Aaauthreach Project Technical Report

Security Languages for Access Control and Authorisation: SAML and XACML Languages Overview

Contributor: Yuri Demchenko <demch@science.uva.nl>

Abstracts

The chapter provides comprehensive overview and introduction to two closely inter-related XML-based security languages standardised by OASIS: the Security Assertion Markup Language (SAML) and the eXtensible Access Control Markup Language (XACML) and their special profiles.

The chapter also provides a short overview of the access control models and related security services to create a background for better understanding the SAML and XACML and their relations.

The chapter also describes the policy Obligations Handling Reference Model (OHRM) that is proposed by the authors to extend the XACML Authorisation handling model for Grid and networking distributed applications.

The chapter describes an example of defining the XACML attributes and policy profile for Network Resource Provisioning (XACML-NRP) and provides practical suggestions for attribute format expression and policy identification.

It is intended that the provided information will be both helpful for specialists who needs a comprehensive introduction into SAML and XACML and will serve as a reference material for advanced users. The chapter refers to two Open Source libraries OpenSAML2.0 and SunXACML1.6 that provide reference implementation for SAML2.0 and XACML2.0. The SAML and XACML examples are provided for better understanding the topic.

Copyright note

1 Introduction

Protecting computer, network, information resources and data from unauthorised use and at the same time ensure their availability are two major task of the security services in open computer and communication systems.

Access control is enforced by access control systems that operates based on the predefined/preconfigured access control policy.

Consistent security context management is an important condition for consistent security services operation in open systems using client/server model and Service Oriented Architecture (SOA) Web Services [1, 2]. Security context need to be transferred between security services that may be located in different administrative and security domains. Under security context in general we understand information that is required and/or can be used for evaluating a service request according to the access control policy (e.g., user credentials or attributes defining their identity, permissions or roles) or access control session credentials/variables (that may include previous conditional authorisation decision, policy obligations, delegations or other session based restrictions).

Security context management is a part of secure object/service management environment. In the protected computer/system execution environment the security context can in a form of environment variables or program variables. However, if the security context need to be communicated between systems or services that run on different computers/systems and between different domains, the information and data need to be protected to ensure data confidentiality, integrity, authenticity and additionally non-repudiation.

This chapter provides overview of the industry standard languages for security assertions expression Security Assertion Markup Language (SAML) [3] that can be used for expressing user attributes/credentials and general purpose security assertions and for rule based access control policy expression eXtensible Access Control Markup Language (XACML) [4]. The chapter also provide a short overview of the basic access control concepts and models.

2 Basic Concepts and Models in Access Control

2.1 Discretionally Access Control and Mandatory Access Control

DAC suggests that the object owner defines a list of subjects or entities which are allowed access to the object. Typical example is file access control list. Only those users specified by the owner may have some combination of read, write, execute, and other permissions to the file. DAC policy tends to be very flexible and is widely used in the commercial and government sectors. However, DAC is known to be inherently weak for two reasons: granting read access is transitive; DAC policy is vulnerable to Trojan horse attacks exploring subject impersonation. Therefore, the drawbacks of DAC are as follows:

- Information can be copied from one object to another; therefore, there is no real assurance on the flow of information in a system.
- No restrictions apply to the usage of information when the user has received it.
- The privileges for accessing objects are decided by the owner of the object, rather than through a system-wide policy that reflects the organization’s security requirements.

ACLs and owner/group/other access control mechanisms are the most common mechanism for implementing DAC policies

Other access control models and policies are grouped in the category of non-discretionary access control (NDAC). As the name implies, policies in this category have rules that are not established at the discretion of the user. Non-discretionary policies establish controls that cannot be changed by users, but only through administrative action. Examples of NDAC are Separation of duty (SOD) and Mandatory Access Control (MAC). SOD policy can be used to enforce constraints on the assignment of users to roles or tasks. An example of such a static constraint is the requirement that two roles be mutually exclusive; if one role requests expenditures and another approves them, the organization may prohibit the same user from being assigned to both roles. Role-Based Access Control (RBAC) uses SOD as a part of its concept.

