David W Chadwick, Linying Su, Romain Laborde,
University of Kent
13 November 2009

Use of XACML Request Context to Obtain an Authorisation Decision

Abstract

The purpose of this document is to specify a protocol for accessing a Policy Decision Point (PDP) by a Grid Policy Enforcement Point (PEP) in order to obtain access control decisions containing obligations. The protocol is a profile of the SAML2.0 profile of XACMLv2 request/response contexts, tailored especially for grid use.

Contents

1. Introduction

This document describes how an XACMLv2 request context can be created and transferred by a Grid Policy Enforcement Point (PEP) to a Police Decision Point (PDP) in order to obtain authorisation decisions (possibly including obligations) for Grid applications. The XACMLv2 request context contains attributes of the subject, resource, action and environment, and is transported to the PDP in a SAMLv2 request message. The XACML response context contains an authorization decision and optional obligations that must be enforced by the PEP, either before, with or after enforcement of the user's request.

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 [BRADNER]

3. Model and Definitions

The authorization architecture is described in [ARCH]. Figures 1 and 2 are simplified versions of the figures in [ARCH] and they show in bold arrows the protocol that is being standardized in this document.

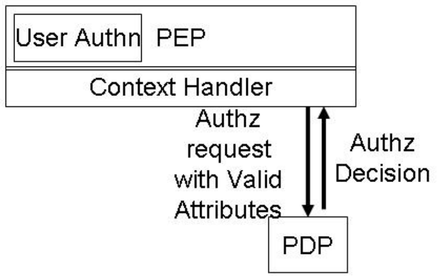

Fig 1 PEP Context Handler – Push Valid Attributes to PDP

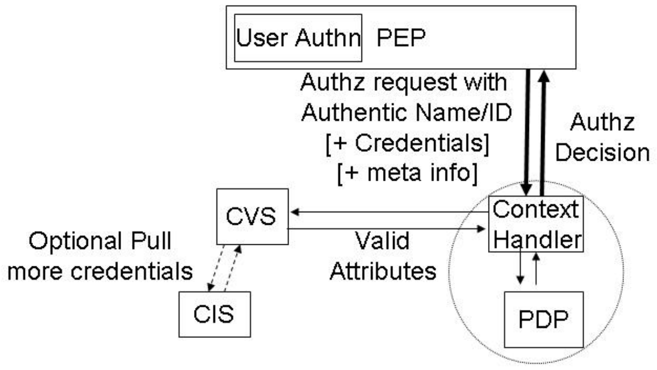
User Authn
PEP
Authz request with Authentic Name/ID [+ Credentials] [+ meta info]
Authz Decision
CVS
Context Handler
Valid Attributes
Optional Pull more credentials
CIS
PDP

Fig 2 PDP Context Handler – Obtain Valid Attributes from CVS

An attribute is a property of an entity. In the context of this document we are only concerned with attributes that are used for authorisation.

Attribute Authority (AA) is an entity (the issuer) that asserts attributes about another entity (the subject or holder).

Attribute Assertion is a statement made by an AA that a subject possesses a particular set of attributes.

Authorisation Credential (subsequently abbreviated to credential in this document) is an attribute assertion digitally signed by the issuer (i.e. it is a security token) so that it can be cryptographically validated by a Credential Validation Service.

The Policy Enforcement Point (PEP) is the component of the authorization service that intercepts the user's request and enforces the access control decisions that are made by the Policy Decision Point (PDP). Before the PDP can make an access control decision, it has to be given the validated attributes of the user. The PEP can undertake this validation action itself and pass the validated attributes to the PDP, as in Figure 1, or the PEP can pass a bag of unvalidated credentials to the PDP, as in Figure 2, and let the PDP work out which credentials are valid and which are not.

The Credential Issuing Service (CIS) – synonymous with the issuing service of Microsoft's Security Token Service – is an application independent service of an AA that issues authorisation credentials to requestors.

