Extending XACML Authorisation Model to Support Policy Obligations Handling in Distributed Applications

Yuri Demchenko[#1], Cees de Laat[#2]
Oscar Koeroo[*3], Hakon Sagehaug[**4]

[1]demch@science.uva.nl, [2]delaat@science.uva.nl
[#]System and Network Engineering Group, University of Amsterdam
[3]okoeroo@nikhef.nl
[*]NIKHEF
[4]hakon.sagehaug@bccs.uib.no
[**]BCCS, UNIFOB AS

Abstract

The paper summarises the recent and on-going developments and discussions in the Grid security community to built interoperable and scalable AuthZ infrastructure for distributed applications. The paper provides a short overview of the XACML policy format and policy obligations definition in the XACML specification. The paper analyses the basic use cases for obligations in computer Grids and on-demand network resource provisioning abstracted to the general complex resource provisioning (CRP) model to identify major requirements and functionalities in obligations handling that further is proposed as a Reference Model for Obligations Handling (OHRM). The paper refers to ongoing implementations of the obligations interoperability and handling framework in such project as EU funded projects EGEE and Phosphorus. The proposed implementation is based on the adoption and extension of the OASIS SAML2.0 profile of XACML specification but defining a number of missing interface definitions and semantic conventions. The purpose of this paper is to facilitate wider discussion of the policy obligations concept based on the described ongoing implementations.

Keywords: Policy Obligations, Reference Model for Policy Obligations Handling, XACML, Complex Resource Provisioning, Generic AAA Authorisation Framework.

1. Introduction

Policy obligation is an important policy enforcement mechanism and a component of the policy based access control architecture. Policy obligations allow defining (mandatory) actions that must be taken in connection to the policy decision but can not be specified in the policy document, or may not be known to the policy administrator. This also allows for separating policy decision stage and policy enforcement stage that may require information unknown for the policy decision point at the time of making policy decision, e.g. the resource or service status, user account status, available pool account, etc..

Policy obligations is a part the XACML policy definition [1], however current XACML policy enforcement model doesn't define how policy obligations can be handled by authorisation system components.

The goal of this paper is to fill the gap between general concept and many implementation details that play significant role in building interoperable AuthZ infrastructure solutions.

This paper discusses typical use cases for policy obligations use in computer Grids and on-demand network resource provisioning abstracted to the general complex resource provisioning (CRP) model and proposes a Reference Model for Obligations Handling (OHRM) that allows for different obligations enforcement scenarios.

In the typical CRP scenarios, the initial policy decision about providing access or allocating resource to a requestor/user is done at the reservation stage but actual policy decision enforcement is done at the resource or service access stage.

As an example, consider a user requesting a dedicated network path/connectivity at specific time from the Grid enabled Network Resource Provisioning System (NRPS). The NRPS checks user credentials/attributes against access control policy, network resource availability and makes advance reservation. As a result of this the user can be assigned one of available pool accounts that should be used when accessing network resource and may be connected to a specific level or quality of service. The access control policy may contain a notion to map user identity or attributes to the pool account but it can not contain a specific account name which is selected from the available pool. Other actions in connection to the policy decision can be assigning user quota, requirements to log specific data or accounting. These kind of requirements is typically solved by using provisional policies or policy obligations.

The paper is organized as follows. Section 2 provides a short overview of the policy obligations definition in XACML. It important to note that this paper and the proposed solutions deal with the policy obligations as they are defined in XACML what is different from the obligation policies as they defined in the Ponder policy language [2].

Section 3 describes the general CRP model that separates resource reservation, resource deployment, and resource access or consumption stages. The section summarises common requirements to the authorisation (AuthZ) service infrastructure to support multidomain CRP and identifies the basic use cases and required functionality for policy obligation handling.

Section 4 introduces OHRM that extended XACML and generic authorisation models to support obligations handling in the distributed AuthZ infrastructure. Section 5 provides implementation suggestions for the OHRM as a component of the AuthZ infrastructure in Grid based applications.

The presented research is a result of authors' participation and contribution to wide international coordination activity in building interoper[ble Grid oriented AuthZ infrastructure, that will allow flexible relation between central AuthZ services and distributed policy enforcement infrastructure.

