SAML 2.0 SSO Extension for Dynamically Choosing Attribute Values

Authors:

George Inman
University of Kent
g.inman@kent.ac.uk

David Chadwick
University of Kent
d.w.chadwick@kent.ac.uk

Status of This Document

This document provides information about a proposed extension to the SAML Core Profile [SAML-CORE]. Distribution is unlimited.

Abstract

This document provides an extension to the existing SAML 2.0 Core Specification [SAML-CORE] to enable a requestor to dynamically request specific attributes from an Identity Provider using a new type of extended authentication request message.

Table of Contents

1. Introduction

This document describes an new SAML 2.0 message type based upon the <samlp:AuthnRequest> message as specified in [SAML-CORE]. This new message type allows a Service Provider (SP) entity to dynamically choose the attributes requested from an Identity Provider at service provision time. It has become common practice for SP entities to combine two separate AuthnRequest and AttributeQuery messages into a single AuthnRequest message when Performing SSO using the Web Browser SSO Profile defined in Section 4.1 of [SAML-PROF]. Whilst it is possible to choose a pre-defined attribute set using the AttributeConsumingServiceIndex attribute of the existing request type, this approach loses much of the flexibility provided by the AttributeQuery message. In particular if trust negotiation is performed prior to the issuing of an AuthnRequest it is possible that the SP will negotiate different attribute sets for different users that it services. Using meta-data and the AttributeConsumingServiceIndex attribute could lead to an explosion in the number of meta-data items that need to be defined.

2. Notational Conventions

The key words ‘MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC 2119 [BRADNER1]

This specification uses schema documents conforming to W3C XML Schema [Schema1] and normative text to describe the syntax and semantics of XML-encoded SAML assertions and protocol messages. Note that in some cases the normative text of this specification imposes constraints beyond those indicated by the schema documents.
Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces (see Table 1) 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, defined in a schema [SAML-XSD].
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace, defined in a schema [SAMLP-XSD].
dcav:	urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser:dynamically-choosing-attribute-values	This is the dynamically choosing attribute values namespace defined in this document and its accompanying schema [DCAV-XSD].
xs:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown. For clarity, the prefix is generally shown in specification text when XML Schema-related constructs are mentioned.

3. Non Normative Description

The SAML Assertion Profile [SAML-CORE] provides a message type for authentication (<samlp:AuthnRequest>) and a separate message type for requesting authorisation attributes (<samlp:AttributeQuery>). In practice however it has become common for these two messages to be combined into a single <samlp:AuthnRequest> message performed at the beginning of an SSO session. As the message structure of the <samlp:AuthnRequest> was designed to accommodate only those features required for authentication, it becomes very difficult to configure the attributes required by the SP into the query message itself. In current implementations of the AuthnRequest the selection of the required attributes can only be performed using a numerical AttributeConsumingServiceIndex attribute which corresponds to a pre-defined metadata entry.

This model has several limitations the first being that it requires a static choice of authorisation attributes to be embedded in the SP's metadata. This means that either the SP will have a single metadata entry that requests all the user attributes that it could possibly need, or will have to define at least one metadata entry for each service it protects. The more ways that users can be authorised to access a service, the more metadata entries will be needed. If the SP chooses to request all available attributes it violates the principles of least privileges and of least attribute disclosure. The latter may lead to violation of data protection legislation. Defining multiple metadata entries may also present an administrative burden when a single SP supports many individual services. Furthermore it may lead to inconsistencies between the attributes specified in the PDP policies that protect the services and those specified in the metadata. If a PDP's policy changes then the metadata will also need to change in order to match the new policy. These problems are not insurmountable when each SP controls the dynamic distribution of its own metadata but in large scale federations with a central trusted third party metadata distributor, this will require the TTP to make the changes to the federation's metadata and then propagate these to the members. It may take several days before the changes are propagated to all members of the federation. Finally the static nature of the existing scheme makes the use of trust negotiation

mechanisms and complex access control policies impossible to use since the SP does not know before the start of the trust negotiation or the policy evaluation which attributes it will require. Yet it must ask for a fixed set of attributes at the time of authentication.

We conclude by noting that the dynamic choice of attributes must have been an original SAML requirement otherwise this facility would not exist in the <samlp:AttributeQuery> element. We also note that SPs in Information Cards can ask for alternative attributes that satisfy its policy. We therefore propose to add a similar facility to the <samlp:AuthnRequest>.