The Credential Validation Service (CVS) is the functional component that conceptually validates the user's credentials according to its configured Credential Validation Policy. The CVS returns the set of valid user attributes to its caller. The protocol for accessing the CVS is specified in [CVS] and will not be discussed further in this document.

The context handler is the functional component that is responsible for creating the request context to the PDP. The context handler is responsible for mapping the valid attributes returned from the CVS into the correct format for passing to the PDP. The context handler is also responsible for marshalling the attributes that describe the user's requested action and target resource, as well as any environmental attributes, and placing these into the authorization decision query. The creation of the request context is described in Section 4.

The PDP is responsible for creating the response context, which contains the authorization decision and optional obligations. The creation of the response context and the handling of obligations is described in Section 5.

The protocol for accessing the PDP is described in Sections 6 and 7.

4. XACMLv2 Request Context

The Authorisation Decision Query contains a data structure known as the XACMLv2 request context, defined in [XACMLv2] and [XAC-SCHEMA]. An XACMLv2 request context contains:

- the validated attributes of the user or a bag of unvalidated credentials (in the Subject element),
- the attributes of the Grid target resource(s) being accessed (in the Resource element),
- parameters of the user's access request (in the Action element), and
- any other attributes that may be needed (in the Environment element).

The XACMLv2 request context may also optionally contain the resource that is being accessed e.g. rows from a database, but this profile does not specify how a resource may be transferred to the PDP.

Attributes that the PEP fills into the various elements of the XACMLv2 request context can be divided into two categories: application independent attributes and application specific attributes. Application independent attributes are derived from the SOAP header of the service invocation that is under access control. Application specific attributes may be based on content from the SOAP body, or may be derived from application specific knowledge.

4.1 Resource Element

The Resource element MUST contain an Attribute element which has the AttributeId attribute with the value "urn:oasis:names:tc:xacml:2.0:resource:resource-id". This is obtained from the wsa:To element of the Soap header, viz:

Description	Address of the invoked service. Value of <wsa:To> element.
XACML request section	Resource
Attribute id	urn:oasis:names:tc:xacml:1.0:resource:resource-id
Value	Content of /soap:Envelope/soap:Header/wsa:To element
Data Type	http://www.w3.org/2001/XMLSchema#string

The Resource element MAY also contain other Attribute elements. Each Attribute element SHOULD contain at least one AttributeValue element. There MUST be an AttributeId and DataType attribute in each Resource Attribute element.

4.2 Subject Element

The subject element may contain either the user's validated attributes or a bag of unvalidated credentials.

4.2.1 Validated Attributes

If a CVS has been used to validate the user's credentials, then as specified in [CVS] the validated attributes are returned from the CVS in a single SAML attribute assertion, encoded in the XACML attribute profile format [SAMLPROF]. These attributes need to be placed into the Subject element of the XACMLv2 request context. How this mapping is performed is described in Section 2.1 of [XAC-SAML], and is repeated below for the convenience of the reader.

Description	Validated attributes of the subject
XACML request section	Subject
Attribute id	The fully-qualified value of the `<saml:Attribute>` `Name` XML attribute SHALL be used.
Value	The `<saml:AttributeValue>` value SHALL be used as the value of the `<xacmlcontext:AttributeValue>` element.
Data Type	The fully-qualified value of the `<saml:Attribute>` `DataType` XML attribute SHALL be used. If the `<saml:Attribute>` `DataType` XML attribute is missing, the XACML `DataType` XML attribute SHALL be `http://www.w3.org/2001/XMLSchema#string`.
Issuer	This field is optional. If present, the string value of the `<saml:Issuer>` element from the SAML Attribute Assertion SHALL be used.

Appendix 1 provides a non-normative set of example subject attributes

4.2.2 Unvalidated Credentials

If the PEP has not called the CVS to validate the credentials prior to calling the PDP, it may pass a bag of unvalidated credentials to the (context handler of the) PDP for validation prior to decision making. These credentials may be in a variety of formats e.g. X.509 public key certificate, X.509 attribute certificate, X.509 proxy certificate, VOMS attribute certificate embedded in a proxy certificate, Kerberos Ticket, Shibboleth attribute, proprietary credentials etc.