Mandatory access control (MAC) policy means that access control policy decisions are made by a central authority, not by the individual owner of an object, and the owner cannot change access rights. An example of MAC occurs in military security, where an individual data owner does not decide who has a Top Secret clearance, nor can the owner change the classification of an object from Top Secret to Secret. MAC is the most mentioned NDAC policy and uses the following approach: protection decisions must not be decided by the object owner; the system must enforce the protection decisions (i.e., the system enforces the security policy over the wishes or intentions of the object owner). Multilevel security models such as the Bell-La Padula Confidentiality and Biba Integrity models are used to formally specify this kind of MAC policy. However, information can pass through a covert channel in MAC, where information of a higher security class is deduced by inference such as assembling and intelligently combining information of a lower security class.

Extended overview and analysis of the basic access control models can be found in the NIST publication [5] or more research oriented paper [6].

2.2 Role Based Access Control

Although RBAC is technically a form of non-discretionary access control, it is often considered as one of the three primary access control policies (the others are DAC and MAC). In RBAC, access decisions are based on the roles that individual users have as part of an organization. Users take on assigned roles (such as professor, student, operator, or manager). Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. The use of roles to control access can be an effective means for developing and enforcing enterprise-specific security policies and for streamlining the security management process.

Under RBAC, users are granted membership into roles based on their competencies and responsibilities in the organization. The operations that a user is permitted to perform are based on the user's role. User membership into roles can be revoked easily and new memberships established as job assignments dictate. Role associations can be established when new operations are instituted, and old operations can be deleted as organizational functions change and evolve. This simplifies the administration and management of privileges; roles can be updated without updating the privileges for every user on an individual basis.

Generic RBAC model [7, 8, 9] provides an industry recognised solution for effective user roles/privileges management and policy based access control. It extends Discretional Access Control (DAC) and Mandatory Access Control (MAC) models with more flexible access control policy management adoptable for typical hierarchical roles and responsibilities management in organisations, but at the same time it suggest a full user access control management from user assignment to granting permissions. This can be suitable for internal organisational environment and particularly for human access rights management but reveals problems when applied to distributed service-oriented environment.

Sandhu in his two research papers [7, 8] describes 4 basic RBAC models:

- Core RBAC (RBAC0) that associates Users with Roles (U-R) and Roles with Permissions (R-P);
- Hierarchical RBAC (RBA1) that adds hierarchy to roles definition;
- Constrained RBAC (RBAC2) that extends RBAC0 with the constrains applied to U-R and R-P assignment;
- Consolidated RBAC (RBAC3) that adds role hierarchy to RBAC2.

RBAC is described in the ANSI INCITS 359-2004 standard [9] that partly re-defined the first three basic RBAC models in the context of static or dynamic separation of duties (SSD vs DSD). In both models, initial Sandhu's and ANSI RBAC, there is a notion of the user session which is invoked by a user and provides instant session-based U-R association. Final result/stage of the RBAC functionality are permissions assigned to the user based on static or dynamic U-R and R-P assignment. RBAC doesn't consider (user) permissions enforcement on the resource or access object. This functionality can be attributed to other more service-oriented frameworks such as ISO/ITU X.811/X.812 Authentication/Authorisation framework [10, 11] or generic AAA Authorisation framework [12, 13].

2.3 Generic AAA Authorisation Framework

Authentication, authorization, and accounting (AAA) is a term used to refer to a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage, and providing the information necessary to bill for services. These combined functions are considered important for effective network management and security.

The generic Authentication, Authorisation, Accounting (AAA) architecture was proposed in RFC2903 [12] and generic AAA Authorisation framework (GAAA-AuthZ) is described in RFC2904 [13] as a development of the ITU-T X.812 Authorisation framework [11] for distributed multidomain systems.

Authentication (AuthN) and Authorisation (AuthZ) are the components of the access control function to ensure that access to a resource or service is granted to the access subject (human, service or process) that has right to use the resource and perform those operation on the resource that it is allowed.

Authentication is the process of identifying a user or an access subject, based on identity credentials which examples are username and password, digital certificates, one-time-tokens, etc. Authentication refers to the confirmation that a user/subject who is requesting services is a valid user of the resources or services requested. Typically AuthN involves comparing a user's authentication credentials with the user credentials stored in a user database (UserDB) or the AuthN/AAA service, or checking validity of the user credentials obtained from the trusted AuthN service or trusted Identity Provider.