In order to support the dynamic choice of alternative sets of attributes we propose a new message type called an <dcav:AuthnAttributeRequest> message which is an extended version of the existing <samlp:AuthnRequest> message type. This message contains the addition of zero or more <dcav:AttributeSets> elements, each of which is used to describe a single attribute request policy. Use of the existing AttributeConsumingServiceIndex attribute is deprecated. This message type is intended to replace the AuthnRequest message as the principal SSO authentication request type for combined attribute and authentication requests.

4. <AuthnAttributeRequest> element

To request that an identity provider issue an assertion containing an authentication statement and a set of attribute statements, a presenter identifies itself to that identity provider and sends it an <AuthnAttributeRequest> message describing the properties and attributes that the resulting assertion needs to contain in order to satisfy its purpose. Among these properties may be information that relates to the content of the assertion and/or information that relates to how the resulting <samlp:Response> message should be delivered to the requester. The process of authentication of the presenter may take place before, during, or after the initial delivery of the <AuthnAttributeRequest> message.

The requester might not be the same as the presenter of the request if, for example, the requester is a relying party that intends to use the resulting assertion to authenticate or authorize the requested subject so that the relying party can decide whether to provide a service.

The <AuthnAttributeRequest> message SHOULD be signed or otherwise authenticated and integrity protected by the protocol binding used to deliver the message.

This message has the complex type **AuthnAttributeRequestType**, which extends **samlp:AuthnRequestType** and adds the following element, which is optional in general, but may be required by specific profiles.

`<dcav:RequestedAttributes>` [One only]
A requestor specifies its attribute requirements (i.e. attribute policy) in a <RequestedAttributes> element that comprises a set of attribute filters in either conjunctive normal form (CNF) or disjunctive normal form (DNF).

The CNF comprises sets of attributes (<One-Of> elements) and one attribute from each set must be returned, unless the set (One-Of element) is marked as optional, in which case it is not required to return any attribute from this set. If the SAML authority has more than one attribute that matches the attributes in an attribute set then it should return the first attribute that matches. If the SAML authority cannot supply a complete set of requested attributes, i.e. one attribute from each non-optional attribute set, then the SAML authority should not return any attributes and should return an error message "unable to supply requested attributes".

The DNF comprises alternative sets of attributes. Within each set of attributes some are mandatory (the <All-Of > element) and some are optional (the <Any-Of> element). Each set should be evaluated disjunctively in turn until one set is matched. It is not required to match and return any of the attribute of the <Any-Of> element. However, the SAML authority MUST have attributes that match each of the attributes in the <All-Of> element in order to meet the requestor's requirements. If the SAML authority does not have a complete set of attributes which meet those of the first set, then it should proceed to the next set to see if it can satisfy this. If the SAML authority has attributes which match multiple attribute sets then it should use those that match the earliest set in the sequence. If the SAML authority does not have a set of attributes that matches any of the attribute sets then the SAML authority MUST NOT return any attributes and it should return an error message "unable to supply requested attributes".

If no <RequestedAttributes> element is specified, it indicates that all attributes allowed by the SAML authority's policy are requested (unless a conflicting AttributeConsumerServiceIndex is defined – although the use of this is deprecated).

The <All-Of> and <Any-Of> elements are of the <AttrSetType> complex type which has the following components:

`<saml:Attribute>` [One or more]
This specifies a set of attributes that should be matched and returned by the SAML authority. If the <saml:Attribute> element is specified in an <All-Of> element then it MUST be returned. If it is specified in an <Any-Of> element then it may be returned. If a given <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then all the specified values should be returned in the response. The response MUST NOT contain any values that are not equal to the values specified in the query. If no value is specified in the query then all values allowed by the SAML authority's policy should be returned. In the absence of equality rules specified by particular profiles or attributes, equality is defined as an identical XML representation of the value. For more information on <saml:Attribute>, see [SAML-CORE] Section 2.7.3.1.

The <One-Of> element is of the <AttrSetOneOfType> complex type which has the following components:

`<saml:Attribute>` [One or more]

This specifies a set of one or more attributes from which one attribute MUST be matched and returned, unless the Optional Boolean is set to True, in which case no attributes may be matched and returned. If a given <saml:Attribute> element contains one or more <saml:AttributeValue> elements, then all the specified values should match and be returned in the response. The response MUST NOT contain any values that are not equal to the values specified in the query. If no value is specified in the query, and the attribute matches, then all values allowed by the SAML authority's policy should be returned. In the absence of equality rules specified by particular profiles or attributes, equality is defined as an identical XML representation of the value. For more information on <saml:Attribute>, see [SAML-CORE] Section 2.7.3.1.