This profile defines a set of encodings for a variety of binary and other credentials, so that they can be passed to the PDP and recognized by the PDP before decoding and validation commences. The PDP may have its own built in CVS to validate the credentials, or it may use the services of a trusted web services CVS using the protocol specified in [CVS], in which case the credentials would be simply copied into the SAML assertion of that protocol.

Description	Attribute id	Value	Data Type
X.509 public key certificate of subject (which may be a proxy certificate)	http://www.ietf.org/rfc/rfc4523.txt#userCertificate	Base 64 encoding of the certificate	http://www.w3.org/2001/XMLSchema#base64Binary
X.509 attribute certificate of subject	urn:oid:2.5.4.58	Base 64 encoding of the certificate	http://www.w3.org/2001/XMLSchema#base64Binary
X.509 public key certificate of a CA	http://www.ietf.org/rfc/rfc4523.txt#cACertificate	Base 64 encoding of the certificate	http://www.w3.org/2001/XMLSchema#base64Binary
SAMLv1.0 Assertion	urn:oasis:names:tc:SAML:1.0:assertion	The SAML assertion in XML	http://www.w3.org/2001/XMLSchema#string
SAMLv1.1 Assertion	urn:oasis:names:tc:SAML:1.0:assertion	The SAML assertion in XML	http://www.w3.org/2001/XMLSchema#string
SAMLv2.0 Assertion	urn:oasis:names:tc:SAML:2.0:assertion	The SAML assertion in XML	http://www.w3.org/2001/XMLSchema#string

The identification of Kerberos tokens is specified in [Kerb].

4.3 Action Element

The action being requested by the user is derived from wsa:Action element of the SOAP header as follows:

Description	Value of <wsa:Action> element
XACML request section	Action
Attribute id	urn:oasis:names:tc:xacml:1.0:action:action-id
Value	content of /soap:Envelope/soap:Header/wsa:Action element
Type	http://www.w3.org/2001/XMLSchema#string

The Action element MAY also contain other Attribute elements, for example, parameters of the particular action. Each Attribute element SHOULD contain at least one AttributeValue element. There MUST be an AttributeId and DataType attribute in each Action Attribute element.

4.4 Environment Element

Each XACMLv2 request context contains an Environment element, containing zero or more environment Attribute elements. This document does not specify how the PEP obtains the environmental attributes, but it may, for example, use the system clock to obtain the time and date attributes.

4.4.1 Credential Push vs Pull Mode

When the CVS is a service called by the PDP (figure 2), the PEP may request the PDP's Context Handler to operate in credential push mode, credential pull mode, or both push and pull modes. Credential push mode is signaled by the presence of one or more credentials in the Subject element, as described in section 4.2.2. and InputContextOnly set to True as described in Section 6. Credential pull mode is signaled by the IDPList attribute in the Environment element as described below and InputContextOnly set to False or missing as described in Section 6. Push and pull mode is signaled by credentials in the Subject element, the IDPList attribute in the Environment element and InputContextOnly set to False or missing.

Description	Value of <wsa:Environment> element
XACML request section	Environment
Attribute id	http://schemas.ogf.org/ogsa-authz/2008/09/attribute/IDPList
Value	<IDPList> element
Type	IDPListType

The <IDPList> element is defined in Section 6 of [CVS], and it specifies a recommended subset of IDPs (i.e. CISs) that should be contacted. Its purpose is to advise the CVS where it may find attributes for the current subject when working in the *credential pull (or pullpush) mode* of operation, in order to prevent the CVS from contacting all its known set of trusted IdPs.

