Proposal for SAML Attribute Changes

Proposal 02, 21 February 2004

Document identifier:

sstc-maler-w28a-attribute-draft-02

Location:

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

Previous draft:

http://www.oasis-open.org/committees/download.php/5336/sstc-maler-w28a-attribute.pdf

Author:

Eve Maler, Sun Microsystems (eve.maler@sun.com)

Contributor:

Rebekah Lepro, NASA Ames Research Center

Abstract:

This document proposes a set of solutions that meet the requirements and goals expressed in Rebekah Lepro's **Attribute Representation in SAML 2.0** document [AttribRep]. Portions of that document have been reproduced here in order to give the full context for attribute-related requirements and changes in SAML V2.0.

Status:

Please send comments to the author.

Rev 01: 6 Feb: Initial draft.

Rev 02: 21 Feb: Includes relevant requirements and proposals from Rebekah's paper.

Table of Contents

1 Existing Schema for Attributes....3
2 Goals and Requirements....4
2.1 Allow for Alignment with Existing Attribute Representations....4
2.2 Identify Consistent Attribute Datatypes....4
2.3 Standardize Semantics of Attribute Naming and Metadata....4
2.4 Cleanly Handle Null-Valued and Multi-Valued Attributes....4
2.5 Make Attribute Complexity Match Power....5
3 Proposed Changes....6
3.1 Handling Datatypes that Map to Other Systems....6
3.2 Clarifying Naming and Adding Metadata....6
3.3 Handling Null-Valued and Multi-Valued Attributes....7
3.4 Keeping the Changes Simple....8
4 References....9

1 Existing Schema for Attributes

Following is the SAML V1.1 assertion schema snippet related to attributes:

```
<element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
<complexType name="AttributeDesignatorType">
    <attribute name="AttributeName" type="string" use="required"/>
    <attribute name="AttributeNamespace" type="anyURI" use="required"/>
</complexType>
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
    <complexContent>
        <extension base="saml:AttributeDesignatorType">
            <sequence>
                <element ref="saml:AttributeValue" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="AttributeValue" type="anyType"/>
```

This results in instances like this in a query:

```
<AttributeDesignator
  AttributeName="any-name"
  AttributeNamespace="URI-representing-set-of-att-names"/>
```

And in instances like this in an assertion sent in response:

```
<Attribute
  AttributeName="any-name"
  AttributeNamespace="URI-representing-set-of-att-names">
any-string-or-structured-value
</Attribute>
```

2 Goals and Requirements

Note: In this proposal, “attribute” always refers to a SAML attribute or similar piece of information about a subject. When XML attribute markup is meant, it is called a “field” or an “XML attribute”.

The earlier attribute proposal [AttribRep] and the TC's discussion on 5 February 2004 [F2FMinutes] highlighted the following goals.

2.1 Allow for Alignment with Existing Attribute Representations

SAML needs to prepare for alignment with LDAP and with XACML's attribute handling, but in a way that doesn't massively inconvenience existing SAML attribute statement users who have no desire to do this mapping.

One critical use case of SAML attribute exchange is the provision of attributes to a policy evaluation process, such as XACML defines. Often, a policy can be represented directly in terms of attribute designators, such as the XACML policy representation. This requires that all information needed to represent that policy in terms of attributes must be available.

Following are the basic differences between SAML's and XACML's attribute representations:

- SAML has two fields that contribute to a unique attribute name, `AttributeName` and `AttributeNamespace`. XACML has a single URI-based field.
- SAML allows specification of an attribute's datatype only through XSD means. XACML has a field for a URI-based datatype identifier.
- SAML supplies issuer information only at the assertion level. XACML supplies it per attribute.

Following are the basic differences between SAML's and X.500/LDAP's attribute representations (refer to Bob Morgan's proposal [MorganX500] for suggested conventions on how to map SAML attributes to X.500/LDAP):

- The X.500/LDAP concept of OIDs and LDAP's ability to provide an attribute's short name has only a loose resemblance to SAML's ability to represent typed attribute namespaces and names.
- X.500 and LDAP have a native ability to represent attribute schemas that themselves have OIDs, whereas SAML relies entirely on XSD (or on out-of-band means) for schemas and constraints.

2.2 Identify Consistent Attribute Datatypes

SAML needs to provide the ability to determine the expected type of an attribute value whether or not an attribute value is present. Several consumers of attribute statements require such datatype information, as a function of the attribute rather than the attribute value. There is currently no way to exchange this information in-band within a SAML assertion that does not contain an attribute value or a SAML query.

Also, SAML needs to allow for datatype information to be supplied consistently for all the values (if there are more than one) of an attribute.