Based on positive AuthN, a user must obtain authorization for doing certain tasks. Authorization is the process of granting or denying a user access to network resources once the user has been authenticated. The amount of information and the amount of services the user will be granted depends on the user's authorization level which is defined by the user attribute credentials. In other words, Authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the context of authentication. Authenticated user is provided with the attributes that are required for authorisation decision.

Accounting is the process of keeping track of a user's activity while accessing the resources or services. Accounting is carried out by logging of session statistics and usage information and used for trend analysis, capacity planning, billing, auditing and cost allocation.

In modern Service Oriented Architecture (SOA) applications a Resource or a Service are protected by the site access control system that relies on both AuthN of the user and/or request message and AuthZ that applies access control policies against the service request. It is essential in a service-oriented model that AuthN credentials are presented as a security context in the AuthZ request and that they can be evaluated by calling back to the AuthN service and/or Attribute Authority (AttrAuth). This also allows for loose coupling of services in distributed hierarchical access control infrastructure.

The GAAA-AuthZ model is illustrated on Figure 1 and includes such major functional components as: Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Authority Point (PAP). It is naturally integrated with the RBAC separated User-Role and Role-Privilege management model that can be defined and supported by separate policies.

The Requestor requests a service by sending a service request ServReq to the Resource's PEP providing as much (or as little) information about the Subject/Requestor, Resource, Action as it decides necessary according to the implemented authorisation model and (should be known) service access control policies.

In a simple scenario, the PEP sends the decision request to the (designated) PDP and after receiving a positive PDP decision relays a service request to the Resource. The PDP identifies the applicable policy or policy set and retrieves them from the Policy Authority, collects the required context information and evaluates the request against the policy.

In order to optimise performance of the distributed access control infrastructure, the Authorisation service may also issue AuthZ assertion in the form of AuthzTicket that confirm access rights. They are based on a positive decision from the Authorisation system and can be used to grant access to subsequent similar requests that match an AuthzTicket. To be consistent, AuthzTicket must preserve the full context of the authorisation decision, including the AuthN context/assertion and policy reference.

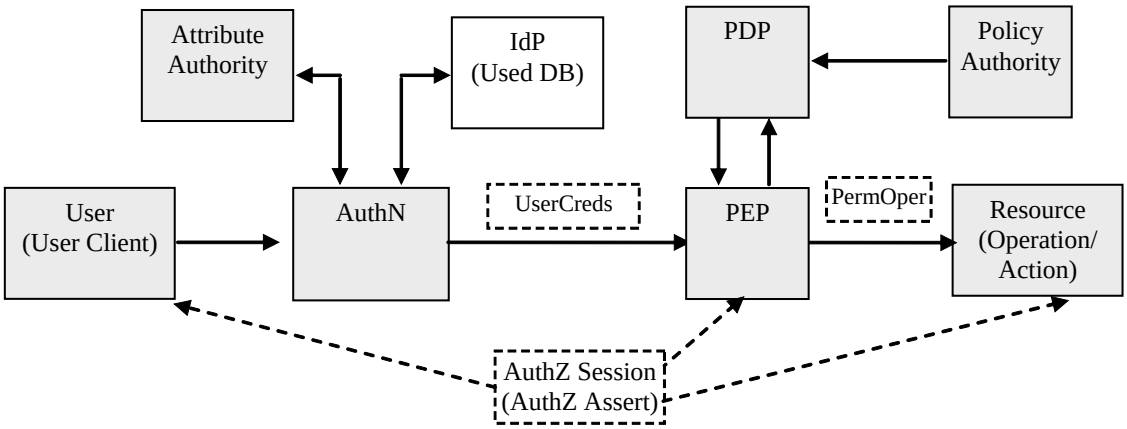

Figure 1. Generic Authentication and Authorisation services interaction.

2.4 Using SAML and XACML to support generic Authorisation scenario

The diagram below illustrates where SAML protocol and assertions and XACML Request/Response messages can be used in a typical policy based decision making [14].

The following sections will provide details about SAML and XACML languages and their use for access control in distributed service- oriented applications.