4.5 An Example XACML Request Context

The following is an example XACML request context for a student from My Org, who wishes to perform get access for 3 (GB) of MRAM (ID = 12345) on the 29th October 2005

```
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=”urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-
schema-os.xsd”>
 <Subject>
  <Attribute AttributeId="urn:oid: 1.2.826.0.1.3344810.1.1.14"
             DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>student</AttributeValue>
  </Attribute>
  <Attribute AttributeId="http://www.ieft.org/rfc/rfc2256.txt#o"
             DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>My Org</AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <Attribute AttributeId="://www.ieft.org/rfc/rfc2256.txt#objectClass"
             DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>MRAM</AttributeValue>
  </Attribute>
  <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
             DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>12345</AttributeValue>
  </Attribute>
 </Resource>
 <Action>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id"
           DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>get</AttributeValue>
  </Attribute>
  <Attribute
AttributeId="http://sec.cs.kent.ac.uk/GGF/XACML/MRAM.get.size"
           DataType="http://www.w3.org/2001/XMLSchema#integer">
      <AttributeValue>3</AttributeValue>
  </Attribute>
 </Action>
 <Environment>
  <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
           DataType="http://www.w3.org/2001/XMLSchema#date">
        <AttributeValue>2005-10-29</AttributeValue>
  </Attribute>
 </Environment>
</Request>
```

5. XACMLv2 Response Context

The data structure returned by an XACMLv2 conformant PDP is called an XACML Response Context. This contains one or more Result elements.

5.1 Result Element

Each Result element contains a Decision element, an optional Status element and an optional Obligations element.

A Result element MAY have a ResourceId attribute and MUST contain a Decision element. It MAY contain a Status Element and MAY contain an Obligation element.

The value of the ResourceId attribute, if present, MUST be obtained from the corresponding resource attribute in the XACMLv2 Request Context i.e. the attribute having the name “urn:oasis:names:tc:xacml:1.0:resource:resource-id”.

The Decision element MUST be set to either: Permit, Deny, Indeterminate or NotApplicable.

This profile does not require the Status Element to be present. The Status Element MAY be present, but its contents are not specified by this profile. The PEP MAY ignore the Status element.

The PEP MUST act on the returned obligations, if any. If the PEP is unable to fulfill any of the returned obligations then it MUST deny access to the subject. Obligations are defined in the next section.

5.2 Obligations Element

The Obligations element contains a set of zero, one or more Obligation Elements.

An Obligation element is defined as a set of attribute assignments that MUST be carried out by the PEP plus a set of attributes that direct the PEP what to do.

Each Obligation element MUST contain an ObligationID attribute and a FulfillOn attribute.

```
<xs:element name="Obligation" type="xacml:ObligationType"/>
     <xs:complexType name="ObligationType">
     <xs:sequence>
            <xs:element ref="xacml:AttributeAssignment" minOccurs="0”
             maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
     <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

5.2.1 FulFillOn Attribute

The FulfillOn attribute MUST take the value of Permit or Deny, and MUST be set to the same value as the Decision element. This means that Obligations cannot be returned for Indeterminate and NotApplicable decisions. The PEP MUST act on any returned obligations. If the PEP is unable to fulfill any of the returned obligations then it MUST deny access to the subject.

5.2.1 Obligation ID

The obligation ID is a directive to the PEP, informing it what type of obligation this is and how to process it. Some example obligation IDs are given in Appendix 2.

5.3 An example XACML Response Context

The following example XACML Response Context permits the student from My Org to access the MRAM, but places an obligation on the PEP to update the coordination database with the amount of memory being used and to add it to the account balance before access is granted.

