SAML 2.0 Session Token Profile Version 1.0

Committee Specification Draft 01

22 February 2011

Specification URIs:

This Version:

http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/csd01/saml-session-token-v1.0-csd01.odt (Authoritative)

http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/csd01/saml-session-token-v1.0-csd01.html

http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/csd01/saml-session-token-v1.0-csd01.pdf

Previous Version:

N/A

Latest Version:

http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/saml-session-token-v1.0.odt (Authoritative)

http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/saml-session-token-v1.0.html

http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/saml-session-token-v1.0.pdf

Technical Committee:

OASIS Security Services (SAML) TC

Chair(s):

Thomas Hardjono, MIT

Nathan Klingenstein, Internet2

Editor(s):

Hal Lockhart, Oracle

Related Work:

XML Schema(s): saml-session-token/v1.0/csd01/xsd/

Declared XML Namespace(s):

urn:oasis:names:tc:SAML:2.0:profiles:session:metadata

Abstract:

Web Servers and Application Servers generally maintain security state information for currently active users, particularly once some type of authentication has occurred. This specification defines a format for communicating such security session state based on the OASIS SAML Assertion. It also specifies two different mechanisms for communicating this information between servers via a standard Web browser.

Status:

This document was last revised or approved by the OASIS Security Services (SAML) TC on the above date. The level of approval is also listed above. Check the "Latest Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at 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 section of the Technical Committee web page (http://www.oasis-open.org/committees/security/ipr.php).

Citation Format:

When referencing this specification the following citation format should be used:

[SAML-SESSION-TOKEN]

SAML 2.0 Session Token Profile Version 1.0. 22 February 2011. OASIS Committee Specification Draft. http://docs.oasis-open.org/security/saml/Post2.0/saml-session-token/v1.0/csd01/saml-session-token-v1.0-csd01.odt.

Notices

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS" and “SAML” are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction (non-normative)

Although the HTTP protocol [RFC2616] is deliberately stateless, efficient implementation of security requirements such as attribute-based authorization and inactivity timeout require maintaining state associated with each active connection. This state may consist of historical information (authentication occurred), relatively static information (user’s attributes) and dynamic information (time of last interaction).

Web applications are commonly implemented by passing requests from browsers to any of a number of servers. These servers may be heterogeneous or homogeneous in function, geographically centralized of distributed. Typically users are unaware that multiple servers are involved. It is therefore desirable to simulate a single system with uniform knowledge and behavior.

This means that a server receiving a request from a browser that last interacted with a different server must have a means to obtain the most recent session state. The only practical method of doing this is to pass the information via the browser using an HTTP cookie [RFC2965]]. The cookie may be used either to pass the encoded session token itself, or if it is too large, to pass a reference to the token.

1.1 Terminology

The keywords "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].

Conventional XML namespace prefixes are used throughout the listings in 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
ds:	http://www.w3.org/2000/09/xmldsig#	This namespace is defined in the W3C XML Schema specification
`md:`	urn:oasis:names:tc:SAML:2.0:metadata	This is the SAML V2.0 metadata namespace [SAML2Meta].
`mdsess:`	urn:oasis:names:tc:SAML:2.0:profiles:session:metadata	This is the SAML V2.0 metadata extension namespace defined by this document and its accompanying schema [MDSESS-XSD]

1.2 Normative References

[MDSESS-XSD] OASIS Working Draft 01, *Metadata Extension Schema for Session Token Profile,* February 2011,

[RFC1951] P. Deutsch, *DEFLATE Compressed Data Format Specification version 1.3,* IETF RFC 1951, May 1996. http://www.ietf.org/rfc/rfc1951.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

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

[RFC2965] D. Kristol, L. Montulli, *HTTP State Management Mechanism,* IETF RFC 2965, October 2000, http://www.ietf.org/rfc/rfc2965.txt

[RFC3513] R. Hinden, S.Deering, *Internet Protocol Version 6 (IPv6) Addressing Architecture.* IETF RFC 3513, April 2003. http://www.ietf.org/rfc/rfc3513.txt

[RFC3986] T. Berners-Lee, et. al. *Uniform Resource Identifier (URI): Generic Syntax,* IETF RFC 3986, January 2005. http://www.ietf.org/rfc/rfc3986.txt

[RFC4648] S. Josefsson, *The Base16, Base32, and Base64 Data Encodings,* IETF RFC 4648, October 2006. http://tools.ietf.org/rfc/rfc4648.txt