2.3 Standardize Semantics of Attribute Naming and Metadata

Currently, SAML's `AttributeNamespace` field is used in several inconsistent ways. One way is to provide scoping, administrative domain, or sourcing information about the attribute, which may be general needs.

2.4 Cleanly Handle Null-Valued and Multi-Valued Attributes

SAML needs to provide the clear, interoperable ability to represent null-valued and multi-valued attributes within a single XML element.

2.5 Make Attribute Complexity Match Power

SAML needs to ensure that any new (and existing) attribute features provide only enough schema complexity to match the power gained therefrom.

3 Proposed Changes

Following are proposed changes, taking into account the goals, requirements, and other proposals made to date.

3.1 Handling Datatypes that Map to Other Systems

To satisfy goals 2.1 and 2.2 regarding datatype mapping, an optional URI-based `ValueType` field should be added to **AttributeDesignatorType** so that it is picked up by both `<AttributeDesignator>` (used in attribute queries) and `<Attribute>` (used in attribute statements):

```
<complexType name="AttributeDesignatorType">
    ...
    <attribute name="ValueType" type="anyURI" use="optional"/>
    ...
</complexType>
```

The default should be that the datatype is application-specific. A new section should be added to the Identifiers section of the core spec to define a URI standing for this semantic:

```
urn:oasis:names:tc:SAML:2.0:valuetype-format:appSpecific
```

Any `ValueType` setting (default or explicit) in a query needs to be exactly matched (in addition to other exact matches, as already defined in the core spec) in order for an attribute to be returned.

> **Note:** This field needs to appear on **AttributeDesignatorType** rather than **AttributeType** in order to satisfy goal 2.2. However, in the TC meeting [F2FMinutes], some people expressed concern about complicating queries with "expected datatype" information. The exact-match proposal above is the simplest possible way to meet goal 2.2, if we indeed decide that we do want to meet it.

Finally, the spec should add the following wording relating URI-based datatypes and XSD datatypes (using the `xsi:type` mechanism) appearing on attribute values:

> *If a datatype is specified on `<AttributeValue>` with xsi:type, it SHOULD be compatible with the `ValueType`.*

> **Note:** As stated, this is not testable in any way and therefore is not worth making a MUST. If we want to strengthen this, we will have to add wording about how to map XSD types to URI references. The obvious way to do this while avoiding an immense amount of overhead is to require the URI reference to dereference to an XSD schema document and have a fragment identifier with a shorthand XPointer that supplies the ID of the `<xs:complexType>` or `<xs:simpleType>` element in that schema document.

3.2 Clarifying Naming and Adding Metadata

To satisfy goal 2.1 regarding mapping of attribute names and goal 2.3, the `AttributeNamespace` field should be renamed to `NameFormat` (and the `AttributeName` field should be renamed to `Name` to follow suit). The schema change is as follows:

```
<complexType name="AttributeDesignatorType">
    <attribute name="Name" type="string" use="required"/>
    <attribute name="AttributeFormat" type="anyURI" use="required"/>
</complexType>
```

The `AttributeFormat` field should be defined in the spec as:

> *A URI reference representing the classification of the attribute name for purposes of interpreting the name. See Section X.X for some URI references that MAY be used as the value of the `NameFormat` attribute and their associated descriptions and processing rules. If no `NameFormat` value is provided, the identifier `urn:oasis:names:tc:SAML:2.0:att-format:unspecified` (see Section X.X.X) is in effect.*

This choice of field names provides brevity, and also consistency with the rest of SAML when it comes to "format" fields. In addition, a new subsection of the core spec's Identifiers section should define the following URI-based name formats:

> *urn:oasis:names:tc:SAML:2.0:att-format:unspecified*
> *The interpretation of the attribute name is left to individual implementations.*

> *urn:oasis:names:tc:SAML:2.0:att-format:x500*
> *The attribute name follows the convention for X.500/LDAP attribute naming [BIBREF]: urn:oid:<OID-as-string>*

> *urn:oasis:names:tc:SAML:2.0:att-format:uuid*
> *The attribute name follows the convention for Windows GUID/UUID identifier naming [BIBREF]: urn:guid:<GUID-as-string>*

> *urn:oasis:names:tc:SAML:2.0:att-format:uri*
> *The attribute name follows the convention for URI references [BIBREF], for example as used in XACML [BIBREF] attribute identifiers. The interpretation of the URI content or naming scheme is application-specific.*

Note: Just checking: Are the URN namespaces `oid:` and `uuid:` indeed registered?