```
<Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-open.org/xacml/xacml-core-2.0-context-schema-os.xsd">
<Result ResourceId="12345">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  <Obligations>
    <Obligation ObligationId="
http://www.ogf.org/authz/2007/08/oblig/coord/chronicle=Before”
     FulfillOn="Permit" >
    <AttributeAssignment AttributeId=
     "http://sec.cs.kent.ac.uk/GGF/XACML/environment/balance"
     DataType="http://www.w3.org/2001/XMLSchema#integer">
      <Apply FunctionId=
             "urn:oasis:names:tc:xacml:1.0:function:integer-add">
         <Apply FunctionId=
          "urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
           <ActionAttributeDesignator AttributeId=
           "http://sec.cs.kent.ac.uk/GGF/XACML/MRAM.get.size"
            DataType="http://www.w3.org/2001/XMLSchema#integer"/>
        </Apply>
        <Apply FunctionId=
          "urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
          <EnvironmentAttributeDesignator AttributeId=
          "http://sec.cs.kent.ac.uk/GGF/XACML/environment/balance"
          DataType="http://www.w3.org/2001/XMLSchema#integer"/>
        </Apply>
      </Apply>
    </AttributeAssignment>
  </Obligation>
</Obligations>
</Result>
</Response>
```

6. XACML Authorization Decision Query and Statement

The SAML2.0 profile of XACMLv2.0 [XAC-SAML] specifies extensions to SAML2.0 to enable:

- an XACML request context and policy to be carried in a SAML request message to a PDP, as an XACMLAuthzDecisionQuery extension;
- an XACML request and response context to be carried in a SAML response message to the PEP, as an XACMLAuthzDecisionStatement extension.

This OGF profile uses Section 4 of the SAML2.0 profile of XACMLv2.0 [XAC-SAML] which specifies the XACMLAuthzDecisionQuery and the XACMLAuthzDecisionStatement.

Section 4.4 describes the XACMLAuthzDecisionQuery. Three Boolean attributes are defined: They MUST be used in this profile as follows

- InputContextOnly – should be set to True if the PDP is not to allow external credentials or attributes to be pulled and used in the authorization decision or False is additional ones can be used. Note that the default setting is False, which means that in the absence of this attribute, any associated CVS is allowed to pull additional credentials if it needs to, or the PDP is allowed to pull additional attributes from its PIP if it needs to. Note that the PEP MUST NOT set InputContextOnly to True and provide a value of the IDPList Environment attribute as described in section 4.1.1 above, as these conflict with each other
- ReturnContext – should be set to True if the PEP wants the PDP to return a Request Context along with the authorization decision. The PDP SHOULD only return those attributes that were used in the decision making, and omit those attributes that were not used in making the returned decision. If InputContextOnly is FALSE then the returned context MUST also contain any pulled attributes that were used in the decision making. The default value for this is False, meaning that a RequestContext MUST NOT be included along with the authorization decision.
- CombinePolicies – should be set to True if the PDP is to combine the policies provided in the RequestContext along with its local policies in making an authorization decision. If set to False, then the PDP MUST only use the policy or policy set in the RequestContext, and there MUST only be one of these present in the RequestContext.

The XACMLAuthzDecisionQuery may contain zero or more Policies, zero or more PolicySets and zero or more `ReferencedPolicies`, according to the wishes of the PEP. This is to allow the PEP to dynamically provide policies to the PDP for each authorization decision.

The `xacml-samlp:AdditionalAttributes element MUST NOT be set, as this only applies to the administration functionality of XACMLv3 PDPs.`

Section 4.1 describes the XACMLAuthzDecisionStatementType. This contains an XACML response context and optionally a request context.

Section 4.10 describes the XACMLAuthzDecision Response which is carried as a new type of SAML assertion, the `XACMLAuthzDecisionStatement. The following additional restriction is defined in this profile:` the <ds:Signature> element SHOULD be missing from the returned assertion. Validation of the assertion message, if required, is provided by TLS/SSL (see section 8).

7. Mapping to Lower Layer Protocols

The XACML request context is carried as an extended SAML Request message, which is itself carried as the body of a SOAP message, which is then carried over http or https (see section 8).

An example SOAP SAML XACML Request message is shown below (copied from [RSA-Interop])

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
    xmlns:soapenv ="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <xacml-samlp:XACMLAuthzDecisionQuery
        xmlns:xacml-samlp="urn:oasis:xacml:2.0:saml:protocol:schema:os"
        ID="e064bd912f83c1544fea110307000acf"
        IssueInstant="2007-05-21T22:00:36Z"
        Version="2.0">
      <xacml-context:Request
          xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os">
        <!-- See Section 4.5 for an example XACML request context element -->
      </xacml-context:Request>
    </xacml-samlp:XACMLAuthzDecisionQuery>
  </soapenv:Body>
</soapenv:Envelope>
```