[SAML2Bind] OASIS Standard, *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[SAML2Core] OASIS Standard, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[SAML2Meta] OASIS Standard, *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[SAML2Prof] OASIS Standard, *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[SAML2AuthnCtx] S. Cantor et al. *Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf

[SAML2IdAssure] R. Morgan et al, *SAML V2.0 Identity Assurance Profiles,* OASIS SSTC, August 2010, http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-assurance-profile-cd-02.odt

[XMLSig] D. Eastlake et al. *XML Signature Syntax and Processing, Second Edition*. World Wide Web Consortium, June 2008. http://www.w3.org/TR/xmldsig-core/

2 Session Management Architectures (non-normative)

In this document the server providing session information is called the Session Authority (SA) and the server using the information is called the Session Consumer (SC). These roles operate only in the context of a single interaction. Usually servers will take on each role in turn. The token is created by the SA and read by the SC.

Session management can be implemented using a variety of architectures. For example, each Web or Application server can implement a session management capability internally as shown in Figure 1. In this case each server acts as both SA and SC.

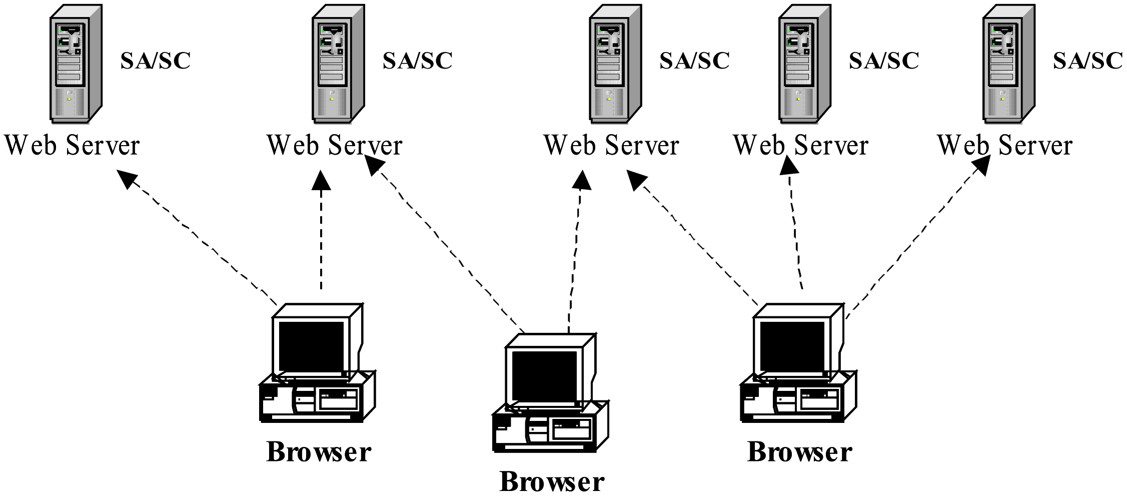

Figure 1 – Every Server a Session Manager

Session management can also be implemented by one or more dedicated session management servers as shown in Figure 2. These are accessed as needed by web and application servers. Depending on the specific design the session manager may act as SA and SC or the roles may be divided between the session manager and web servers.