`Optional` [Optional]

This Boolean attribute, if set to TRUE, indicates that the SAML authority need not match and provide any of the attributes and values specified in this <One-Of> set of attributes. The default value is FALSE.

A single <One-Of>, <Any-Of> or <All-Of> element MUST NOT contain two <saml:Attribute> elements with the same Name and NameFormat values (that is, a given attribute MUST be named only once in an attribute set).

If a <dcav:RequestedAttributes> element is present in the request then the AttributeConsumerIndex attribute SHOULD NOT be present. If it is present it MUST be ignored by the recipient in favour of the <dcav:RequestedAttributes> element contained in the request. If however no <dcav:RequestedAttributes> element is present in the request, then the AttributeConsumerIndex attribute MAY be used to define the set of attributes to return.

Please see Section 3.4.1.4 of [SAML-CORE] for general processing rules regarding this message. The additional rules given below are taken from Section 3.3.2.3 of [SAML-CORE] but constrained for this message type.

In response to a successful <AuthnAttributeRequest> message, a SAML authority returns an SSO assertion with additional attribute statements as follows:

- If the <RequestedAttributes> element is present in the request, it constrains/filters the attributes and optionally the values returned, as noted above.

- The attributes and values returned MAY also be constrained by application-specific policy considerations.

The second-level status codes urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile and urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue MAY be used to indicate problems with the interpretation of attribute or value information in a request.

The following schema fragment defines the <AuthnAttributeRequest> element and its **AuthnAttributeRequestType c**omplex type:

```
<element name="AuthnAttributeRequest" type="dcav:AuthnAttributeRequestType"/>
    <complexType name="AuthnAttributeRequestType">
        <complexContent>
            <extension base="samlp:AuthnRequestType">
                <sequence>
                     <element ref="dcav:RequestedAttributes"
                      minOccurs="0" maxOccurs="1"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

The following schema fragment defines the <dcav:RequestedAttributes> element and its **RequestedAttributesType** complex type:

```
<xs:element name="RequestedAttributes" type="dcav:RequestedAttributesType"/>
     <xs:complexType name="RequestedAttributesType">
        <xs:choice>
          <xs:element ref="dcav:CNF" minOccurs="0" maxOccurs="1"/>
          <xs:element ref="dcav:DNF" minOccurs="0" maxOccurs="1"/>
        </xs:choice>
    </xs:complexType>
```

RequestedAttributes is a choice between a conjunctive normal form of attribute filters or a disjunctive normal form of attribute filters.

The following schema fragment defines the <dcav:CNF> element and its **CNFType c**omplex type:

```
<xs:element name="CNF" type="dcav:CNFType"/>
     <xs:complexType name="CNFType">
        <xs:sequence>
           <xs:element ref="dcav:One-Of"
               minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

The conjunctive normal form comprises different sets of attributes, and one attribute MUST be chosen from each set, unless the set is marked as optional. The following schema fragment defines the <dcav:One-Of> element and its **AttSetOneOfType c**omplex type:

```
<xs:element name="One-Of" type="dcav:AttSetOneOfType"/>
   <xs:complexType name ="AttSetOneOfType">
        <xs:sequence>
            <xs:element ref="saml:Attribute"
            minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Optional" type="xs:boolean"
use="optional" default="false"/>
    </xs:complexType>
```

The following schema fragment defines the <dcav:DNF> element and its **DNFType c**omplex type:

```
<xs:element name="DNF" type="dcav:DNFType"/>
     <xs:complexType name="DNFType">
        <xs:sequence>
           <xs:sequence>
           <xs:element ref="dcav:All-Of"
               minOccurs="1" maxOccurs="unbounded"/>
           <xs:element ref="dcav:Any-Of"
               minOccurs="0" maxOccurs="unbounded"/>
           </xs:sequence>
        </xs:sequence>
    </xs:complexType>
```

The disjunctive normal form comprises different sets of sets of attributes, some of which MUST be present (the All-Of sets) and some which may be present (the Any-Of sets).