The presented research and proposed solutions are specifically oriented for using with the popular Grid middleware being developed in the framework of large international projects such as EGEE[1] and Globus Alliance[2].

2. Policy Obligations definition in XACML

A XACML policy is defined for the target tuple "Subject-Resource-Action" (S-R-A) which can also be completed with the Environment (S-R-A-E) component to add additional context to instant policy evaluation [1]. The XACML policy can also specify the Obligations as actions that must be taken on positive or negative authorisation decisions. Introducing policy obligations allows for more flexible policy definition by separating stateless conditions that are based on the information provided in the access control request and stateful conditions that may depend on the target system/resource state. This functionality is important for accounting in consumable resource provisioning or mapping requestor's identity to the resource pre-defined internal (pool) accounts, what is a common approach in computer Grids.

The XACML authorisation model corresponds to the X.812 Authorisation [3] and GAAA-AuthZ [4] models and includes the following major functional modules: Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Authority Point (PAP), and multifunctional Context Handler (CtxHandler) that support all necessary communications between PEP and PDP.

In the XACML authorisation model, a decision request sent in a Request message provides context for the policy-based decision. The policy applicable to a particular decision request may be composed of a number of individual rules or policies, each of which can contain obligations. XACML specifies a number of policy and rule combination algorithms. The Response message may contain multiple Result elements, which are related to individual Resources. The Result element may contain in the Obligations element a set of obligations returned by PDP extracted from applicable policies. It is stated that Obligations are returned by PDP "as is", i.e. in a form as they are written in the policy.

There are no standard definitions in XACML version 2.0 how the obligated actions should be processed. It should rely on the bilateral agreement between a resource manager/owner defining policies and the PEP that will enforce PDP's decision. The XACML specification requires that PEPs must deny access unless they understand and can enforce all obligations returned in the PDP Response message.

[1] http://www.eu-egee.org/
[2] http://www.globus.org/

3. Authorisation Infrastructure for Multidomain CRP

3.1. CRP in Computer Grids and on-demand Network Resource Provisioning

Two major use cases for the general CRP [5, 6] are computer Grids and on-demand Network Resource Provisioning. Although different in current implementations, they can be abstracted to the same CRP operational model when considering their implementation with the Grid or Web Services and using common Grid middleware. This abstraction is considered as an important aspect to provide a common access control infrastructure for distributed Grid-enabled resources that may include both computing resources and connecting them dedicated network infrastructure.

Current network resource provisioning models use simple pre-allocation/pre-scheduling. At the time of access, users just need to authenticate themselves and use prescheduled resource. Similar approach is used in current Grid applications. The Grid job manager provides all resources allocation and coordination for the user tasks/jobs, and runs these tasks according to schedule. Such model can be considered as provider centric and leaves less freedom to a user. This model obviously implies significant restrictions for such highly dynamic environment as Grid resource allocations and on-demand provisioning. If this kind of technologies to become widely used, more standardization efforts are needed to define the whole CRP process and authorisation service components functionality to support different types of advance reservations for consumable resource, such as [7]:

- fixed advance reservations that imply strict time/amount constraints (e.g., fixed time and network bandwidth);
- deferrable advance reservation that allows some degree of freedom in the time domain with fixed amount (or bandwidth);
- malleable advance reservation that allows variable duration and amount for the fixed consumption amount (e.g., transfer the necessary volume of data in the pre-defined time-frame with bandwidth that can be variable).

The typical on-demand resource provisioning includes three major stages: resource reservation, resource deployment, and the reserved resource access or consumption. In its own turn, the reservation stage includes the following basic steps: (1) resource lookup, (2) complex resource composition (including alternatives), (3) reservation of individual resources and their association with the global reservation ID/ticket. In multi-domain CRP, the reservation stage may require execution of complex procedures that may also request individual resources authorisation. This process can be driven by a meta-scheduling system and controlled by the provisioning/reservation workflow that should also handle domain related AuthZ policies.

Such specific usecase as multidomain CRP may require that resource reservation policy in each successive domain relies on the previous domain positive AuthZ decision or is conditional to executing some actions or obligations in other domains. First requirement can be satisfied by using XACML Environment element that contains AuthZ decision from the previous domain. Conditional (or provisional) AuthZ decision can be achieved by using XACML policy obligations supported by consistent policy obligations enforcement mechanisms. This functionality requires extended AuthZ context exchange between domains and can be achieved by using AuthZ ticket that was proposed in [8] and can communicate full AuthZ decision context in a secure way.