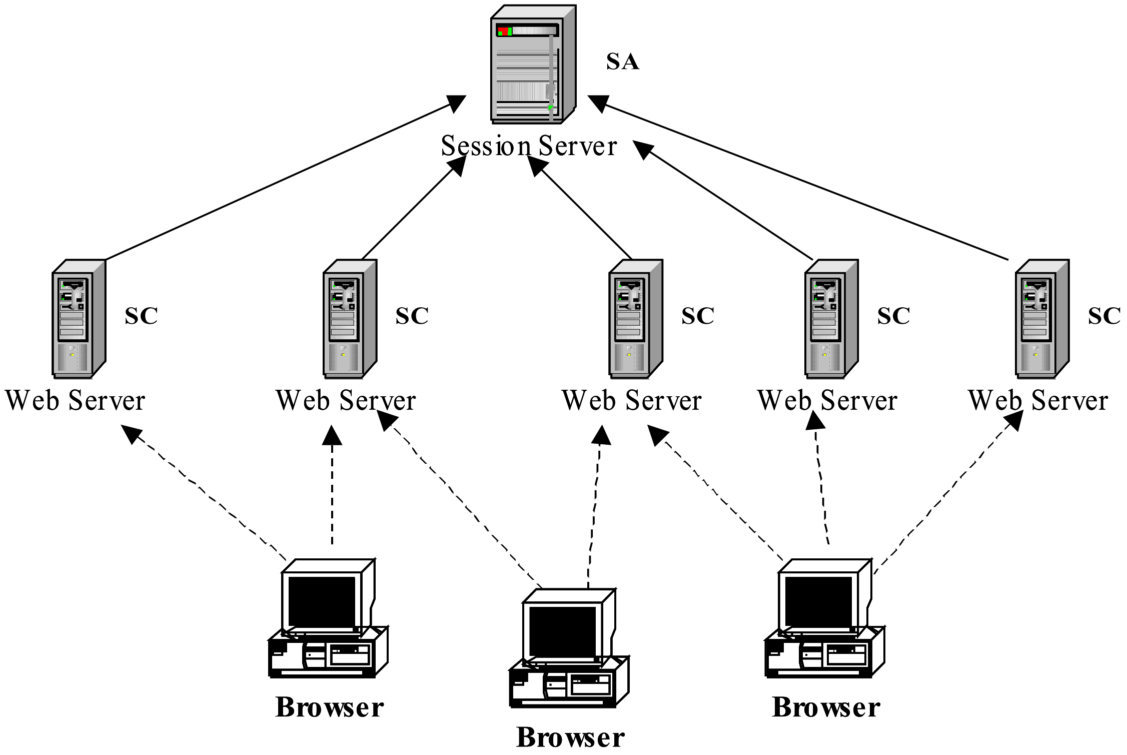

Figure 2 – Dedicated Session Management Servers

3 Session Management Algorithm (normative)

This section describes the processing used to by a server which is acting as both an SA and SC. There are two variants, depending on whether the cookie contains the Token or is a reference to the Token.

3.1 Stateful Token Algorithm

When the session state is encoded into the cookie, interactions are entirely between web browsers and session managers. There is no direct communications between the SA and SC as shown in Figure 3.

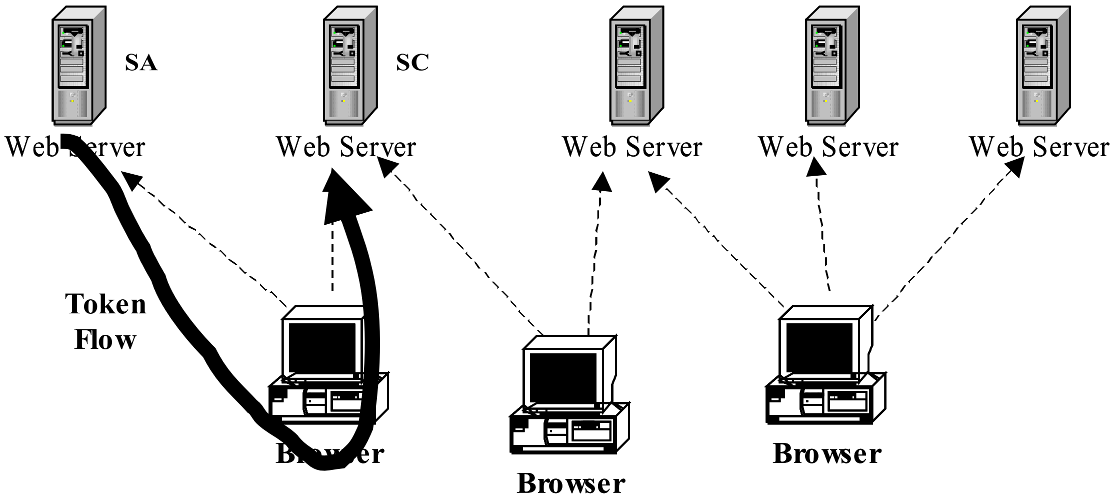

Figure 3 – Stateful Cookie

1. When an application request is received, the SC first checks to see if a session cookie of the type supported (stateful or reference) is present. The name of the supported cookie type MAY be obtained from metadata. If the cookie is not present, the SC MUST proceed as it would with any request from a user who has not authenticated. Depending on the request this may mean permitting it, causing authentication to be performed or taking some other action.
2. If the cookie contains a session reference, the SC MUST use the reference to obtain the cookie as described in Section 3.2. If the cookie is stateful, it contains the Token. In either case processing continues with the next step.
3. The SC must verify the signature of the Token. The ability to determine the correct key to use for this purpose implies some type of key management function. If the signature is not valid, the SC MUST discard the request with no action, so as to reduce the effect of denial of service attacks by unauthorized users. (Administrative reporting of potential attacks may occur.) If the signature is not present and the Token was not received over a secure channel, the SC SHOULD discard the request.
4. The `<saml:Conditions>` element MUST be checked for validity as described in Section 2.5 of [SAML2Core]. If the Token is not valid, the SC MUST treat the request as unauthenticated. Other checks MAY be performed to ensure the Token contains the required information.