The following schema fragment defines the <dcav:All-Of> and <dcav:Any-Of> elements, and their associated **AttrSetType c**omplex type:

```
<xs:element name="All-Of" type="dcav:AttSetType"/>
<xs:element name="Any-Of" type="dcav:AttSetType"/>
        <xs:complexType name="AttSetType">
            <xs:sequence>
                <xs:element ref="saml:Attribute"
                    minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
```

5. Example SAML 2.0 SSO message flow using the <AuthnAttributeRequest> message element

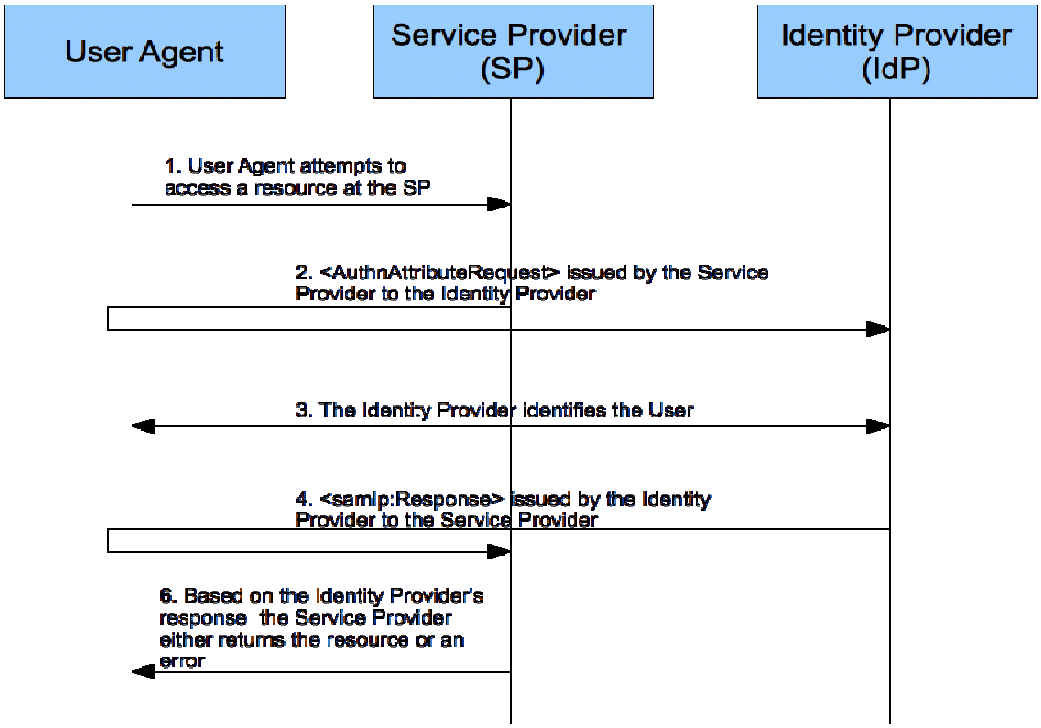

Figure 1.

Figure 1 provides a basic example of how SSO can be achieved using an AuthnAttributeRequest message. Please note that this profile is almost identical to the existing SSO profile defined in Section 4 of [SAML-PROF]. The only change to the protocol flow constitutes the replacement of the original <samlp:AuthnRequest> with the <AuthnAttributeRequest> message defined in this profile. Unless specifically mentioned below the processing rules defined in Section 4 of [SAML-PROF] should be applied to this profile.

1. HTTP Request to Service Provider

In step 1, the principal, via an HTTP User Agent, makes an HTTP request for a secured resource at the service provider without a security context. At this stage the service provider obtains the location of an endpoint at an identity provider for the authentication request protocol that supports its preferred binding. The means by which this is accomplished is out of the scope of this specification.

2. <AuthnAttributeRequest> issued by Service Provider to Identity Provider

In step 2, the service provider issues an <AuthnAttributeRequest> message to be delivered by the user agent to the identity provider. Each attribute required to authorise the user is included in the <dcav:RequestedAttributes> element of the request. Either the HTTP Redirect, HTTP POST, or HTTP Artifact binding can be used to transfer the message to the identity provider through the user agent.

3. Identity Provider identifies Principal

In step 3, the principal is identified by the identity provider by some means outside

the scope of this profile. This may require a new act of authentication, or it may reuse an existing authenticated session.

4. Identity Provider issues <Response> to Service Provider

In step 4, the identity provider issues a <Response> message to be delivered by the user agent to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer the message to the service provider through the user agent. The message may indicate an error, or will include (at least) an authentication assertion. The HTTP Redirect binding MUST NOT be used, as the response will typically exceed the URL length permitted by most user agents.