A new optional `Source` field should also be added to **AttributeType**. The schema change is as follows:

```
<complexType name="AttributeType">
    <complexContent>
        <extension base="saml:AttributeDesignatorType">
            ...
            <attribute name="Source" type="string" use="optional"/>
            ...
        </extension>
    </complexContent>
```

The field should be defined in the spec as:

> *The source location or database from which the attribute came. Interpretation of the source information is application-specific.*

Finally, an `<xs:anyAttribute>` wildcard should also be added to **AttributeType**, to allow the arbitrary addition of global XML attributes onto the `<Attribute>` element. The schema change is as follows:

```
<complexType name="AttributeType">
    <complexContent>
        <extension base="saml:AttributeDesignatorType">
            ...
            <anyAttribute/>
        </extension>
    </complexContent>
</complexType>
```

This will permit the addition of various kinds of scope data and other context necessary to interpret the attribute value, without prematurely forcing all SAML users to use a long list of predefined fields that may not meet their needs.

3.3 Handling Null-Valued and Multi-Valued Attributes

The SAML V2.0 core spec, rev 05 [SAMLCore2.0], already includes wording that addresses goal 1.4:

> *<AttributeValue> [Any number]: The value of the attribute. If an attribute contains more than one discrete value, it is RECOMMENDED that each value appear in its own <AttributeValue> element. If the attribute exists but has no value, then the <AttributeValue> element MUST be omitted.*

The `ValueType` field proposed in a previous section applies to all the `<AttributeValue>` elements in an attribute statement. However, it's possible for each of these elements to have a different `xsi:type`

field specifying a different XSD datatype. To attempt to satisfy the part of goal 2.2 that refers to type consistency among multiple values of an attribute, the following wording should be added:

> *If more than one* `<AttributeValue>` *element is supplied for an attribute, and any of the elements have a datatype assigned through* `xsi:type`*, then all of the* `<AttributeValue>` *elements must have the identical datatype assigned.*

> **Note:** We could decide instead to use SHOULD here, or not to say anything on this point. But without this strong statement, saying anything about matching XSD to URI-based datatypes (as suggested in a section above) gets a bit more complicated.

3.4 Keeping the Changes Simple

In keeping with goal 2.5, the changes proposed are structurally not very invasive. Following is a summary of the proposed schema changes in previous sections:

```
<element name="AttributeDesignator" type="saml:AttributeDesignatorType"/>
<complexType name="AttributeDesignatorType">
    <attribute name="Name" type="string" use="required"/>
    <attribute name="NameFormat" type="anyURI" use="required"/>
    <attribute name="ValueType" type="anyURI" use="optional"/>
</complexType>
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
    <complexContent>
        <extension base="saml:AttributeDesignatorType">
            <sequence>
                <element ref="saml:AttributeValue" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <attribute name="Source" type="string" use="optional"/>
            <anyAttribute/>
        </extension>
    </complexContent>
</complexType>
<element name="AttributeValue" type="anyType"/>
```

This results in instances like this in a query (where `ValueType` is optional):

```
<AttributeDesignator
  Name="any-name-here"
  NameFormat="URI-indicating-how-to-interpret-name"
  ValueType="URI-indicating-desired-datatype-match"
/>
```

And in instances like these in an assertion sent in response (where `ValueType` and `Source` are optional):

```
<Attribute
  Name="any-name-here"
  NameFormat="URI-indicating-how-to-interpret-name"
  ValueType="URI-indicating-datatype"
  Source="source-location-or-database"/>
any-string-or-structured-value-here
</Attribute>
```

The changes do not affect the basic type hierarchy: **AttributeDesignatorType**>**AttributeType**. The new fields are optional (with carefully specified semantics for the case of their absence) in order to avoid adding new types and elements for the present vs. absent options.

4 References

[AttribRep] R. Lepro, “Attribute Representation in SAML v2.0”, proposal to OASIS SSTC, document identifier draft-sstc-attribute-02, 31 December 2003. http://www.oasis-open.org/committees/download.php/4884/draft-sstc-attribute-02.pdf.

[F2FMinutes] Minutes of the OASIS SSTC, 3-5 February 2004. http://lists.oasis-open.org/archives/security-services/200402/msg00123.html.

[MorganX500] R.L. “Bob” Morgan, “Conventions for Use of X.500/LDAP Attribute Types in SAML”, proposal to OASIS SSTC, document identifier draft-morgan-saml-attr-x500-00, 5 November 2003. http://www.oasis-open.org/committees/download.php/4124/draft-morgan-saml-attr-x500-00.pdf.

[SAMLCore2.0] E. Maler et al., Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, revision 05, document identifier sstc-saml-core-2.0-draft-05, 17 February 2004. http://www.oasis-open.org/committees/download.php/5519/sstc-saml-core-2.0-draft-05-diff.pdf.

A. Notices

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

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