5. The `Address` XML Attribute of the `<saml:SubjectConfirmationData>` element in the Token MAY be compared to the IP address from which the request originated and if they are different, the request discarded.

6. Idle time out MAY be implemented by configuring each SC with a maximum idle time value. Typically, the value will be the same for all SCs hosting the same application type, but this algorithm does not depend on this being the case. It is simply assumed that each SC is configured with a maximum idle time value by some means unspecified in this document. In practice, maximum idle time values might range from 5 minutes to 30 minutes.
   If idle timeout is enabled, the SC subtracts the value of the `urn:oasis:names:tc:SAML:2.0:profiles:session:timeLastActive` SAML Attribute from the current time and compares the result to the maximum idle time value. If the difference exceeds the maximum value, the Token is discarded, any existing session information for that user is cleared and the user is informed that the session has timed out because of inactivity. The request MUST be treated as unauthenticated.

7. Maximum login time (sometimes called session time limit) MAY be implemented by configuring each server with a maximum login time value. This may be a single value or depend on the type of login performed most recently. Maximum login time limits typically range from 1 hour to 24 hours.
   If maximum login time is enabled, the SC subtracts the value of the `AuthnInstant` XML Attribute of the `<saml:AuthnStatement>` from the current time and compares the result to the maximum login time. If the time since the last authentication exceeds the maximum value, the request MUST be treated as unauthenticated.

8. After these checks, the SC MAY make use of the information in the Token, for authorization, personalization or other purposes.

9. When the HTTP response is sent, the server acts as a Session Authority (SA). If a stateful cookie is being employed, the SC MUST construct a Token containing the current values as described in Section 4. The Token is then signed and inserted in the cookie of the response.
   If a session reference cookie is being employed, the SA MUST generate the session reference value and insert the URL and reference in the cookie as described in Section 6. The SA MUST implement a responder at the given URL which returns a Token with the same contents as would have been put in a stateful cookie. The SA MAY generate the Token in advance or at the time it is requested.

10. As an optimization, the server MAY maintain a Token Freshness value, which allows Tokens to be reused if they were created recently. For example, the value might be something like 30 seconds. If the value of the `IssueInstant` XML Attribute of the `<saml:AuthnStatement>` subtracted from the current time is less the Token Freshness value, the received Token (or session reference) is put in the cookie instead of creating and signing a new Token. This reduces the overhead of a series of closely spaced requests at the cost of reducing the precision of the idle timeout and maximum login time algorithms.

3.2 Session Reference Algorithm

Instead of the cookie containing the Token, it MAY instead merely contain a reference to the session. The actual session Token is obtained by making a query to the SA which generated the reference. In this case the cookie contains two parts: a server endpoint in the form of a URI and a large random number. In this case, the SA and SC communicate directly as shown in Figure 4