5. Service Provider grants or denies access to Principal

In step 5, having received the response from the identity provider, the service provider can respond to the principal's user agent with its own error, or can establish its own security context for the principal and return the requested resource. Note that an identity provider can initiate this profile at step 5 and issue a <Response> message to a service provider without the preceding steps.

Please note that more information on the Steps 1,3,4 and 5 can be found in [SAML-PROF] Sections 4.1.3. It should also be noted that as the <AuthnAttributeRequest> message is an extension of the <samlp:AuthnRequest> message all associated profile based processing rules defined for <samlp:AuthnRequest> messages are inherited by the <AuthnAttributeRequest> message type.

In Step 2 the <AuthnAttributeRequest> should be constructed according to the same profiles as a <samlp:AuthnRequest> message is but should contain a <dcav:RequestedAttributes> element to identify the attributes required by the SP. In the example request below the Service Provider requests that the response should contain a single givenName attribute with a value of either George or David.

```
<dcav:AuthnAttributeRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:dcav=" urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser:dynamically-choosing-
attribute-values" ID="Request1" Version="2.0"  IssueInstant="2007-11-26T07:35:00Z "
ForceAuthn=”true” AssertionConsumerServiceURL=”sp.kent.ac.uk”  ProviderName=”Kent SP” >
<saml:NameIDPolicy Format=”urn:oasis:names:tc:SAML:2.0:nameid-format:transient”
SPNameQualifier=”sp.kent.ac.uk” AllowCreate=”true” />
<saml:Conditions NotBefore=“"2009-11-26T07:35:00Z"  NotOnOrAfter="2007-11-
26T07:40:00Z"/>
  <dcav:RequestedAttributes>
   <dcav:CNF>
    <dcav:One-Of>
    <saml:Attribute AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"
             AttributeName="urn:mace:dir:attribute-def:givenName">
                   <saml:AttributeValue
                   xsi:type="xsd:string">George</saml:AttributeValue>
     </saml:Attribute>
    <saml:Attribute AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"
             AttributeName="urn:mace:dir:attribute-def:givenName">
                   <saml:AttributeValue
                   xsi:type="xsd:string">David</saml:AttributeValue>
    </saml:Attribute>
   </dcav:One-Of>
   </dcav:CNF>
  </dcav:RequestedAttributes>
</dcav:AuthnAttributeRequest>
```

In Step 4 the Identity Provider has authenticated the user and constructs a response to its initial request message. If the IdP is capable of providing a givenName attribute

with either of the matching values then it should include an <saml:AttributeStatement> element in the response containing that attribute value. It SHOULD NOT however return any attributes or values that were not specifically requested in the response. In the example below the IdP has provided a givenName value of George.

```
        <samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        ID="Response1 "
        InResponseTo="Request1" Version="2.0"
        IssueInstant="2007-11-26T07:35:00Z" Destination="http://sp.kent.ac.uk/"
        Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained”>

        <saml2: Assertion Version="2.0" IssueInstant="2009-11-26T07:35:00Z" ID="authn1">
                <saml2:Issuer>https://idp1.kent.ac.uk</saml2:Issuer>
                <ds:Signature>...</ds:Signature>
                <saml2:Subject>
                        <saml2:NameID>
                                randomid
                        </saml2:NameID>
                </saml2:Subject>
                <saml2:AuthnStatement AuthnInstant=”2009-11-26T07:35:00Z” SessionNotOnOrAfter
                =”2009-11-26T07:40:00Z” >
                        <saml2:AuthnContext>
                                <AuthnContextClassRef>
                                urn:mace:dir:constant:nist-sp-800-63:1
                                </AuthnContextClassRef>
                        </saml2:AuthnContext>
                        <saml2:SubjectLocality  Address=”129.12.16.129”
                        DNSName=”https://idp1.kent.ac.uk” />
                </saml2:AuthnStatement>
                <AttributeStatement>
                       <saml:Attribute
    AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"
                        AttributeName="urn:mace:dir:attribute-def:givenName">
                                <saml:AttributeValue xsi:type="xsd:string">
                                 George
                                </saml:AttributeValue>
                       </saml:Attribute>
                </AttributeStatement>
                <saml2:Conditions NotOnOrAfter="2007-11-26T07:40:00Z ">
                        <saml2:AudienceRestriction>
                                <saml2:Audience>sp.kent.ac.uk</saml:Audience>
                        </saml2:AudienceRestriction>
                </saml2:Conditions>
        </saml2: Assertion>
<samlp:Response>
```

6. References

[SAML-CORE] OASIS. “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, 15 March 2005

[SAML-PROF] OASIS “Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0”, 15 March 2005

[BRADNER] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt.