In the discussed multidomain CRP model, domains or sites (as associations of entities) are defined by common policy under single administration, common namespace and semantics, shared trust relations and authorities, etc. Access control to all domain or site related resources can be controlled by Domain or Site Central AuthZ Service (DCAS or SCAS) that can simplify common domain/site access control policies management and resource management in general. However for such complex resources as Optical Networks or Computer Grid clusters there is no possibility to define policy components dependent on the resources state or other event related to the user task execution. This problem can be solved by adding stateful policy obligations to the site-central stateless policies. Such functionality is available in the XACML but it needs to be supported by common obligations handling model and mechanisms, which however should be defined in a way independent of a policy format to ensure interoperability of different AuthZ frameworks that may be present in a distributed multidomain environment.

In the context of the distributed AuthZ architecture for CRP, obligations provide an important mechanism for policy decision enforcement in the provisioned network resources, in particular, obligations can be used for mapping global user ID/account to local accounts or groups, assigning quotas or usage limits, services/resources combination with implied conditions (e.g., computing and storage resources).

3.2. Authorisation service components to support multidomain CRP

To support multidomain CRP, the AuthZ infrastructure should provide the following functionality:

- AuthZ session management to support complex AuthZ decision and multiple resources access, including multiple resources belonging to different administrative and security domains.
- AuthZ tickets with extended functionality to support AuthZ session management, delegation and obligated policy decisions.
- Authorisation and reservation tokens as policy enforcement mechanisms.
- Policy obligations handling to support conditional (or provisional) policy decision that can be made by site/domain central AuthZ service.

Figure 1 illustrates the major GAAA-AuthZ components and how they interact when evaluating a service request [9, 6].

The authorisation service is called from the service/application interface via the AuthZ gateway (that can be just an interceptor process called from the service or application) that intercepts a service request ServiceRequset (ServiceId, AuthN, AuthZ) that contains a service name (and variables if necessary) and AuthN/AuthZ attributes/credentials. The AuthZ Gateway extracts necessary information and sends an AuthZ request AuthzRequest (ServiceId, Subject, Action), that contains a service name ServiceId, the requestor's identification and credentials, and the requested Action(s), to the PEP. The major PEP's task is to convert AuthZ request's semantics into the PDP request which semantics is actually defined by the used policy. When using an XACML policy and correspondingly an XACML PDP, the PEP will send an XACML AuthZ request to the PDP in the format (Subject, Resource, Attributes, (Environment)). The Policy Decision Point (PDP) evaluates the request and makes a decision whether to grant access or not. Based on a positive AuthZ decision (in one domain) the AuthZ ticket (AuthzTicket) can be generated by the PDP or PEP and communicated to the next domain where it may be processed as a security context or policy evaluation environment. If in general case the XACML policy contains obligations, they are returned in the XACMLAzResponse (AuthzDecision, Obligations). The PEP (or in more general case Context Handler) calls the Obligation Handler to process obligations.

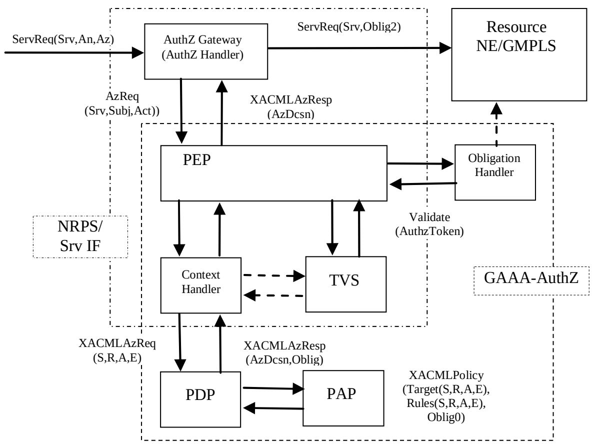

Figure 1. GAAA-AuthZ components providing service request evaluation

The service request may contain AuthZ ticket that hold extended AuthZ session context or AuthZ token that can just reference a local or global reservation ID, or identify an AuthZ session in which context the request is sent. The AuthZ token validation is performed by the Ticket/Token Validation Service (TVS). The TVS is typically called from the PEP and returns a confirmation if the ticket or token is valid and match the AuthZ request context. Introducing TVS as a separate function or service allow creating flexible token based policy enforcement infrastructure for on-demand network resource provisioning [6].