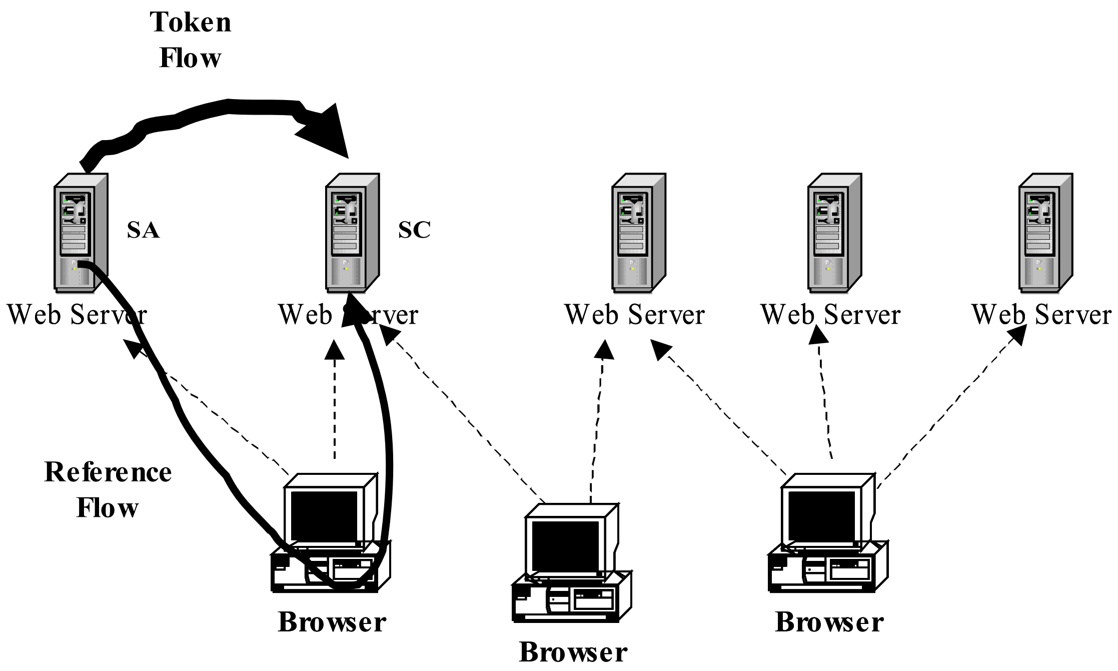

Figure 4 – Session Reference Cookie

The SC MUST call the indicated endpoint, providing the reference as an input value, as described in Section 6. The SA checks to see if the reference corresponds to a valid session. If not, it MUST return an error. If it does correspond to a valid session, the SA must return a session Token, constructed as described above. If this back channel connection is integrity protected, e.g. using TLS, then the SA MAY choose not to sign the Token. The SC MUST process the Token as described in section 3.1 beginning with step 3.

4 Token Format (normative)

The format of the Session Token is based on the `<saml:Assertion>` element defined by [SAML2Core]. The Assertion MUST contain exactly one `<saml:AuthnStatement>` element and at exactly one `<saml:AttributeStatement>` element. The contents of the Assertion and the Statements are specified in the following sections.

4.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:profiles:session

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

Description: Given below.

Updates: None.

4.2 Assertion Header

The assertion header MUST contain the following items.

`Version` [Required]

The SA MUST set the value of the `saml:Version` attribute to “2.0” as required by [SAML2Core]. The SC SHOULD verify this value.

`ID` [Required]

The SA MUST set the value of the `saml:ID` or `xs:ID` to a unique identifier as required by [SAML2Core].

`IssueInstant` [Required]

The SA MUST set the value of the `saml:IssueInstant` to the time the Token was created as required by [SAML2Core]. When the cookie contains a session reference, it MAY differ from the user’s `TimeLastActive`.

`<saml:Issuer>` [Required]

The Session Authority MUST set this value to its own name.

`<ds:Signature>` [Optional]

When the Assertion is carried in a cookie, the SA MUST sign it. See Section 5. If the Assertion is signed, the SC MUST verify the signature before processing it.

`<saml:Subject>` [Required]

The SA MUST create a `<saml:Subject>` element containing the following Elements and Attributes except as noted below.

`<saml:NameID>` [Optional]

Any deployment of this specification MUST profile the use of the `NameID` element and its associated Attributes: `NameQualifier`, `SPNameQualifier`, `Format` and `SPProviderID`. This includes making their use required, prohibited or optional.

`<saml:SubjectConfirmation>` [Required]

The SA MUST include a `<saml:SubjectConfirmation>` which contains a Subject Conformation `saml:Method` attribute.

`Method` [Required]

The Subject Confirmation `saml:Method` MUST have a value of

```
urn:oasis:names:tc:SAML:2.0:cm:bearer
```

`<saml:SubjectConfirmationData>` [Required]

The SA MUST set the `<saml:SubjectConfirmationData>` element to have the following attribute.

`Address` [Required]

The SA MUST set the value of the `saml:Address` attribute to contain the address of the browser in IPv4 dotted decimal format, e.g. “198.51.100.1” or in IPv6 address format as described in Section 2.2 of [RFC3513], e.g.,"2001:db8::1". The SC MAY compare the value to the known address of the browser.

`<saml:Conditions>` [Required]

The SC MUST set the `<saml:Conditions>` element to contain the following attributes.

`NotBefore` [Required]

`NotOnOrAfter` [Required]

The SA MUST set these so as to delimit the validity interval of the Token. The SC MUST check the conditions element, including the validity interval as specified in section 2.5 of [SAML2Core].