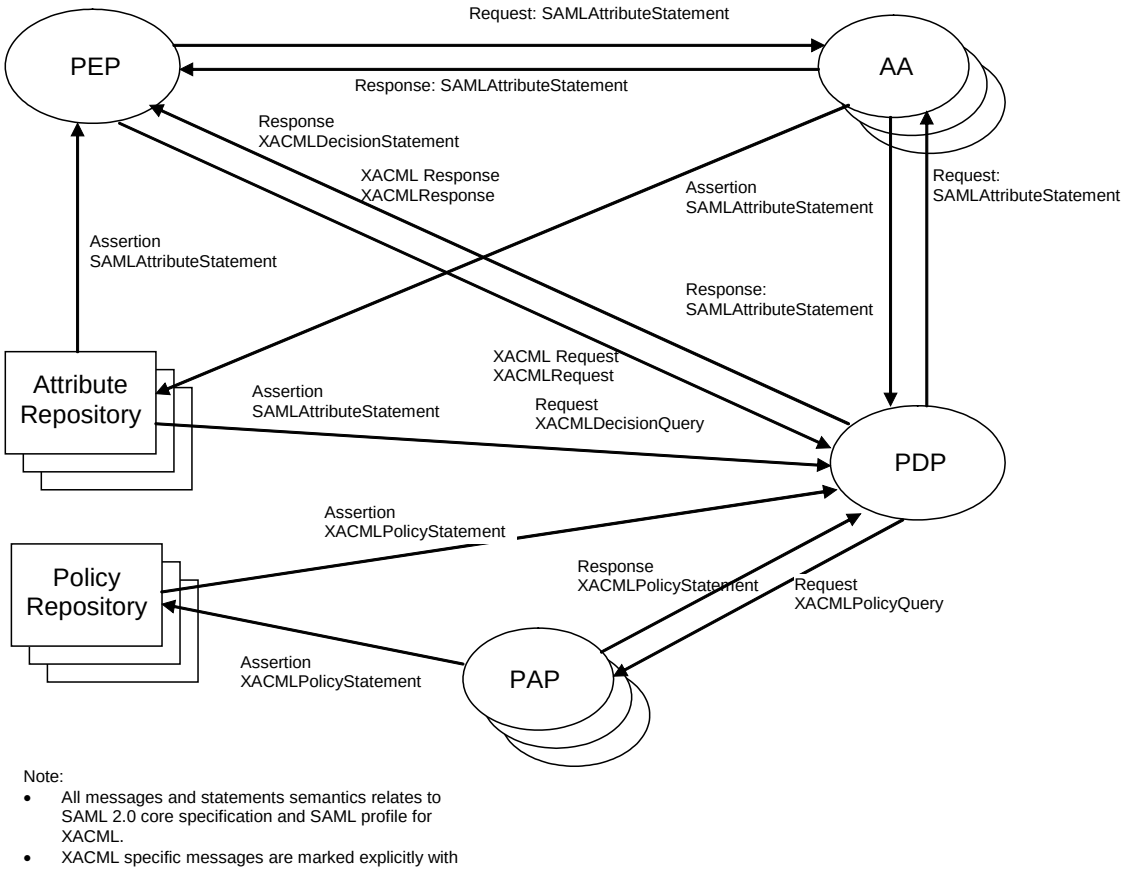

Figure 2. Using SAML and XACML for messaging and assertions

3 SAML security assertions expression and exchange format

3.1 SAML Overview

Security Assertion Markup Language (SAML) is a an XML-based standard for expressing and communicating authentication, authorisation and attribute information between distributed services.

The SAML operational security model suggests that all participating entities are members of the same security federation that have established business agreements, trust relations and share common attributes semantics [15]. More advanced SAML and Web Services based protocols can support attributes and assertions exchange between different federations and security domains.

SAML Version 1.1 specification was published in 2003 and has been broadly used in identity management, web access applications and Web services security. Current SAML Version 2.0 specification was published in 2006 and adopted experience of the two major SAML implementation areas such as Shibboleth [15] and Liberty Alliance Identity Federation Framework [16].

The major SAML application areas include:

Web Single Sign-On (WebSSO) allows a user who has authenticated to one web site to access other web sites that are the members of the same federation. SAML enables SSO providing a mean to communicate an authentication assertion from the original login site to other sites a user wants to access or where the user request is forwarded or redirected. The assertion then can be verified and validated and user authentication is confirmed.