Using AuthZ tickets during the reservation stage for communicating interdomain AuthZ context is essential to ensure effective decision making. At the service access/consumption stage the reserved resource may be simply identified by the reservation ID created as a result of the successful reservation process. The proposed and implemented in GAAA-AuthZ AuthZ ticket format [8] allows communicating Obligations between domain based authorisation services as a part of the AuthZ context.

4. Obligation Handling Reference Model (OHRM)

In this section we will discuss the proposed Obligations Handling Reference Model (OHRM) to support typical

CRP use cases. Obligations are included into the policy definition and returned by PDP to PEP which in its turn should take actions as prescribed in the obligation instructions or statements.

Figure 2 below illustrates the proposed model for processing obligations in the general case of the Site Central AuthZ Service (SCAS). The SCAS means that all site/domain located resources and services use a central AuthZ service that maintains a common set of policies for this domain. The described processing model is compliant to the model used in XACML [1] but specifically focuses on the obligations handling dataflow and adds Web services based AuthZ callout interface.

A number of assumptions are made to reflect possible options in AuthZ service infrastructure implementation and different type of Obligations both stateful and stateless that are concerned with assigning pool accounts, enforcing quotas, controlling usable resource (e.g., number of resource access, purchased video/music listening time, etc.), logging and accounting.

It is important to notice that obligations are an integral part of the policy and typically included into the policy at the stage of its creation by the policy administrator or resource owner. For the manageability purpose, policy is considered stateless and the statefulness of obligations is achieved by the obligation handlers. The obligations enforcement process may include few stages and can be resulted either in modifying the service request (e.g., map from subject to account name/type) or by changing the resource/system sate or environment variables.

The obligations handling model allows two types of obligations execution: at the time of receiving obligations from the PDP and at the later time when accessing a resource or performing an authorised action. First type is described below; the second type of handling obligations can be achieved by using AuthZ ticket that holds obligations together with the AuthZ decision.