[Schema1] H. S. Thompson et al. XML Schema Part 1: Structures. World Wide Web Consortium Recommendation, May 2001. http://www.w3.org/TR/xmlschema-1/. Note that this specification normatively references [Schema2], listed below.

[Schema 2] Paul V. Biron, Ashok Malhotra. XML Schema Part 2: Datatypes. World Wide Web Consortium Recommendation, May 2001. See http://www.w3.org/TR/xmlschema-2/.

[SAML-XSD] S. Cantor et al., SAML assertions schema. OASIS SSTC, March 2005. Document ID saml-schema-assertion-2.0. See http://www.oasis-open.org/committees/security/

[SAMLP-XSD] S. Cantor et al. SAML protocols schema. OASIS SSTC, March 2005. Document ID saml-schema-protocol-2.0. See http://www.oasis-open.org/committees/security/.

[DCAV-XSD] G.Inman. AuthnAttributeRequest Schema, January 2010. Document ID saml-combined-auth-att-Request.xsd see http://sec.cs.kent.ac.uk/permis/documents/schema/saml-combined-auth-att-Request.xsd

Appendix 1: XSD Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<xs:schema

targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser:dynamically-
choosing-attribute-values"
    xmlns="urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser:dynamically-
choosing-attribute-values"
    xmlns:dcav="urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser:dynamically-
choosing-attribute-values"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:kent="urn:kent:SAML:2.0:AuthnAttribute"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="http://docs.oasis-open.org/security/saml/v2.0/saml-
schema-assertion-2.0.xsd"/>
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
        schemaLocation="http://docs.oasis-open.org/security/saml/v2.0/saml-
schema-protocol-2.0.xsd"/>
    <xs:annotation>
        <xs:documentation>
            Document identifier: saml-combined-auth-att-Request.xsd
            Location: http://sec.cs.kent.ac.uk/permis/documents/saml-combined-
auth-att-Request.xsd
            Revision history:
            V0.4 (March, 2012):
            Updated Draft Schema.
        </xs:documentation>
    </xs:annotation>

    <xs:element name="AuthnAttributeRequest"
type="dcav:AuthnAttributeRequestType"/>
    <xs:complexType name="AuthnAttributeRequestType">
```

```
        <xs:complexContent>
            <xs:extension base="samlp:AuthnRequestType">
                <xs:sequence>
                    <xs:element ref="dcav:RequestedAttributes"
                        minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

   <xs:element name=”RequestedAttributes” type=”dcav:RequestedAttributesType”
     <xs:complexType name=”RequestedAttributesType”>
        <xs:choice>
          <xs:element ref="dcav:CNF" minOccurs="0" maxOccurs="1"/>
          <xs:element ref="dcav:DNF" minOccurs="0" maxOccurs="1"/>
        </xs:choice>
    </xs:complexType>

   <xs:element name="CNF" type="dcav:CNFType"/>
     <xs:complexType name="CNFType">
        <xs:sequence>
           <xs:element ref="dcav:One-Of"
               minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

<xs:element name="DNF" type="dcav:DNFType"/>
     <xs:complexType name="DNFType">
        <xs:sequence>
           <xs:sequence>
           <xs:element ref="dcav:All-Of"
               minOccurs="1" maxOccurs="unbounded"/>
           <xs:element ref="dcav:Any-Of"
               minOccurs="0" maxOccurs="unbounded"/>
           </xs:sequence>
        </xs:sequence>
    </xs:complexType>

  <xs:element name="One-Of" type="dcav:AttSetOneOfType"/>
   <xs:complexType name ="AttSetOneOfType">
        <xs:sequence>
            <xs:element ref="saml:Attribute"
            minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Optional" type="xs:boolean" use="optional"
default="false"/>
    </xs:complexType>

<xs:element name="All-Of" type="dcav:AttSetType"/>
<xs:element name="Any-Of" type="dcav:AttSetType"/>
        <xs:complexType name="AttSetType">
            <xs:sequence>
                <xs:element ref="saml:Attribute"
                    minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>

</xs:schema>
```