`<saml:Advice>` [Prohibited]

The SA MUST NOT include an `<saml:Advice>` element in the Token.

The SA MAY include any other elements or attributes specified in [SAML2Core] which are not explicitly required or prohibited by this document.

4.3 Authentication Statements

The Assertion MUST contain exactly one `<saml:AuthnStatement>` element. It MUST contain the following XML attribute.

`AuthnInstant` [Required]

The SA MUST set the `AuthnInstant` to the time authentication occurred, as defined in [SAML2Core]. The SC MAY use this value to implement a maximum login time.

`<saml:AuthnContext>` [Required]

The contents of the Authentication Context MUST conform to [SAML2AuthnCtx].

The SA MUST set the Authentication Strength attribute in the Attribute Statement, (see section 4.3), to correspond to the value assigned to the authentication method present in the Authentication Statement.

The level of assurance (LOA) associated with this Authentication MAY be expressed as specified in [SAML2IdAssure].

4.4 Attribute Statement

The Assertion MUST contain exactly one `<saml:AttributeStatement>` element.

The following SAML Attributes MUST be present.

Session Id

This attribute has a name format type of `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`. The name of the attribute is `urn:oasis:names:tc:SAML:2.0:profiles:session:sessionId`.

The value of this attribute is of type string and the SA MUST set it to contain the unique identifier of the session. (This is not the same as the session reference described in section 6.) The SC MAY use this value as an index to the stored session information.

Authentication Strength

This attribute has a name format type of `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`. The name of the attribute is `urn:oasis:names:tc:SAML:2.0:profiles:session:authenticationStrength`.

The value of this attribute is of type integer in the range of 0-99. It is a deployment-specific value associated with every type of Authentication supported by the deployment, where a higher number represents a more secure method. The SA MUST set the value of the attribute to correspond to the value assigned to the authentication method represented in the Authentication Statement present in the Assertion. Authentication method is defined as a specific Authentication Context Class with specific instance values or ranges of values.

The means by which the mapping of Authentication methods to AuthenticationStrength is communicated to SAs and SCs is outside the scope of this Profile.

Time Last Active

This attribute has a name format type of `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`. The name of the attribute is `urn:oasis:names:tc:SAML:2.0:profiles:session:timeLastActive`.

The SA MUST set the value to contain the datetime of the completion of the last request. The SC MAY use this value implement an idle timeout algorithm.

Token Format Version

This attribute has a name format type of `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`. The name of the attribute is `urn:oasis:names:tc:SAML:2.0:profiles:session:tokenFormatVersion`.

The SA MUST set the value to contain a string value contain the major and minor version numbers of the Token format being used, e.g. “2.3”. The Token format version is the same as the version of this Profile, that is: “1.0”.

The Attribute Statement MAY contain other Attributes as specified in [SAML2Core].

5 Token Carried in Cookie (normative)

If size allows, the session token MAY be carried in the cookie. The cookie name can be determined by out of band agreement or via metadata.

When the token is carried in the cookie, it MUST be signed as specified in [SAML2Core]. The Token MAY also be encrypted as specified in [SAML2Core].

5.1 Compression

The Token MAY be compressed to reduce its size. Compression MUST be done after signing and encryption. The only compression method specified by this document is the DEFLATE algorithm. [RFC1951] After compression the resulting binary string MUST be encoded using Base64.[RFC4648]

The use of compression MAY be indicated via metadata. Implementations MAY define alternative compression methods and corresponding metadata values.

6 Session Reference Carried in Cookie (normative)

Instead of transmitting the Assertion in the cookie, the SA MAY instead put a reference to the Assertion in the cookie. The reference then MAY be used to retrieve the Assertion.

When this approach is used, the cookie value MUST consist of an HTTP scheme URL followed by the “?” character, followed by “ID=” followed by an unguessable number of at least 256 bits represented as a positive decimal integer. The entire value MUST be percent encoded as described in Section 2 of [RFC3986].

The URL represents a server endpoint which supports the SAML URI Binding as specified in [SAML2Bind].

The SA using this scheme MUST respond to protocol requests by returning the indicated Assertion with the session information.

The Token MUST be carried over secure transport and/or signed as specified in [SAML2Core]. The Token MAY also be encrypted as specified in [SAML2Core].

7 Metadata (normative)

This section defines metadata which MAY be used to communicate cookie names and other properties associated with a Session Authority.