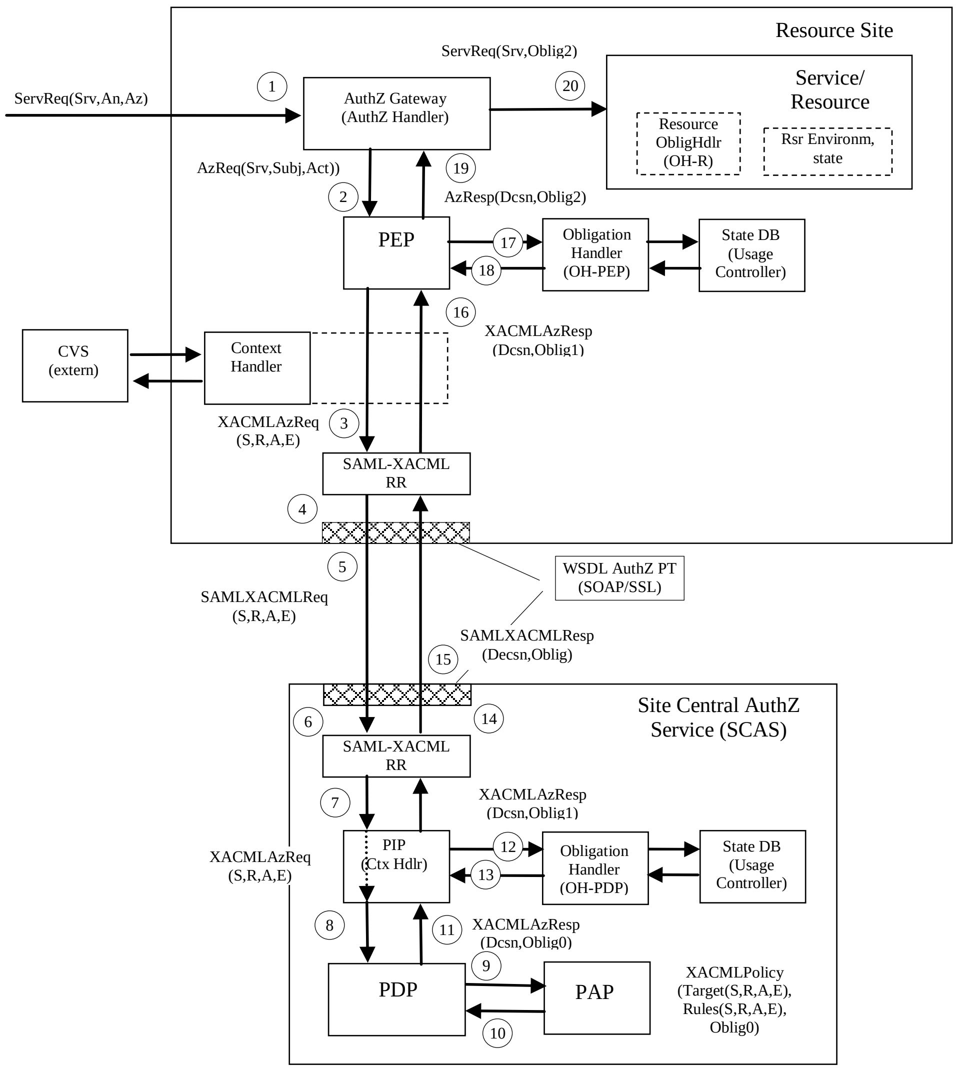
Resource Site
ServReq(Srv,Oblig2)
ServReq(Srv,An,Az)
1
AuthZ Gateway (AuthZ Handler)
20
Service/ Resource
Resource ObligHdlr (OH-R)
Rsr Environm, state
AzReq(Srv,Subj,Act))
19
2
AzResp(Dcsn,Oblig2)
PEP
17
18
Obligation Handler (OH-PEP)
State DB (Usage Controller)
16
XACMLAzResp (Dcsn,Oblig1)
CVS (extern)
Context Handler
XACMLAzReq (S,R,A,E)
3
SAML-XACML RR
4
5
WSDL AuthZ PT (SOAP/SSL)
SAMLXACMLReq (S,R,A,E)
SAMLXACMLResp (Decsn,Oblig)
15
Site Central AuthZ Service (SCAS)
6
14
SAML-XACML RR
7
XACMLAzResp (Dcsn,Oblig1)
XACMLAzReq (S,R,A,E)
PIP (Ctx Hdlr)
12
13
Obligation Handler (OH-PDP)
State DB (Usage Controller)
8
11
XACMLAzResp (Dcsn,Oblig0)
9
PDP
PAP
10
XACMLPolicy (Target(S,R,A,E), Rules(S,R,A,E), Oblig0)

Figure 2. Generic Authorisation dataflow and Obligations handling in distributed AuthZ service.

For the general (stateful) obligations handling process we can distinguish the following stages (note: not all stages are necessary to be implemented in a simple use case but they may exist in different cases):

```
Obligation0 = tObligation =>
 => Obligation1 (“OK?”, (Attributes1 V
    Environment1)) =>
  => Obligation2 (“OK?”, (Attributes2 V
     Environment2)) =>
    => Obligation3 (Attributes3 V
       Environment3)
```

1) ***Obligation0*** *(stateless)* - obligations are returned by the PDP in a form as they are written in the policy. These obligations can be also considered as a kind of templates or instructions, tObligation. (Important to mention that due to security reason obligations format and semantics should not use executable code or reference to locally executed commands).

2) ***Obligation1*** *or* ***Obligation2*** – obligations have been handled by the obligation handler at the SCAS/PDP side and/or at the PEP side correspondingly, depending on implementation. In this case templates or instructions of the Obligation0 are replaced with the real attributes in Obligation1, e.g. in a form of “name-value” pair. During this stage, the obligation handler can actually enforce obligations or modify obligations and send them further for enforcement by the resource. Introducing Obligation1 and Obligation2 handling stages gives flexibility to the proposed model as in many cases of the remote PEP and PDP location both sides may not have necessary information for the full obligations enforcement.