Attribute-Based Authorisation allows granting or denying user access to the protected resources based on user attributes that can be groups, roles or other specific to applications user characteristics. SAML provides a mechanism to communicate user attributes in addition to the user identity. User identity and attributes are managed and provide by the Identity Provider (IdP) and Attribute Authority Service (AAS) that operates as a part of federation. Separating IdP/AAS from Authentication and Authorisation services simplifies typically distributed identity and access control infrastructure management.

Web Services Security (WS-Security) framework uses SAML as one kind of the security tokens within SOAP messages to convey security and identity information between actors in Web services interactions. The WS-Security SAML Token Profile is used by the Liberty Alliance's Identity Web Services Framework (ID-WSF) [18], Web Services Trust and Web Service Federation frameworks to support SSO, identity federation, identity mapping and other services.

3.2 SAML Basic Concepts and Components

SAML specification and architecture defines basic building components that allow a number of use cases and supports transfer of identity, attribute and authorisation information between autonomous entities that have established trust relations. The core SAML specification defines the structure and content of both assertion and protocol messages used to transfer this information.

The means by which lower-level communication or messaging protocols (such as HTTP or SOAP) are used to transport SAML assertion or protocol messages is defined by the SAML bindings. SAML profiles define constrains and/or extensions to SAML assertions, protocol or binding to support the usage of SAML for a particular use case or application.

Two other concepts used for building and deploying interoperable SAML environment are metadata and authentication context.

Metadata defines a way to express and share configuration information between SAML parties and include the following data: site's supported SAML bindings, operational roles (IdP, Service Provider (SP), etc), identifier information, supporting identity attributes, federation names, and trusted keys information for encryption and signing.

Authentication context defines a way to provide information regarding the type and strength of authentication that a user employed when they authenticated at an identity provider. This information is provided as a part of an assertion's authentication statement. An SP can also include an authentication context in a request to an IdP to request that the user be authenticated using a specific set of authentication requirements, such as a multi-factor authentication.

Figure 3 below illustrates relations between the basic SAML concepts and components and more details provided below [15]. Figure 4 illustrates two basic use cases of the SAML protocol response message containing SAML assertions that is carried by SOAP over HTTP (using HTTP and SOAP

binding) (a) and the SAML assertion use in Web Services Security (using WS-Security SOAP token profile [19]).

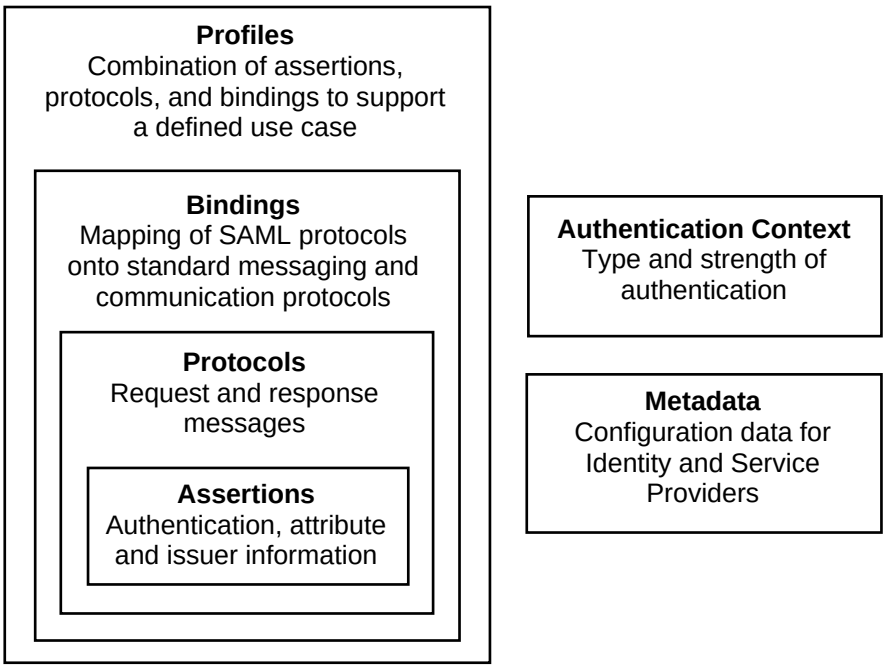

Figure 3. SAML components [15].