The SAML V2.0 metadata specification [SAML2Meta] defines the following namespace:

```
urn:oasis:names:tc:SAML:2.0:metadata
```

By convention, the namespace prefix `md:` is used to refer to the above namespace.

This specification defines a new namespace:

```
urn:oasis:names:tc:SAML:2.0:profiles:session:metadata
```

The prefix `mdsess:` is used here and in the accompanying schema to refer to this new namespace. In what follows, any unqualified element or type is assumed to belong to this new namespace.

7.1 Element <md:RoleDescriptor>

The `<md:RoleDescriptor>` element defined in [SAML2Meta] is an abstract extension point that contains descriptive information common across various entity roles. New roles can be defined by extending its abstract **md:RoleDescriptorType** complex type, which is the approach taken here.

7.2 CookieName and CookieNameType

Complex type `mdsess:CookieNameType` holds information intended to describe cookies used by this profile. The `<mdsess:CookieName>` element is defined to be of type `mdsess:CookieNameType`. The value of the `<mdsess:CookieName>` element is a string which is the cookie name. It contains the following XML attributes.

`CookieContent` [Required]

Required attribute that indicates the format of the content of the cookie. The values defined by this specification are:

`urn:oasis:names:tc:SAML:2.0:profiles:session:metadata:token`

This indicates that the SAML Assertion is carried in the cookie as described in Section 5 of this document.

`urn:oasis:names:tc:SAML:2.0:profiles:session:metadata:reference`

This indicates that the cookie contains a reference to the Token as described in Section 6 of this document.

`CookieCompression` [Optional]

Optional attribute that indicates what kind of compression, if any has been performed on the contents of the cookie. If the attribute is not present it indicates no compression has been done. The values defined by this specification are:

`urn:oasis:names:tc:SAML:2.0:profiles:session:metadata:nocompression`

This indicates that no compression has been done.

`urn:oasis:names:tc:SAML:2.0:profiles:session:metadata:rfc1951`

This indicates that the contents of the cookie have been compressed using the DEFLATE algorithm as described in Section 5.1 of this document.

The following schema fragment defines the `<mdsess:CookieName>` element and `mdsess:CookieNameType` complex type:

```
<element name="CookieName" type="mdsess:CookieNameType">

  <complexType name="CookieNameType" >
    <simpleContent>
      <extension base="string">
        <attribute name="CookieContent" type="anyURI" use="required"/>
        <attribute name="CookieCompression" type="anyURI" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>
```

7.3 Complex Type SessionAuthorityDescriptorType

Complex type **SessionAuthorityDescriptorType** extends complex type `<md:RoleDescriptor>` to represent information about SessionAuthorities.. It adds the `<mdsess:CookieName>` element to the items defined by the `<md:RoleDescriptor>`.

The following schema fragment defines the **SessionAuthorityDescriptorType** complex type:

```
   <complexType name="SessionAuthorityDescriptorType" >
    <complexContent>
      <extension base="md:RoleDescriptorType">
        <sequence>
          <element ref="mdsess:CookieName" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
</schema>
```