The result of obligations processing/enforcement, can be returned in a form of modified AuthzResponse (Obligation1) or in a form of global resource environment changes that will be taken into account at the time when the requested service/resource are provided or delivered. In both cases (and specifically in the last case) obligation handler should return notification about fulfilled obligated actions, e.g. in a form of Boolean value “False” or “True”, which will be taken into account by PEP or other processing module to finally permit or deny service request by PEP.

3) ***Obligation3*** – this is the final stage when obligations actually take effect, which can be defined as obligations “termination”. This is done by the resource itself or by trusted services managed/controlled by the resource.

In the proposed model, option with Obligation1 handling stage at the SCAS or PDP side is introduced to illustrate a case when we need to implement a stateful PDP/SCAS. This is achieved by adding obligations handling functionality to the Context Handler module which functionality is defined flexibly in the XACML specification.

One of the important aspects of the general obligations handling model is not discussed here, namely logical or time wise sequence of enforcing obligations. The solution was proposed at Open Grid Forum (OGF) OGSA Authorisation Working Group (AUTHZ-WG) [9] to add special Chronicle attribute to the Obligation element in XACML, but this idea has not been further discussed.

5. Implementation suggestions

The proposed OHRM is a result of the extended discussion at the joint Authorisation interoperability working group (AUTHZ-INTEROP WG) [10] between the major Grid consortia EGEE in Europe and Open Science Grid (OSG) in US with participation of the Globus Toolkit development team.

AUTHZ-INTEROP WG identified common attributes used for AuthZ in Grid and major types of obligations that are required in many Grid applications. All this is summarised in the document “An XACML Attribute and Obligation Profile for Authorization Interoperability in Grids” [11]. The attributes and obligations profile provides a basis for interoperability between potentially different AuthZ service implementations and can be used as a basis for defining new profiles, in particular for network resource provisioning.

Implementation of the proposed OHRM is based on the SAML 2.0 Profile of XACML 2.0 [12] that specifies extensions to the SAML 2.0 assertions and protocol [13] to support communication between XACML PEP and PDP. This is considered as another important component to achieve interoperability between different AuthZ frameworks in the distributed AuthZ infrastructure that not necessary use XACML policy language.

5.1. SAML-XACML Library

Initial implementation of the SAML-XACML library as extension to the OpenSAML2.0 library has been done in the framework of the GAAA-AuthZ Toolkits project [14, 15] and currently being contributed to the OpenSAML project [16]. The library is intended to be used with the Globus AuthZ Framework (GT-AuthZ) [17].

The SAML-XACML library allows plugging in multiple ObligationHandlers that support different types of obligations (that are identified by ObligationId) and can be called either from the PEP or from the SAML-XACML interface modules that handle request/response messages.

5.2. Obligations expression convention

Obligations expression in XACML can be described by the following general Obligation term:

```
   Obligation = Apply (TargetAttribute, Operation (Variables)), or
   Obligation = Apply (TargetAttribute, Operation (Variables), Chronicle)
```

Below example is provided only for illustration how account mapping obligation can be expressed in the XACML2.0 compliant format. Obligation type is identified by ObligationId attribute which value for this example contains value “map.poolaccount” that can used to call out to a designated ObligationHandler. (Note, the example uses a dedicated to the project namespace “http://authz-interop.org/xacml”).

```
<!-- Obligations format option 1 (UID, GID explicitly
mentioned as separate XML elements inside
AttributeAssignment element) -->
<Obligations>
<Obligation
   ObligationId="http://authz-
interop.org/xacml/obligation/map.poolaccount"
   FulfillOn="Permit">

<!-- This part specifies to what kind of attribute the
next ‘map.to’ action is applied to -->
<AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attr
ibute: requesting-subject"
DataType="http://www.w3.org/2001/XMLSchema#string">
    &lt;SubjectAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subj
ect-id"

DataType="http://www.w3.org/2001/XMLSchema#string"/&gt
;
</AttributeAssignment>

<!-- This is actual account attribute name/value to
which it should be mapped -->
```

```
<AttributeAssignment
AttributeId="http://authz-
interop.org/xacml/obligation/attribute/uidgid"
DataType="http://www.w3.org/2001/XMLSchema#string">
  &lt;UnixId
DataType="http://www.w3.org/2001/XMLSchema#string"&gt;
    okoeroo&gt;UnixId&gt;
  &lt; GroupPrimary
DataType="http://www.w3.org/2001/XMLSchema#string"&gt;
    computergroup&gt;GroupPrimary&gt;
  &lt;GroupSecondary
DataType="http://www.w3.org/2001/XMLSchema#string"&gt;
    datagroup&gt;GroupSecondary&gt;
</AttributeAssignment>
</Obligation>
</Obligations>
```

