conveyed back to the source site, the artifact unambiguously references an assertion. The artifact is conveyed via redirection to the destination site, which then acquires the referenced assertion from the source site by some further steps. Typically, this involves the use of a registered SAML protocol binding. This technique is used in the browser/artifact profile of SAML.
- **Form POST:** SAML assertions are uploaded to the browser within an HTML form and conveyed to the destination site as part of an HTTP POST payload when the user submits the form. This technique is used in the browser/POST profile of SAML.

Cookies are not employed in these profiles, as cookies impose the limitation that both the source and destination site belong to the same "cookie domain."

In the discussion of the web browser SSO profiles, the term *SSO assertion* will be used to refer to an assertion that has a `<saml:Conditions>` element with `NotBefore` and `NotOnOrAfter` attributes present, and also contains at least one or more authentication statements about the subject. Note that an SSO assertion MAY also include additional information about the subject, such as attributes.

4.1.1 Browser/Artifact Profile of SAML

4.1.1.1 Required Information

Identification: urn:oasis:names:tc:SAML:1.0:profiles:artifact-01

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

SAML Confirmation Method Identifiers: The "SAML artifact" confirmation method identifier is used by this profile. The following RECOMMENDED identifier has been assigned to this confirmation method:

urn:oasis:names:tc:SAML:1.0:cm:artifact

Description: Given below.

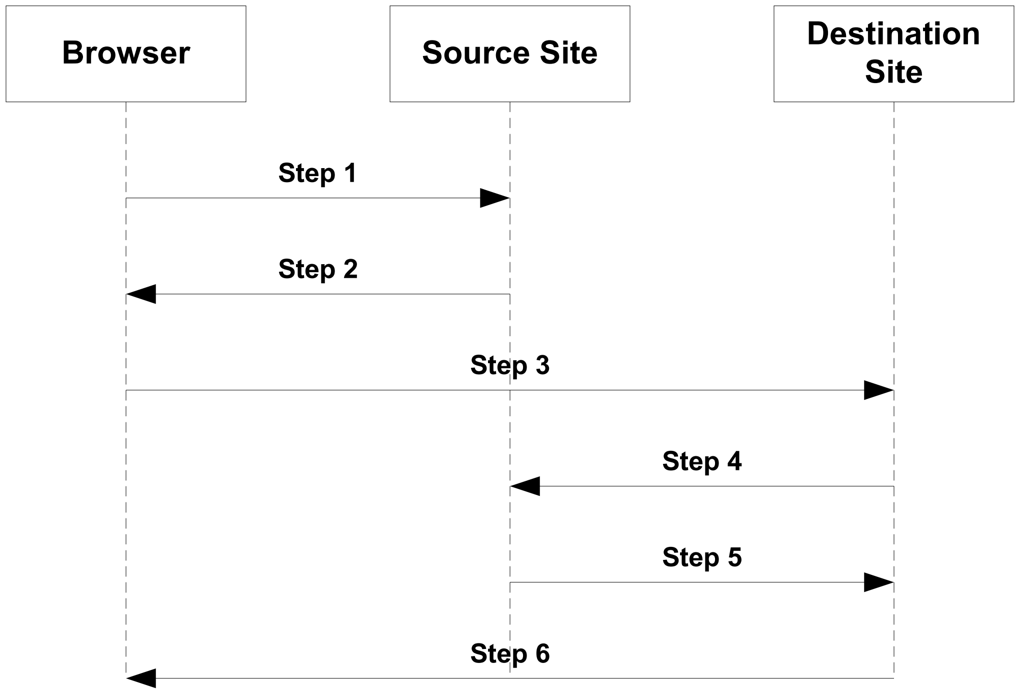
Updates: None.

4.1.1.2 Preliminaries

The browser/artifact profile of SAML relies on a reference to the needed assertion traveling in a SAML artifact, which the destination site must dereference from the source site in order to determine whether the user is authenticated.

> **Note:** The need for a "small'' SAML artifact is motivated by restrictions on URL size imposed by commercial web browsers. While RFC 2616 [RFC2279] UTF-8, a transformation format of ISO 10646, http://www.ietf.org/rfc/rfc2279.txt. does not specify any restrictions on URL length, in practice commercial web browsers and application servers impose size constraints on URLs, for a maximum size of approximately 2000 characters (see Section 8). Further, as developers will need to estimate and set aside URL "real estate" for the artifact, it is important that the artifact have a bounded size, that is, with predefined maximum size. These measures ensure that the artifact can be reliably carried as part of the URL query string and thereby transferred successfully from source to destination site.

The browser/artifact profile consists of a single interaction among three parties (a user equipped with a browser, a source site, and a destination site), with a nested sub-interaction between two parties (the source site and the destination site). The interaction sequence is shown in the following figure, with the following sections elucidating each step.