The request message above contains 3 protocol layers:

1. soapenv: is the SOAP layer, comprising a soapenv: Envelope and Body.
2. xacml-samlp: is the enhanced SAML protocol layer containing the XACML extension to carry XACML authorization decision queries, described in [XAC-SAML] and [XAC-Errata].
3. xacml-context: is the XACMLv2 layer containing the request context, which is described in [XACMLv2].

The XACML response context is carried as an XACML authorization decision statement within a SAML assertion, which is itself carried in a SAML Response message, which is carried in the body of a SOAP message.

An example SOAP SAML XACML Response message is shown below (copied from [RSA-Interop])

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
  <soapenv:Body>
    <samlp:Response
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      ID="A12345602"
      Version="2.0"
      IssueInstant="2007-05-09T00:00:01Z">
      <samlp:Status>
        <samlp:StatusCode
          Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
      <saml:Assertion
          xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
          Version="2.0"
          ID="A12345603"
          IssueInstant="2007-05-09T00:00:01Z">
        <saml:Issuer>xacml.interop.com</saml:Issuer>
        <saml:Statement
            xmlns:xacml-saml="urn:oasis:xacml:2.0:saml:assertion:schema:os"
            xsi:type="xacml-saml:XACMLAuthzDecisionStatementType">
          <xacml-context:Response
              xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os">
            <!-- See Section 5.3 for an example response context element -->
          </xacml-context:Response>
        </saml:Statement>
      </saml:Assertion>
    </samlp:Response>
  </soapenv:Body>
</soapenv:Envelope>
```

The response message above contains 3 protocol layers:

1. soapenv: is the SOAP layer, comprising a soapenv: Envelope and Body.
2. samlp: is the SAML Protocol layer containing the standard SAML Response message. This contains a saml:Assertion, which in turn contains a saml:Statement. This contains the new XACMLAuthzDecisionStatementType described in [XAC-SAML] and [XAC-Errata].
3. xacml-context: is the XACMLv2 layer containing the response context, which is described in [XACMLv2].

8. Security Considerations

The PEP and PDP MUST perform mutual authentication of each other, unless a trusted channel is already established between them. Mutual authentication MUST be undertaken by transport layer security (TLS/SSL).

Message confidentiality SHOULD be assured between the PEP and the PDP, unless a trusted channel is already established between them. Message confidentiality SHOULD be undertaken by transport layer security (TLS/SSL).