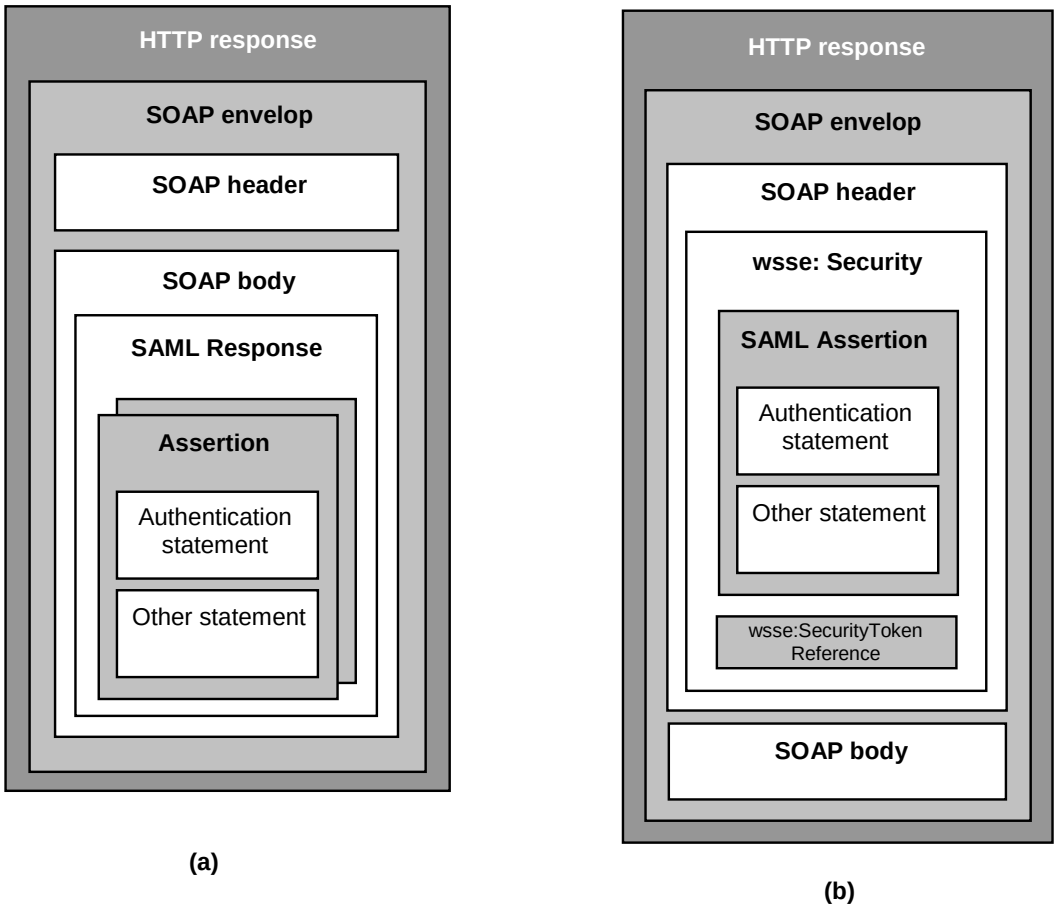

Figure 4. SAML protocol and assertions over HTTP (a) and with Web Services Security (b).

SAML Assertions

SAML allows for one party to assert security information in the form of statements about a subject. An assertion contains some basic required and optional information that applies all assertions, and usually contains a subject of the assertion, conditions used to validate the assertion, and assertion statements. SAML defines three kinds of statements that can be carried within an assertion:

Authentication statements: These are created by the party that successfully authenticated a user. At a minimum, they describe the particular means used to authenticate the user and the specific time at which the authentication took place.

Attribute statements: These contain specific identifying attributes about the subject (for example, that user “John Doe” is a member of “Project A” with role “researcher”).

Authorization decision statements: These are issued based on the authorisation decision may state what the subject is entitled to do on the given resource (for example, “John Doe” is permitted to “create-reservation”, “start-experiment-session” on the resource “Electronic Microscope XPS8076”). Authorisation decision statement defined by the SAML2-XACML2 profile may contain full authorisation context (see details below).

