Candidate Solution – Work Item 28b
Attribute Reconciliation between SAML and XACML

29 September 2003

Document identifier:
  sstc-saml-solution-wi28b-draft-01
  v. 0.1

Author Information:
  Rebekah Lepro, NASA Ames Research Center

Abstract:
This non-normative document describes a solution proposal for work item 28b from the SAML Scope and Work Items document, dated 9 September 2003 (document identifier: sstc-saml-scope-2.0-draft-06).
This document describes the use case, the current SAML components affected by the use case, a proposed solution and finally the motivation for the solution.
Introduction

Current Representation

Description

Version 1.1 of the SAML specification addresses the Attribute and AttributeDesignator elements in sections 2.4.4.1. Their use within the AttributeStatement and AttributeQuery constructs are outlined in sections 2.4.4 and 3.3.4 respectively. The relevant fragments of the SAML core and protocols are shown here:

AttributeDesignator:

```xml
<element name="AttributeDesignator"
type="saml:AttributeDesignatorType"/>
<complexType name="AttributeDesignatorType">
  <attribute name="AttributeName" type="string" use="required"/>
  <attribute name="AttributeNamespace" type="anyURI" use="required"/>
</complexType>
```

Attribute:

```xml
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
  <complexContent>
    <extension base="saml:AttributeDesignatorType">
      <sequence>
        <element ref="saml:AttributeValue" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

AttributeStatement:

```xml
<element name="AttributeStatement" type="saml:AttributeStatementType"/>
<complexType name="AttributeStatementType">
  <complexContent>
    <extension base="saml:SubjectStatementAbstractType">
      <sequence>
        <element ref="saml:Attribute" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Attribute Query:

```xml
<element name="AttributeQuery" type="samlp:AttributeQueryType"/>
<complexType name="AttributeQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <sequence>
        <element ref="saml:AttributeDesignator"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```
Rationale for current representation

Attribute statements were envisioned as the means for an authority to provide information about the characteristics of a particular subject within an assertion. The SAML TC strived for flexibility in attribute information, interoperability in conjunction with hooks for extension and preservation of a basic framework for attributes that contains only native SAML elements. In particular, the SAML TC wanted querying of attribute values to be SAML aware rather than just Xpath/Xquery aware. Thus, choices were made regarding particular facets of attribute representation in the standard. Details on the motivation for these choices are indicated below.

Attribute identifiers

The current attribute-naming scheme eliminates the need for uri parser to pull apart attribute designators. Further, the scheme maintains consistency with the selected action-naming scheme. Permit decision on acceptable authorities according to: attributes by authority or attributes by namespace. Although TC was willing to accept the inconsistency required by the use of QName idiom for SOAP-style status, it wanted to avoid the QName idiom in attribute-naming. Finally, the SAML TC chose to represent attributes as name/value pairs to maximize flexibility in attribute information using native SAML elements.

Attribute datatype

Currently SAML does not associate a datatype directly with an AttributeDesignator. The current SAML attribute approach assumes that any attribute type not directly defined by the xsd schema is defined in an external schema file shared out-of-band between the entities exchanging attribute information. The SAML TC presumed an existing need to know domain specific data to request attributes, thus supporting the selected schema exchange scenario. However, the typing of an attribute’s value preceded the provision for multiple values in a single attribute, thus permitting a situation where two attribute values for one attribute can contain differently typed data.

Attribute issuer

Currently SAML does not associate an issuer directly with an attribute. Rather, it specifies an issuer string that the RP can interpret to correctly identify the issuer that created the assertion containing the AttributeStatement. Notice that the issuer is associated directly with the assertion rather than the attribute. The SAML TC assumed a wide variety of RP would request attributes from a relatively small number of authorities issuing assertions. The issuer of an assertion should be unambiguous to the intended relying party so it may accept or reject the assertion as coming from a trusted authority.
As only intended audiences are assumed to interpret the value, those audiences were presumed to understand the data format of the issuer value via out of band agreement.

**Motivation**

**Abstract**

SAML places higher emphasis on standard methods to communicate attribute information than on any semantics of the communicated data. To support higher interoperability with XACML specified behavior and to reduce the burden on system developers that must define and interpret the semantics of attribute exchanges within the SAML framework, several modifications to the current SAML attribute definitions are proposed. These modifications seek to reduce reliance on schemas exchanged out of band between entities to communicate attribute meta-data by specifying mechanisms to include commonly used meta-data directly in the attributes exchanges via the framework. External schemas are still expected to exchange additional domain specific information not provided in these new attributes. Thus attribute data currently assumed interpreted by a schema negotiated and exchanged out of band between entities participating in the actual attribute communication. The TC assumed that these entities needed to know domain specific data to request attributes anyhow.

**XACML representation**

On several key concepts, the XACML defined handling for attribute metadata differs from the SAML approach. XACML processing can extract the information it deems necessary from various attribute formats and include that within a single attribute definition. This section identifies the motivation behind XACML choices, as they differ from SAML choices. The underlying motivation stems from XACML’s emphasis on enabling communicating parties to speak the same language within the actual communicated data. Further, this emphasis permits use of and reference to a set of standard and common functions in policies without requiring external schemas to communicate domain specific data.

**Attribute identifier**

XACML represents an attribute identifier in the AttributeDesignator construct with a single uri predicate. This uri is used for attribute matching during XACML processing. Further, this representation maintains consistency with LDAP processing model.

**Attribute datatype**

Each XACML AttributeDesignator carries a required attribute to indicate the attribute data type in uri form. By including data type information in a standard format, XACML allows for type checking in a standardly defined set of functions available to all policy writers. Further, mandatory data typing assures that all values consumed within a single function are consistently typed.
Attribute issuer

Each attribute designator can carry an optional attribute to represent the attribute issuer. This allows a policy to specify the issuer to use while matching attributes.

Interoperability between SAML and XACML

One critical use case of SAML attribute exchange is the provision of attributes to a policy evaluation process. To eliminate the need for attribute parsing, a policy can be represented directly in terms of attribute designators, such as the XACML policy representation. Therefore, we hope to create a mechanism to interoperate these standards at an attribute level.

Proposed Solution

The proposed solution affects only the AttributeDesignatorType definition. The Attribute, AttributeStatement and AttributeQuery constructs will inherit these modifications directly from the AttributeDesignator.

Schema Changes

AttributeDesignatorType:

```
<complexType name="AttributeDesignatorType">
  <attribute name="AttributeName" type="string" use="required"/>
  <attribute name="AttributeNamespace" type="anyURI" use="required"/>
  <attribute name="DataType" type="anyURI" use="optional"/>
  <attribute name="Issuer" type="string" use="optional"/>
  <attribute name="Format" type="anyURI" use="optional"/>
</complexType>
```

Affected facets

**AttributeName and AttributeNamespace**

An attribute identifier will still be represented by two distinct attributes. However, the semantics of their values will be modified. AttributeNamespace will indicate the taxonomy of attribute identifiers, such as uri, oid or uuid. Additional support for interoperability will be provided by the definition standard URIs for common taxonomies. Attribute Name will carry the full attribute identifier that is local to the specified taxonomy. For example, if this attribute will be referenced in an XACML policy, the value of AttributeNamespace will contain URI for the “uri taxonomy” and the value of AttributeName will be the complete attribute identifier expected within XACML request.

**Datatype**

An attribute containing a URI value will represent the datatype of a particular AttributeDesignator. The XACML specification already specifies a list of URIs for primitive datatype based on XML Schema types. Implementations are still free to define
additional URIs that specify a particular datatype. As some taxonomies, such as oid, imply the data type of any associated AttributeValue, the DataType attribute is optional.

**Issuer and Format**

Two additional attributes will be added to each AttributeDesignator construct. The Issuer attribute is a string that represents the attribute issuer. The Format attribute contains a URI that indicates how the value of the Issuer attribute should be represented.

**Motivation for modifications**

**AttributeName and AttributeNamespace**

This approach continues to defer the wildcarding issue to the XACML TC as a function of the policy distribution mechanism. Further, it eliminates questions regarding how to split a single URI identifier (as used in XACML) into a namespace and attribute name when requesting/populating attributes via SAML. Thus, we can easily identify the complete identifier to be completely used as the AttributeIdentifier of the XACML AttributeDesignator. Finally, using the namespace to identify the taxonomy of the attribute allows cleaner reuse of existing type schemes such as X500 attributes.

**DataType**

Current data-typing relies on the xsi:type mechanism within the AttributeValue element or inferred directly from attribute identifier. Associating an explicit type with an AttributeDesignator cleanly separates attribute identifier from data-typing information. Further, XACML requires data type support at attribute level for functions applied to attribute values. Finally, XACML functions such as mapping work on a particular type definition and convert the provided value to an appropriately mapped value rather than the type of value provided within the AttributeValue.

**Issuer and Format**

Including both an issuer and issuer format standardizes the interpretation of common types of communicated values to determine whether the issuer represents a trusted authority. This representation also reduces impact on system developers by providing standard identifiers for commonly used representations. Further, specification of the issuer format provides a clue regarding how to lookup an issuer-name. Associating an issuer (and issuer format) attribute with each attribute also provides a way to handle scoping of attributes. For example, it provides a means to differentiate between the standard attribute “faculty” issued by the Dept. of Computer Science at OSU vs. “faculty” at issued by the Dept. of Literature at OSU, both of which rely on a common OSU-wide attribute authority without requiring creation of two different attributes or attribute authorities.

**Compatibility with SAML 1.x**

As each newly defined attribute is optional, a SAML 1.x that does not carry the attribute should still be readily interpreted. The major source of incompatibility will lay in the altered semantics of the AttributeNamespace attribute. However, these semantics were
purposely open in SAML 1.x. Therefore, unless there is naming collision with the standard namespace URIs used as values for this attribute, the value can be treated as other information that must be negotiated out of band.