```
Note that SAML does not provide a means for encrypting
(confidentially protecting) entire request messages, except via the
underlying transport layer security, although it does allow entire
assertions to be encrypted in the response. The latter however is
not sufficient to confidentially protect details about the subject
of the XACML request context.
```

9. Contributors

David W. Chadwick
The Computing Laboratory
University of Kent

D.W.Chadwick@kent.ac.uk

Linying Su
The Computing Laboratory
University of Kent
L.Su-97@kent.ac.uk

Romain Laborde
Institut de Recherche en Informatique de Toulouse (IRIT), Université Paul Sabatier, 118 Route de Narbonne, F-31062, TOULOUSE CEDEX 9, France
laborde@irit.fr

10. Acknowledgments

The authors would like to thank the UK JISC and the EC TrustCoM and TAS3 projects for jointly funding this work. The authors would also like to thank Erik Rissanen of SICS who produced a related profile for use in the EC TrustCoM project, and upon which some of this document is based.

11. Intellectual Property Statement

12. Disclaimer

13. Full Copyright Notice

14. References

[ARCH] David Chadwick. “Functional Components of Grid Service Provider Authorisation Service Middleware”, OGF GWD-I, 20 June 2009.
[CVS] David Chadwick, Linying Su. “Use of WS-TRUST and SAML to access a Credential Validation Service”. OGF GWD-R-P, 25 June 2009.
[HUGHES] Hughes J. and Maler E. Technical Overview of the OASIS Security Assertion Markup Language (SAML) V1.1, May 2004.
[BRADNER] Bradner, S. Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997.
[RFC3281] S. Farrell, R. Housley. “An Internet Attribute Certificate Profile for Authorization”. RFC 3281, April 2002.
[RSA-Interop] OASIS. “XACML 2.0 RSA 2008 Interop Scenarios Version 0.12, Working Draft” 15 April 2008
[SAML] OASIS. “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, 15 March 2005
[SAMLPROF] OASIS “Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, 15 March 2005
[VOMS] Vincenzo Ciaschini, Valerio Venturi, Andrea Ceccanti. “The VOMS Attribute Certificate Format”. OGF GWD-R-P, 25 June 2009
[XACMLv2] OASIS “eXtensible Access Control Markup Language (XACML) Version 2.0” OASIS Standard, 1 Feb 2005
[XAC-SAML] OASIS “SAML 2.0 profile of XACML v2.0” Committee Draft, 16 April 2009
[Kerb] OASIS “Web Services Security Kerberos Token Profile 1.1”. OASIS Standard Specification, 1 February 2006

Appendix 1. Example Subject Attributes

Non-normative

The following attribute IDs may be used with commonly used attributes.

Description	Attribute id	Value	Data Type
User’s DN from cert or proxy cert	urn:oasis:names:tc:xacml:1.0:subject:subject-id	LDAP string representation of the user’s DN	urn:oasis:names:tc:xacml:1.0:data-type:x500Name
VOMS Attribute	urn:oid: 1.3.6.1.4.1.8005.100.100.4	Value taken from the VOMS AC, comprising: <vo name>://<fqhn>:<port>,<group name>/Role=<role name>	http://www.ogf.org/authz/2007/08/attrDT/IetfAttrSyntax (See Note 1)
VOMS VO Name	urn:<to-be-defined>:attr:VOMSVOName	Value taken from VOMS attribute	`http://www.w3.org/2001/XMLSchema#string`
VOMS Group Name	urn:<to-be-defined>:attr :VOMSGroupName	Value taken from VOMS attribute	`http://www.w3.org/2001/XMLSchema#string`

VOMS Role Name	urn:<to-be-defined>: attr:VOMSRoleName	Value taken from VOMS attribute	`http://www.w3.org/2001/XMLSchema#string`
Permis Role	urn:oid: 1.2.826.0.1.3344810.1.1.14	Value taken from permisRole attribute	`http://www.w3.org/2001/XMLSchema#string`

Note1. leftAttrSyntax is specified in [RFC3281] and its use is described in [VOMS]

Appendix 2. Example Obligation Elements

Non-normative

The following types of obligation are given as examples

Description	Obligation ID
Send an email notification	`urn:oasis:names:tc:xacml:example:obligation:email`
Update the coordination database	http://www.ogf.org/authz/2007/08/oblig/coord/chronicle=[Before|With|After]
Run the grid job under this username	http://www.ogf.org/authz/2007/08/oblig/userAccount

A2.1 Send an email notification

This obligation ID may be used when the PDP wishes an email notification to be sent to an email address. The body of the obligation (attribute assignments) will contain assignements for message fields such as subject, from, to, date, and the body of the message.

A2.2 Update the coordination database

This obligation ID may be used when the PDP has an external coordination database that is used to store some aspect of the RetainedADI defined in the ISO Access Control Framework [ISO]. The RetainedADI is the history of prior access control decisions that is used to inform future access control decisions. The chronicle parameter specifies when the obligation SHOULD be enforced by the PEP, with respect to enforcing the user's access request. It can take one of three values: Before, With or After.

A2.3 Run the grid job under this username

This obligation ID may be used when the policy contains the user ID that a grid job should be run under. The body of the obligation contains one or more attribute assignments which set the userid to a particular value.