SAML Protocols

SAML defines a number of generalised request/response protocols:

Assertion Query and Request Protocol: This is the basic SAML protocol that defines a set of queries by which SAML authentication, authorisation or attribute assertions may be obtained. The Query form of this protocol defines how a relying party can ask for assertions (new or existing) on the basis of a specific subject and the desired statement type.

Authentication Request Protocol: Defines a means by which a principal (or an agent acting on behalf of the principal) can request assertions containing authentication statements and, optionally, attribute statements. This protocol is used in Web Browser SSO Profile when redirecting a user from an SP to an IdP in order authenticate user and optionally obtain user attributes.

Single Logout Protocol: Defines a mechanism to allow logout of active sessions associated with a principal. The logout can be directly initiated by the user, or initiated by an IdP or SP because of a session timeout, administrator command, etc.

Artifact Resolution Protocol: Provides a mechanism by which SAML protocol messages may be passed by reference using a small, fixed-length value called an artifact. The artifact receiver uses the Artifact Resolution Protocol to ask the message creator to dereference the artifact and return the actual protocol message.

Name Identifier Management and Name Identifier Mapping Protocols: Provide mechanisms to change or map the value or format of the name identifier used to refer to a principal. The issuer of the request can be either the service provider or the identity provider.

SAML Bindings

SAML bindings detail exactly how the various SAML protocol messages can be carried over underlying transport protocols. The bindings defined by SAML V2.0 are:

HTTP POST, Redirect, Artifact Bindings: Define how SAML protocol messages can be transported using HTTP POST, redirect or artefact messages.

SAML SOAP Binding: Defines how SAML protocol messages are transported within SOAP 1.1 messages, with details about using SOAP over HTTP.

Reverse SOAP (PAOS) Binding: Defines a multi-stage SOAP/HTTP message exchange that permits an HTTP client to be a SOAP responder. Used in the Enhanced Client and Proxy Profile and particularly designed to support WAP gateways.

SAML URI Binding: Defines a means for retrieving an existing SAML assertion by resolving a URI.

SAML Profiles

SAML profiles define how the SAML assertions, protocols, and bindings are combined and constrained to provide greater interoperability in particular usage scenarios. The profiles usually named by used protocol and a defined application area and include the following major profiles:

Web Browser SSO Profile: Defines how SAML entities use the Authentication Request Protocol and SAML Response messages and assertions to achieve single sign-on with standard web browsers. It defines how the messages are used in combination with the HTTP Redirect, HTTP POST, and HTTP Artifact bindings.

Assertion Query/Request Profile: Defines how SAML entities can use the SAML Query and Request Protocol to obtain SAML assertions over a synchronous binding, such as SOAP.

Enhanced Client and Proxy (ECP) Profile: Defines a specialized SSO profile where specialized clients or gateway proxies can use the Reverse-SOAP (PAOS) and SOAP bindings.

Single Logout Profile: Defines how the SAML Single Logout Protocol can be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.

Identity Provider Discovery Profile: Defines one possible mechanism for service providers to learn about the identity providers that a user has previously visited.

Other profiles are defined for Artifact Resolution Protocol, Name Identifier Management and Name Identifier Mapping Profile.

3.3 SAML Assertion datamodel and format

3.3.1 SAML top level elements

Figures below provide more detailed breakdown for SAML 2.0 Assertion format. The root element is called Assertion and mandatory contains the Issuer element and attributes Version, ID and IssueInstant. Depending on the profile the Assertion element may contain one or many statements such as defined in the standard AuthnStatement, AuthzDecisionStatement, AttributeStatement, or application defined statement that can be added through the abstract Statement element providing standard extension point. Other optional elements include Subject which is important in many profiles and use cases dealing with the identity information, Conditions and Advice. SAML Assertion may contain attached signature defined by the XML Signature standard.