8 Example (non-normative)

The following is an example of a session token.

```
<saml:Assertion ID="_a75e1c55-01d7-40cc-929f-d627c72ebdfc"
    IssueInstant="2010-11-25T13:16:02Z" Version="2.0"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
    xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
    xmlns:xs=”http://www.w3.org/2001/XMLSchema”
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  <saml:Issuer>sessionauthority.example.com</Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <ds:SignatureMethod
        Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256"/>
      <ds:Reference URI="#_a75e1c55-01d7-40cc-929f-d627c72ebdfc">
        <ds:Transforms>
          <ds:Transform
            Algorithm="http://www.w3.org/2000/09/xmldsig#envelopedsignature"
/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <InclusiveNamespaces PrefixList="#default saml ds xs xsi"
              xmlns="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>Kcl ... </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> ... </ds:SignatureValue>
    <ds:KeyInfo>
      <ds:KeyName>SessionKey003<ds:KeyName/>
    </ds:KeyInfo>
  </ds:Signature>
  <saml:Subject>
    <saml:NameID NameQualifier="Repository6">John.Smith</NameID>
    <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"
      <saml:SubjectConfirmationData Address="192.168.1.2"
    </saml:SubjectConfirmation>
  </saml:Subject>
  <saml:Conditions NotBefore="2010-11-25T13:16:02Z"
    NotOnOrAfter="2010-11-25T13:20:02Z">
  </saml:Conditions>
  <saml:AuthnStatement AuthnInstant="2010-11-25T13:15:13Z">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
          urn:oasis:names:tc:SAML:2.0:ac:classes:Password
      </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
  <saml:AttributeStatement>
    <saml:Attribute NameFormat=
                  "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:session:sessionId"
      xsi:type="xs:string" >
        258673
    </saml:Attribute>
    <saml:Attribute NameFormat=
                  "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oasis:names:tc:SAML:2.0:profiles:session:AuthenticationSt
rength"
      xsi:type="xs:integer" >
 >
        20
    </saml:Attribute>
    <saml:Attribute NameFormat=
                   "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oasis:names:tc:SAML:2.0:profiles:session:TimeLastActive"
    xsi:type="xs:dateTime" >
        2010-11-25T13:16:02Z
    </saml:Attribute>
    <saml:Attribute NameFormat=
                   "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
  Name="urn:oasis:names:tc:SAML:2.0:profiles:session:TokenFormatVersion"
      xsi:type="xs:string" >
       1.0
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

For the purpose of this example, it is assumed that the deployment as assigned and **AuthenticationStrength** value of 20 to the password authentication method.

9 Security Considerations (non-normative)

The short summary is that this proposal has essentially the same security properties as existing deployed products.

The primary threats are: 1) Token forgery, 2) Token capture and unauthorized use and 3) unauthorized disclosure of Token contents.

When the Assertion is carried in the cookie, the signature will prevent forgery.

Capture of the Token as it traverses the network use can easily be prevented by protecting the browser session with TLS. This has been rare in past because of performance concerns. However, recently Google has publicized work showing that Running TLS has a minimal effect on capacity and throughput. They are also working on reducing latency, particularly in the initial handshake.

Depending on the application, it may be possible to capture a cookie via a cross-site scripting exploit. This can be mitigated by setting the HttpOnly attribute to the cookie. While this has not yet been standardized by the IETF yet, most browsers implement it by not allowing a cookie so marked to be accessed from a script.

Cookies can also be subject to interception if presented to some web sites without using TLS. Setting the “Secure” property on the cookie as specified in [RFC2965]. Cookies may also be captured if any server in the domain is controlled by an attacker, whether or not TLS is used.

IP address checking will generally be effective in preventing this type of impersonation, but the widespread use of Network Address Translation (NAT) makes this questionable. It would seem that an attacker who could intercept messages from a point along the network path from browser to server and could also transmit from that point, could spoof the IP address. Encrypting the Assertion would hide the IP Address there, but it would still appear in the IP header.

Another threat is that one sever could take the token from a user and use it to impersonate that user to another server. This scheme assumes that servers can be trusted not to do this, just as they are trusted not to misuse the passwords users type in.

If unauthorized disclosure is a concern, the Assertion can be encrypted as specified in [SAML2Core]. However, if an unauthorized party can obtain a copy of the token, whether encrypted or not, it can be presented to impersonate the user. Therefore the utility of encrypting the Assertion is unclear. Generally, exposure of a user’s session state information to that user will not be considered a threat.

When the cookie carries only a reference, no integrity check is required. If the value is invalid, the SAML request will fail. (Technically SAML will return an empty response.) Again, interception of the cookie will permit impersonation, but this seems to be a threat to any cookie-based scheme.

10 Conformance

A Session Authority conforms to this specification if it

- generates Assertions conforming to Section 3 and 4,
- uses the cookie naming scheme specified in Section 7, and
- transmits the Assertion using the method defined in Section 5 or Section 6.

A Session Consumer conforms to this specification if it

- can process an Assertion as specified in Section 3 and 4,
- can process a cookie named as specified in Section 7, and
- access an Assertion using the method defined in Section 5 or Section 6.

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged

Participants:

[Participant name, affiliation | Individual member]
[Participant name, affiliation | Individual member]
[Participant name, affiliation | Individual member]

Appendix B. Non-Normative Text

Appendix C. Revision History

- WD01 Initial version
- WD02 – Removed Cookie Naming, Added Required Information, Changed protocol to URI Binding
- WD03 – Added example session token.
- WD04 – Make processing algorithm stateless, allow NameID to be omitted from Subject, remove session start time, allow optional compression, define metadata, various corrections and improvements
- WD05 – Remove `saml:` prefix from XML Attributes, Change validation to refer to SAML Core, Fix metadata schema, various editorial and format fixes.
- WD06 – Correct introductory sentence of section 4 to indicate not all elements are required and mark individual elements and attributes as required, optional or prohibited.