Terminology from RFC 1738 [RFC1738] is used to describe components of a URL. An HTTP URL has the following form:

```
http://<HOST>:<port>/<path>?<searchpart>
```

The following sections specify certain portions of the `<searchpart>` component of the URL. Ellipses will be used to indicate additional but unspecified portions of the `<searchpart>` component.

HTTP requests and responses MUST be drawn from either HTTP 1.1 [RFC2279] UTF-8, a transformation format of ISO 10646, http://www.ietf.org/rfc/rfc2279.txt. or HTTP 1.0 [RFC1945]. Distinctions between the two are drawn only when necessary.

4.1.1.3 Step 1: Accessing the Inter-Site Transfer Service

In step 1, the user's browser accesses the inter-site transfer service at host https://<inter-site transfer host name>, with information about the desired target at the destination site attached to the URL.

No normative form is given for step 1. It is RECOMMENDED that the HTTP request take the following form:

```
GET <path>?…TARGET=<Target>…<HTTP-Version>
<other HTTP 1.0 or 1.1 components>
```

Where:

`<inter-site transfer host name>`

This provides the host name and optional port number at the source site where an inter-site transfer service is available.

`<path>`

This provides the path components of an inter-site transfer service URL at the source site.

`Target=<Target>`

This name-value pair occurs in the `<searchpart>` and is used to convey information about the desired target resource at the destination site.

Confidentiality and message integrity MUST be maintained in step 1.

4.1.1.4 Step 2: Redirecting to the Destination Site

In step 2, the source site's inter-site transfer service responds and redirects the user's browser to the assertion consumer service at the destination site.

> **Note:** In the browser/artifact profile, the URL used by the source site to access the assertion consumer service at the destination site is referred to as the *artifact receiver URL*.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<artifact receiver host name and path>?<SAML
searchpart>
<other HTTP 1.0 or 1.1 components>
```

Where:

`<artifact receiver host name and path>`

This provides the host name, port number, and path components of an artifact receiver URL associated with the assertion consumer service at the destination site.

`<SAML searchpart>= …TARGET=<Target>…SAMLart=<SAML artifact> …`

A single target description MUST be included in the `<SAML searchpart>` component. At least one SAML artifact MUST be included in the SAML `<SAML searchpart>` component; multiple SAML artifacts MAY be included. If more than one artifact is carried within `<SAML searchpart>`, all the artifacts MUST have the same `SourceID`.

According to HTTP 1.1 [RFC2279] UTF-8, a transformation format of ISO 10646, http://www.ietf.org/rfc/rfc2279.txt. and HTTP 1.0 [RFC1945], the use of status code 302 is recommended to indicate that "the requested resource resides temporarily under a different URI". The response may also include additional headers and an optional message body as described in those RFCs.

Confidentiality and message integrity MUST be maintained in step 2. It is RECOMMENDED that the inter-site transfer URL be protected by SSL 3.0 or TLS 1.0 (see Section 6). Otherwise, the one or more artifacts returned in step 2 will be available in plain text to an attacker who might then be able to impersonate the subject.

4.1.1.5 Step 3: Accessing the Artifact Receiver URL

In step 3, the user's browser accesses the artifact receiver service at host https://<artifact receiver host name>, with a SAML artifact representing the user's authentication information attached to the URL.

The HTTP request MUST take the form:

```
GET <path>?…<SAML searchpart>…<HTTP-Version>
<other HTTP 1.0 or 1.1 request components>
```

Where:

`<artifact receiver host name>`

This provides the host name and optional port number at the destination site where the artifact receiver service URL associated with the assertion consumer service is available.

`<path>`

This provides the path components of the artifact receiver service URL at the destination site.

`<SAML searchpart>= …TARGET=<Target>…SAMLart=<SAML artifact> …`

A single target description MUST be included in the `<SAML searchpart>` component. At least one SAML artifact MUST be included in the `<SAML searchpart>` component; multiple SAML artifacts MAY be included. If more than one artifact is carried within `<SAML searchpart>`, all the artifacts MUST have the same `SourceID`.

Confidentiality and message integrity MUST be maintained in step 3. It is RECOMMENDED that the artifact receiver URL be protected by SSL 3.0 or TLS 1.0 (see Section 6). Otherwise, the artifacts transmitted in step 3 will be available in plain text to any attacker who might then be able to impersonate the assertion subject.

4.1.1.6 Steps 4 and 5: Acquiring the Corresponding Assertions

In steps 4 and 5, the destination site, in effect, dereferences the one or more SAML artifacts in its possession in order to acquire a SAML assertion that corresponds to each artifact.

These steps MUST utilize a SAML protocol binding for a SAML request-response message exchange between the destination and source sites. The destination site functions as a SAML requester and the source site functions as a SAML responder.

The destination site MUST send a `<samlp:Request>` message to the source site, requesting assertions by supplying assertion artifacts in the `<samlp:AssertionArtifact>` element.

If the source site is able to find or construct the requested assertions, it responds with a `<samlp:Response>` message with the requested assertions. Otherwise, it responds with a `<samlp:Response>` message with no assertions. The `<samlp:Status>` element of the `<samlp:Response>` MUST include a `<samlp:StatusCode>` element with the value `Success`.

In the case where the source site returns assertions within `<samlp:Response>`, it MUST return exactly one assertion for each SAML artifact found in the corresponding `<samlp:Request>` element. The case where fewer or greater number of assertions is returned within the `<samlp:Response>` element MUST be treated as an error state by the destination site.

The source site MUST implement a “one-time request” property for each SAML artifact. Many simple implementations meet this constraint by an action such as deleting the relevant assertion from persistent storage at the source site after one lookup. If a SAML artifact is presented to the source site again, the source site MUST return the same message as it would if it were queried with an unknown artifact.

The selected SAML protocol binding MUST provide confidentiality, message integrity, and bilateral authentication. The source site MUST implement the SAML SOAP binding with support for confidentiality, message integrity, and bilateral authentication.

The source site MUST return a response with no assertions if it receives a `<samlp:Request>` message from an authenticated destination site *X* containing an artifact issued by the source site to some other destination site *Y*, where *X* <>*Y*. One way to implement this feature is to have source sites maintain a list of artifact and destination site pairs. The `<samlp:Status>` element of the `<samlp:Response>` MUST include a `<samlp:StatusCode>` element with the value `Success`.

At least one of the SAML assertions returned to the destination site MUST be an *SSO assertion*.

Authentication statements MAY be distributed across more than one returned assertion.

Every subject-based statement in the assertion(s) returned to the destination site MUST contain a `<saml:SubjectConfirmation>` element as follows:

- The `<saml:ConfirmationMethod>` element MUST be set to urn:oasis:names:tc:SAML:1.0:cm:artifact.
- The `<SubjectConfirmationData>` element SHOULD NOT be specified.

Based on the information obtained in the assertions retrieved by the destination site, the destination site MAY engage in additional SAML message exchanges with the source site.

4.1.1.7 Step 6: Responding to the User’s Request for a Resource

In step 6, the user’s browser is sent an HTTP response that either allows or denies access to the desired resource.

No normative form is mandated for the HTTP response. The destination site SHOULD provide some form of helpful error message in the case where access to resources at that site is disallowed.

4.1.1.8 Artifact Format

The artifact format includes a mandatory two-byte artifact type code, as follows:

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

Note: Depending on the level of security desired and associated profile protocol steps, many viable architectures could be developed for the SAML artifact [CoreAssnEx] [ShibMarlena]. The type code structure accommodates variability in the architecture.

The notation `B64(TypeCode RemainingArtifact)` stands for the application of the base64 [RFC2045] transformation to the catenation of the `TypeCode` and `RemainingArtifact`. This profile defines an artifact type of type code 0x0001, which is REQUIRED (mandatory to implement) for any implementation of the browser/artifact profile. This artifact type is defined as follows:

```
TypeCode          := 0x0001
RemainingArtifact := SourceID AssertionHandle
SourceID          := 20-byte_sequence
AssertionHandle   := 20-byte_sequence
```

`SourceID` is a 20-byte sequence used by the destination site to determine source site identity and location. It is assumed that the destination site will maintain a table of `SourceID` values as well as the URL (or address) for the corresponding SAML responder. This information is communicated between the source and destination sites out-of-band. On receiving the SAML artifact, the destination site determines if the `SourceID` belongs to a known source site and obtains the site location before sending a SAML request (as described in Section 4.1.1.6).

Any two source sites with a common destination site MUST use distinct `SourceID` values. Construction of `AssertionHandle` values is governed by the principle that they SHOULD have no predictable relationship to the contents of the referenced assertion at the source site and it MUST be infeasible to construct or guess the value of a valid, outstanding assertion handle.

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

- Each source site selects a single identification URL. The domain name used within this URL is registered with an appropriate authority and administered by the source site.
- The source site constructs the `SourceID` component of the artifact by taking the SHA-1 hash of the identification URL.
- The `AssertionHandle` value is constructed from a cryptographically strong random or pseudorandom number sequence [RFC1750] generated by the source site. The sequence consists of values of at least eight bytes in size. These values should be padded to a total length of 20 bytes.

4.1.1.9 Threat Model and Countermeasures

This section utilizes materials from [ShibMarlena] and [Rescorla-Sec].

4.1.1.9.1 Stolen Artifact

Threat: If an eavesdropper can copy the real user's SAML artifact, then the eavesdropper could construct a URL with the real user's SAML artifact and be able to impersonate the user at the destination site.

Countermeasure: As indicated in steps 2, 3, 4, and 5, confidentiality MUST be provided whenever an artifact is communicated between a site and the user's browser. This provides protection against an eavesdropper gaining access to a real user's SAML artifact.

If an eavesdropper defeats the measures used to ensure confidentiality, additional countermeasures are available:

- The source and destination sites SHOULD make some reasonable effort to ensure that clock settings at both sites differ by at most a few minutes. Many forms of time synchronization service are available, both over the Internet and from proprietary sources.
- SAML assertions communicated in step 5 MUST include an SSO assertion.
- The source site SHOULD track the time difference between when a SAML artifact is generated and placed on a URL line and when a `<samlp:Request>` message carrying the artifact is received from the destination. A maximum time limit of a few minutes is recommended. Should an assertion be requested by a destination site query beyond this time limit, the source site MUST not provide the assertions to the destination site.
- It is possible for the source site to create SSO assertions either when the corresponding SAML artifact is created or when a `<samlp:Request>` message carrying the artifact is received from the destination. The validity period of the assertion SHOULD be set appropriately in each case: longer for the former, shorter for the latter.
- Values for `NotBefore` and `NotOnOrAfter` attributes of SSO assertions SHOULD have the shortest possible validity period consistent with successful communication of the assertion from source to destination site. This is typically on the order of a few minutes. This ensures that a stolen artifact can only be used successfully within a small time window.
- The destination site MUST check the validity period of all assertions obtained from the source site and reject expired assertions. A destination site MAY choose to implement a stricter test of validity

for SSO assertions, such as requiring the assertion's `IssueInstant` or `AuthenticationInstant` attribute value to be within a few minutes of the time at which the assertion is received at the destination site.