6. Conclusion

The results presented in this paper are the part of the ongoing research and development of the generic AAA Authorization framework and its targeted integration with Grid oriented authorisation frameworks such as LCAS/LCMAPS and GT4-AuthZ. This work is being conducted in the framework of different EU funded projects including EGEE and Phosphorus.

In this paper we proposed the Reference Model for policy Obligations Handling (OHRM) as further extension of the XACML authorisation model that is motivated and based on the real needs for interoperable distributed authorisation infrastructure for multidomain Complex Resource Provisioning.

The proposed OHRM allows for flexible policy definition in CRP, in particular, separation of the stateless authorisation rule-based policy definition that can be natively expressed in XACML and stateful policy enforcement using obligations.

The authors believe that the proposed model for policy obligations handling and related technical solutions will provide a good basis for interoperability and further discussion on different aspects of the general obligations definition and handling framework.

7. References

[1] "eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard", 1 February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_ control-xacml-2.0-core-spec-os.pdf

[2] Damianou, N., N. Dulay, E. Lupu, M. Sloman. The Ponder Policy Specification Language, Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan. 2001, Springer-Verlag LNCS 1995, pp. 18-39.

[3] ITU-T Rec. X.812 (1995) | ISO/IEC 10181-3:1996, Information technology - Open systems interconnection - Security frameworks in open systems: Access control framework. [Online]. Available: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.812-199511-I!!PDF-E&type=items

[4] RFC 2904 - "AAA Authorization Framework" J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, D. Spence, August 2000 - ftp://ftp.isi.edu/in-notes/rfc2904.txt

[5] Demchenko Y., L. Gommans, C. de Laat, “Using SAML and XACML for Complex Authorisation Scenarios in Dynamic Resource Provisioning”, Proceedings The Second International Conference on Availability, Reliability and Security (ARES 2007), April 10-13, 2007, Vienna.

[6] AAA Architectures for multi-domain optical networking scenario's, Phosphorus Project Deliverable D4.1, September 2007. [Online] Available: http://www.ist-phosphorus.eu/files/deliverables/Phosphorus-deliverable-D4.1.pdf

[7] Support for advance reservations in scheduling, Phosphorus Project Deliverable D5.4, September 2007. [Online] Available: http://www.ist-phosphorus.eu/files/deliverables/Phosphorus-deliverable-D5.4.pdf

[8] Demchenko Y., L. Gommans, C. de Laat, “Using SAML and XACML for Complex Resource Provisioning in Grid based Applications”, Proceedings IEEE Workshop on Policies for Distributed Systems and Networks (POLICY 2007), Bologna, Italy, 13-15 June 2007

[9] OGSA Authorization WG (OGSA-AUTHZ-WG) [Online]. https://forge.gridforum.org/projects/ogsa-authz

[10] Joint EGEE, OSG, Globus Authorization Interoperability Working Group [Online]. http://home.fnal.gov/~garzogli/privilege/AuthZInterop/info.html

[11] An XACML Attribute and Obligation Profile for Authorization Interoperability in Grids. Joint EGEE, OSG, and Globus document. [Online]. https://edms.cern.ch/document/929867/1

[12] SAML 2.0 Profile of XACML 2.0, Version 2. Working Draft 2, 26 June 2006. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/xacml-2.0-profile-saml2.0-v2.zip

[13] "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[14] AAAuthreach Project Information Page [Online]. http://staff.science.uva.nl/~demch/projects/aaauthreach/

[15] OpenSAML2.0 Extension Library to Support SAML2.0 profile of XACML2.0. [Online] http://www.bccs.uib.no/~hakont/SAMLXACMLExtension/

[16] OpenSAML Project. [Online]. https://spaces.internet2.edu/display/OpenSAML/Home

[17] GT 4.0: Security: Authorization Framework. [Online]. Available: http://www.globus.org/toolkit/docs/4.0/security/authzframe/