- If a received authentication statement includes a `<saml:SubjectLocality>` element with the IP address of the user, the destination site MAY check the browser IP address against the IP address contained in the authentication statement.

4.1.1.9.2 Attacks on the SAML Protocol Message Exchange

Threat: The message exchange in steps 4 and 5 could be attacked in a variety of ways, including artifact or assertion theft, replay, message insertion or modification, and MITM (man-in-the-middle attack).

Countermeasure: The requirement for the use of a SAML protocol binding with the properties of bilateral authentication, message integrity, and confidentiality defends against these attacks.

4.1.1.9.3 Malicious Destination Site

Threat: Since the destination site obtains artifacts from the user, a malicious site could impersonate the user at some new destination site. The new destination site would obtain assertions from the source site and believe the malicious site to be the user.

Countermeasure: The new destination site will need to authenticate itself to the source site so as to obtain the SAML assertions corresponding to the SAML artifacts. There are two cases to consider:

1. If the new destination site has no relationship with the source site, it will be unable to authenticate and this step will fail.
2. If the new destination site has an existing relationship with the source site, the source site will determine that assertions are being requested by a site other than that to which the artifacts were originally sent. In such a case, the source site MUST not provide the assertions to the new destination site.

4.1.1.9.4 Forged SAML Artifact

Threat: A malicious user could forge a SAML artifact.

Countermeasure: Section 4.1.1.8 provides specific recommendations regarding the construction of a SAML artifact such that it is infeasible to guess or construct the value of a current, valid, and outstanding assertion handle. A malicious user could attempt to repeatedly "guess" a valid SAML artifact value (one that corresponds to an existing assertion at a source site), but given the size of the value space, this action would likely require a very large number of failed attempts. A source site SHOULD implement measures to ensure that repeated attempts at querying against non-existent artifacts result in an alarm.

4.1.1.9.5 Browser State Exposure

Threat: The SAML browser/artifact profile involves "downloading" of SAML artifacts to the web browser from a source site. This information is available as part of the web browser state and is usually stored in persistent storage on the user system in a completely unsecured fashion. The threat here is that the artifact may be "reused" at some later point in time.

Countermeasure: The "one-use" property of SAML artifacts ensures that they cannot be reused from a browser. Due to the recommended short lifetimes of artifacts and mandatory SSO assertions, it is difficult to steal an artifact and reuse it from some other browser at a later time.

4.1.2 Browser/POST Profile of SAML

4.1.2.1 Required Information

Identification: urn:oasis:names:tc:SAML:1.0:profiles:browser-post

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

SAML Confirmation Method Identifiers: The "Bearer" confirmation method identifier is used by this profile. The following identifier has been assigned to this confirmation method:

urn:oasis:names:tc:SAML:1.0:cm:bearer

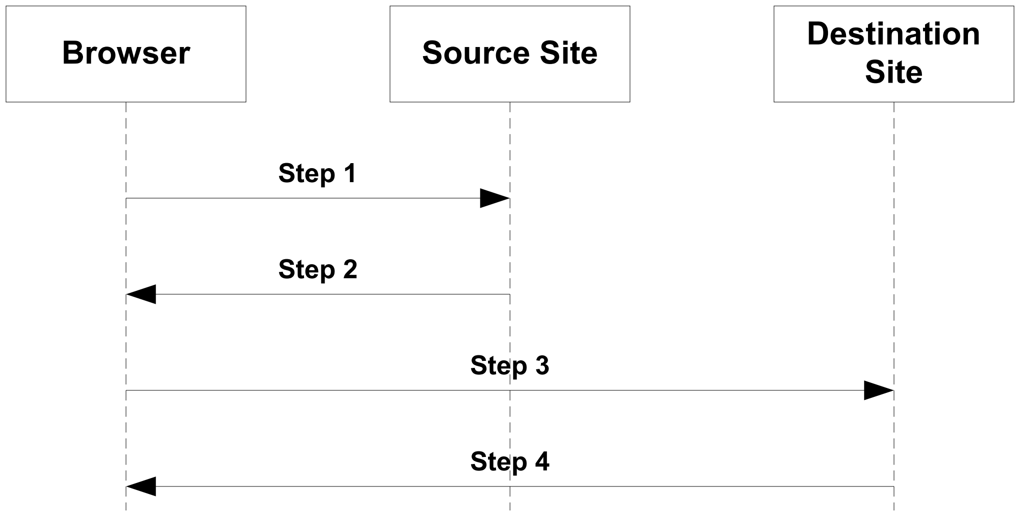
Description: Given below.

Updates: None.

4.1.2.2 Preliminaries

The browser/POST profile of SAML allows authentication information to be supplied to a destination site without the use of an artifact. The following figure diagrams the interactions between parties in the browser/POST profile.

The browser/POST profile consists of a series of two interactions, the first between a user equipped with a browser and a source site, and the second directly between the user and the destination site. The interaction sequence is shown in the following figure, with the following sections elucidating each step.



4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service

In step 1, the user’s browser accesses the inter-site transfer service at host https://<inter-site transfer host name>, with information about the desired target at the destination site attached to the URL.

No normative form is given for step 1. It is RECOMMENDED that the HTTP request take the following form:

```
GET <path>?…TARGET=<Target>…<HTTP-Version>
<other HTTP 1.0 or 1.1 components>
```

Where:

`<inter-site transfer host name>`

This provides the host name and optional port number at the source site where an inter-site transfer service is available.

`<path>`

This provides the path components of an inter-site transfer service URL at the source site.

`Target=<Target>`

This name-value pair occurs in the `<searchpart>` and is used to convey information about the desired target resource at the destination site.

4.1.2.4 Step 2: Generating and Supplying the Response

In step 2, the source site generates HTML form data containing a SAML response message which contains an SSO assertion.

> **Note:** In the browser/POST profile, the URL used to access the assertion consumer service at the destination site is referred to as the assertion consumer URL.

The HTTP response MUST take the form:

```
<HTTP-Version> 200 <Reason Phrase>
<other HTTP 1.0 or 1.1 components>
```

Where:

`<other HTTP 1.0 or 1.1 components>`

This MUST include an HTML FORM (see Chapter 17, [HTML401]) with the following FORM body:

```
<Body>
<FORM Method=”Post” Action=”https://<assertion consumer host name and
path>” …>
<INPUT TYPE=”hidden” NAME=”SAMLResponse” Value=”B64(<response>)”>
…
<INPUT TYPE=”hidden” NAME=”TARGET” Value=”<Target>”>
</Body>
```

`<assertion consumer host name and path>`

This provides the host name, port number, and path components of an assertion consumer URL at the destination site.

Exactly one SAML response MUST be included within the FORM body with the control name `SAMLResponse`; multiple SAML assertions MAY be included in the response. At least one of the assertions MUST be an SSO assertion. A single target description MUST be included with the control name `TARGET`.

The notation `B64(<response>)` stands for the result of applying the Base64 Content-Transfer-Encoding to the response, as defined by [RFC2045] §6.8, and SHOULD consist of lines of encoded data of up to 76 characters. The first encoded line begins after the opening quote signifying the "value" attribute of the SAMLResponse form element.

The character set used to represent the encoded data is determined by the "charset" attribute of the Content-Type of the HTML document containing the form. The character set of the XML document resulting from decoding the data is determined in the normal fashion, and defaults to UTF-8 if no character set is indicated.

The SAML response MUST be digitally signed following the guidelines given in [SAMLCore]. Assertions included in the SAML response MAY be digitally signed.

Confidentiality and message integrity MUST be maintained for step 2. It is RECOMMENDED that the inter-site transfer URL be protected by SSL 3.0 or TLS 1.0 (see Section 6). Otherwise, the assertions returned will be available in plain text to any attacker who might then be able to impersonate the assertion subject.

4.1.2.5 Step 3: Posting the Form Containing the Response

In step 3, the browser submits the form containing the SAML response using the following HTTP request to the assertion consumer service at host https://<assertion consumer host name>.

> **Note:** Posting the form can be triggered by various means. For example, a "submit" button could be included in Step 2 by including the following line:
>
> ```
> <INPUT TYPE="Submit" NAME="button" Value="Submit">
> ```
>
> This requires the user to explicitly "submit" the form for the POST request to be sent. Alternatively, JavaScript™ can be used to avoid an additional "submit" step from the user as follows [Anders]:
>
> ```
> <HTML>
> <BODY Onload="document.forms[0].submit()">
>       <FORM METHOD="POST" ACTION=" https://<assertion consumer host name
> and path>">
>       …
>       <INPUT TYPE="HIDDEN" NAME="SAMLResponse"
>         VALUE="base64 encoded SAML Protocol Response">
>       <INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">
>      </FORM>
> </BODY>
> </HTML>
> ```

The HTTP request MUST include the following components:

```
POST <path> <HTTP-Version>
<other HTTP 1.0 or 1.1 request components>
```

Where:

<assertion consumer host name>

This provides the host name and optional port number at the destination site where the assertion consumer service URL is available.

`<path>`

This provides the path components of the assertion consumer service URL at the destination site.

`<other HTTP 1.0 or 1.1 request components>`

This consists of the form data set derived by the browser processing of the form data received in step 2 according to § 17.13.3 of [HTML401]. Exactly one SAML response MUST be included within the form data set with control name `SAMLResponse`; multiple SAML assertions MAY be included in the response. A single target description MUST be included with the control name set to `TARGET`.

The SAML response MUST include the `Recipient` attribute [SAMLCore] with its value set to https://`<assertion consumer host name and path>`. At least one of the SAML assertions included within the response MUST be an SSO assertion.

The destination site MUST ensure a "single use" policy for SSO assertions communicated by means of this profile.

> **Note:** The implication here is that the destination site will need to save state. A simple implementation might maintain a table of pairs, where each pair consists of the assertion ID and the time at which the entry is to be deleted (where this time is based on the SSO assertion lifetime.). The destination site needs to ensure that there are no duplicate entries. Since SSO assertions containing authentication statements are recommended to have short lifetimes in the web browser context, such a table would be of bounded size.

Confidentiality and message integrity MUST be maintained for the HTTP request in step 3. It is RECOMMENDED that the assertion consumer URL be protected by SSL 3.0 or TLS 1.0 (see Section 6).

Otherwise, the assertions transmitted in step 3 will be available in plain text to any attacker who might then impersonate the assertion subject.

Every subject-based statement in the assertion(s) returned to the destination site MUST contain a `<saml:SubjectConfirmation>` element. The <ConfirmationMethod> element in the <SubjectConfirmation> MUST be set to urn:oasis:names:tc:SAML:1.0:cm:bearer.

4.1.2.6 Step 4: Responding to the User's Request for a Resource

In step 4, the user's browser is sent an HTTP response that either allows or denies access to the desired resource. The TARGET form element may be used to decide how to respond to the request and what resource to return, possibly via a redirect or some other means,

No normative form is mandated for the HTTP response. The destination site SHOULD provide some form of helpful error message in the case where access to resources at that site is disallowed.

4.1.2.7 Threat Model and Countermeasures

This section utilizes materials from [ShibMarlena] and [Rescorla-Sec].

4.1.2.7.1 Stolen Assertion

Threat: If an eavesdropper can copy the real user's SAML response and included assertions, then the eavesdropper could construct an appropriate POST body and be able to impersonate the user at the destination site.

Countermeasure: As indicated in steps 2 and 3, confidentiality MUST be provided whenever a response is communicated between a site and the user's browser. This provides protection against an eavesdropper obtaining a real user's SAML response and assertions.

If an eavesdropper defeats the measures used to ensure confidentiality, additional countermeasures are available:

- The source and destination sites SHOULD make some reasonable effort to ensure that clock settings at both sites differ by at most a few minutes. Many forms of time synchronization service are available, both over the Internet and from proprietary sources.
- SAML assertions communicated in step 3 MUST include an SSO assertion.
- Values for `NotBefore` and `NotOnOrAfter` attributes of SSO assertions SHOULD have the shortest possible validity period consistent with successful communication of the assertion from source to destination site. This is typically on the order of a few minutes. This ensures that a stolen assertion can only be used successfully within a small time window.
- The destination site MUST check the validity period of all assertions obtained from the source site and reject expired assertions. A destination site MAY choose to implement a stricter test of validity for SSO assertions, such as requiring the assertion's `IssueInstant` or `AuthenticationInstant` attribute value to be within a few minutes of the time at which the assertion is received at the destination site.
- If a received authentication statement includes a `<saml:SubjectLocality>` element with the IP address of the user, the destination site MAY check the browser IP address against the IP address contained in the authentication statement.

4.1.2.7.2 MITM Attack

Threat: Since the destination site obtains bearer SAML assertions from the user by means of an HTML form, a malicious site could impersonate the user at some new destination site. The new destination site would believe the malicious site to be the subject of the assertion.

Countermeasure: The destination site MUST check the Recipient attribute of the SAML response to ensure that its value matches the https://`<assertion consumer host name and path>`. As the response is digitally signed, the `Recipient` value cannot be altered by the malicious site.

4.1.2.7.3 Forged Assertion

Threat: A malicious user, or the browser user, could forge or alter a SAML assertion.

Countermeasure: The browser/POST profile requires the SAML response carrying SAML assertions to be signed, thus providing both message integrity and authentication. The destination site MUST verify the signature and authenticate the issuer.

4.1.2.7.4 Browser State Exposure

Threat: The browser/POST profile involves uploading of assertions from the web browser to a source site. This information is available as part of the web browser state and is usually stored in persistent storage on the user system in a completely unsecured fashion. The threat here is that the assertion may be “reused” at some later point in time.

Countermeasure: Assertions communicated using this profile must always include an SSO assertion. SSO assertions are expected to have short lifetimes and destination sites are expected to ensure that SSO assertions are not re-submitted.

5 Confirmation Method Identifiers

The SAML assertion and protocol specification [SAMLCore] defines `<ConfirmationMethod>` as part of the `<SubjectConfirmation>` element. The `<SubjectConfirmation>` element SHOULD be used by the relying party to confirm that the request or message came from the System Entity that corresponds to the subject in the statement. The `<ConfirmationMethod>` element indicates the specific method that the relying party should use to make this judgment. This may or may not have any relationship to an authentication that was performed previously. Unlike the authentication method, the subject confirmation method will often be accompanied by some piece of information, such as a certificate or key, in the `<SubjectConfirmationData>` and/or `<ds:KeyInfo>` elements that will allow the relying party to perform the necessary check.

It is anticipated that profiles and bindings will define and use several different values for `<ConfirmationMethod>`, each corresponding to a different SAML usage scenario. Some examples are as follows:

- A website employs the browser/artifact profile of SAML to sign in a user. The `<ConfirmationMethod>` element in the resulting assertion is set to urn:oasis:names:tc:SAML:1.0:cm:artifact.
- There is no login, but an application request sent to a relying party includes SAML assertions and is digitally signed. The associated public key from the `<ds:KeyInfo>` element is used for confirmation.

5.1 Holder of Key

URI: urn:oasis:names:tc:SAML:1.0:cm:holder-of-key

A `<ds:KeyInfo>` element MUST be present within the `<SubjectConfirmation>` element.

As described in [XMLSig], the `<ds:KeyInfo>` element holds a key or information that enables an application to obtain a key. The subject of the statement(s) in the assertion is the party that can demonstrate that it is the holder of the key.

5.2 Sender Vouches

URI: urn:oasis:names:tc:SAML:1.0:cm:sender-vouches

Indicates that no other information is available about the context of use of the assertion. The relying party SHOULD utilize other means to determine if it should process the assertion further.

5.3 SAML Artifact

URI: urn:oasis:names:tc:SAML:1.0:cm:artifact

The subject of the assertion is the party that presented a SAML artifact, which the relying party used to obtain the assertion from the party that created the artifact. See also Section 4.1.1.1.

5.4 Bearer

URI: urn:oasis:names:tc:SAML:1.0:cm:bearer

The subject of the assertion is the bearer of the assertion. See also Section 4.1.2.1.

6 Use of SSL 3.0 or TLS 1.0

In any SAML 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).

6.1 SAML SOAP Binding

TLS-capable implementations MUST implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite and MAY implement the TLS_RSA_AES_128_CBC_SHA cipher suite [AES].

6.2 Web Browser Profiles of SAML

SSL-capable implementations of the browser/artifact profile or browser/POST profile of SAML MUST implement the SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite.

TLS-capable implementations MUST implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher suite.

7 Alternative SAML Artifact Format

7.1 Required Information

Identification: urn:oasis:names:tc:SAML:1.0:profiles:artifact-02

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

Description: Given below.

Updates: None.

7.2 Format Details

An alternative artifact format is described here:

```
TypeCode            := 0x0002
RemainingArtifact := AssertionHandle SourceLocation
AssertionHandle   := 20-byte_sequence
SourceLocation    := URI
```

The `SourceLocation` URI is the address of the SAML responder associated with the source site. The `assertionHandle` is as described in Section 4.1.1.8, and governed by the same requirements. The `SourceLocation` URI is mapped to a sequence of bytes based on use of the UTF-8 [RFC2279] encoding. The destination site MUST process the artifact in a manner identical to that described in Section 4.1.1, with the exception that the location of the SAML responder at the source site MAY be obtained directly from the artifact, rather than by look-up, based on `sourceID`.

> **Note**: the destination site MUST confirm that assertions were issued by an acceptable issuer, not relying merely on the fact that they were returned in response to a `<samlp:Request> message`.

8 URL Size Restriction (Non-Normative)

This section describes the URL size restrictions that have been documented for widely used commercial products.

A Microsoft technical support article [MSURL] provides the following information:

> The information in this article applies to:
>
> Microsoft Internet Explorer (Programming) versions 4.0, 4.01, 4.01 SP1, 4.01 SP2, 5, 5.01, 5.5
>
> SUMMARY
>
> Internet Explorer has a maximum uniform resource locator (URL) length of 2,083 characters, with a maximum path length of 2,048 characters. This limit applies to both POST and GET request URLs.
>
> If you are using the GET method, you are limited to a maximum of 2,048 characters (minus the number of characters in the actual path, of course).
>
> POST, however, is not limited by the size of the URL for submitting name/value pairs, because they are transferred in the header and not the URL.
>
> RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, does not specify any requirement for URL length.
>
> REFERENCES
>
> Further breakdown of the components can be found in the Wininet header file. Hypertext Transfer Protocol -- HTTP/1.1 General Syntax, section 3.2.1
>
> Last Reviewed: 9/13/2001
>
> Keywords: kbDSupport kbFAQ kbinfo KB208427
>
> An article about Netscape Enterprise Server provides the following information:
>
> Issue: 19971110-3 Product: Enterprise Server
>
> Created: 11/10/1997 Version: 2.01
>
> Last Updated: 08/10/1998 OS: AIX, Irix, Solaris
>
> Does this article answer your question?
>
> Please let us know!
>
> Question:
>
> How can I determine the maximum URL length that the Enterprise server will accept? Is this configurable and, if so, how?
>
> Answer:
>
> Any single line in the headers has a limit of 4096 chars; it is not configurable.

9 References

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

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

[CoreAssnEx] Core Assertions Architecture, Examples and Explanations, http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf.

[HTML401] HTML 4.01 Specification, W3C Recommendation 24 December 1999, http://www.w3.org/TR/html4.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[SAMLSec] E. Maler et al. Security Considerations for the OASIS *Security Assertion Markup Language (SAML)*, OASIS, September 2003, Document ID oasis-sstc-saml-sec-consider-1.1. http://www.oasis-open.org/committees/security/.

[SAMLReqs] Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002, http://www.oasis-open.org/committees/security/.

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

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

[ShibMarlena] Marlena Erdos, Shibboleth Architecture DRAFT v1.1, http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html .

[SOAP1.1] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium Note, May 2000, http://www.w3.org/TR/SOAP.

[SSL3]	A. Frier et al., *The SSL 3.0 Protocol*, Netscape Communications Corp, November 1996.
[WEBSSO]	RL “Bob” Morgan, Interactions between Shibboleth and local-site web sign-on services, http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt
[WSS-SAML]	P. Hallam-Baker et al., *Web Services Security: SAML Token Profile*, OASIS, March 2003, http://www.oasis-open.org/committees/wss.
[XMLSig]	D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web Consortium, http://www.w3.org/TR/xmldsig-core/.

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:

- TBD

976 ## B. Revision History

Rev	Date	By Whom	What
00	15 Sept 2003	Frederick Hirsch	Initial draft for SAML 2.0 from SAML 1.1 Standard - changed status and date, removed TC and contributor lists, changed editor list, imported template styles
01	10 Oct 2003	Frederick Hirsch	Removed deprecated Status as child of Body, also removed artifact URI - Addresses portion of action item 82
02	02 Jan 2004	Frederick Hirsch	Update to Spectools 03 Nov 03 template, updated formats, added revision history

C. Notices