In the compact XML DTD format the Assertion element can be descried as:

```
<!ELEMENT Assertion (Issuer, Signature?, Subject?, Conditions?, Advice?,
(Statement | AuthnStatement | AuthzDecisionStatement | AttributeStatement)*)>
<!ATTLIST Assertion
      Version CDATA #REQUIRED
```

```
       ID ID #REQUIRED
       IssueInstant CDATA #REQUIRED
>
```

The Subject element consists of two basic components – subject ID that can be expressed in different formats and SubjectConfirmation that provides information how the subject identity was verified or authenticated. Both types of information can be encrypted.The Subject element contains the following sub-elements:

```
<!ELEMENT Subject (((BaseID | NameID | EncryptedID), SubjectConfirmation*) |
SubjectConfirmation+)>
<!ELEMENT SubjectConfirmation (SubjectConfirmationData?)>
<!ATTLIST SubjectConfirmation
       Method CDATA #REQUIRED
>
<!ELEMENT SubjectConfirmationData (#PCDATA | *)*>
<!ATTLIST SubjectConfirmationData
       NotBefore CDATA #IMPLIED
       NotOnOrAfter CDATA #IMPLIED
       Recipient CDATA #IMPLIED
       InResponseTo CDATA #IMPLIED
       Address CDATA #IMPLIED
>
<!ELEMENT SubjectLocality EMPTY>
<!ATTLIST SubjectLocality
       Address CDATA #IMPLIED
       DNSName CDATA #IMPLIED
>
```

SAML Assertion provides the facility to describe conditions for assertion/credentials use and validity in the Conditions element that contains time validity constrains attributes, and elements that describe audience/community restriction, proxy/delegation restrictions and can also be extended to other application defined conditions.

The Advice element contains any additional information that the SAML authority wishes to provide. This information may be ignored by applications without affecting either the semantics or the validity of the assertion. Some potential uses of the Advice element include evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions), timing and distribution points for updates to the assertion, etc.

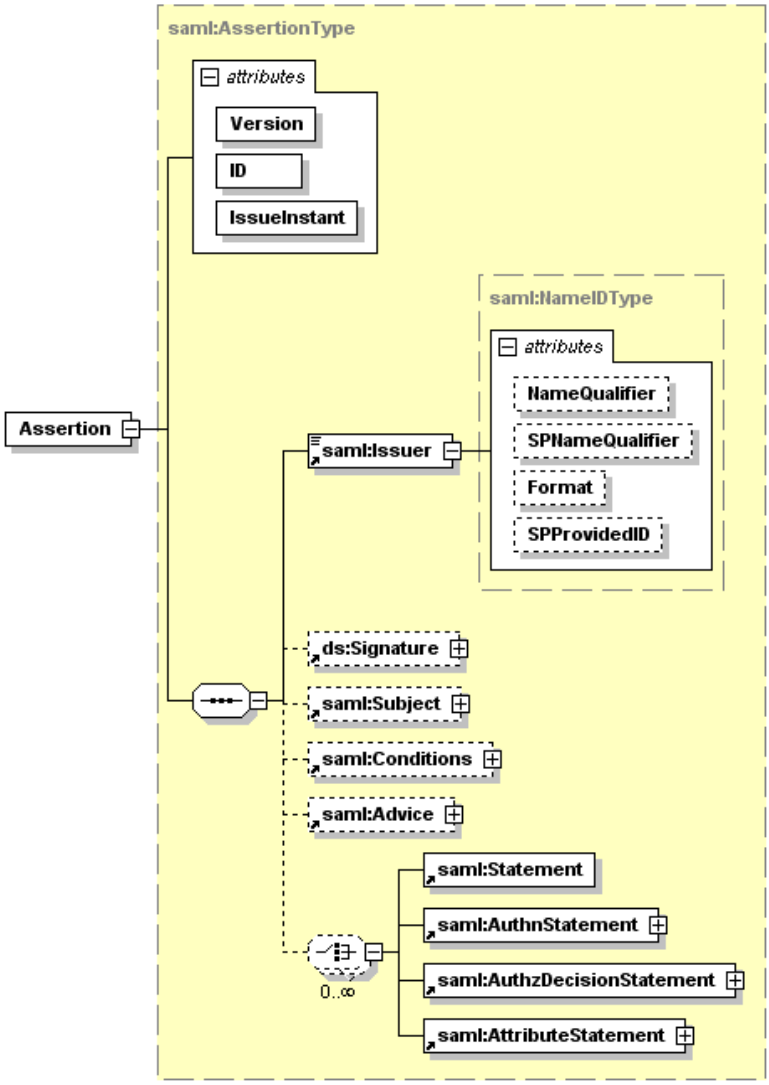

Figure 5. SAML Assertion top elements

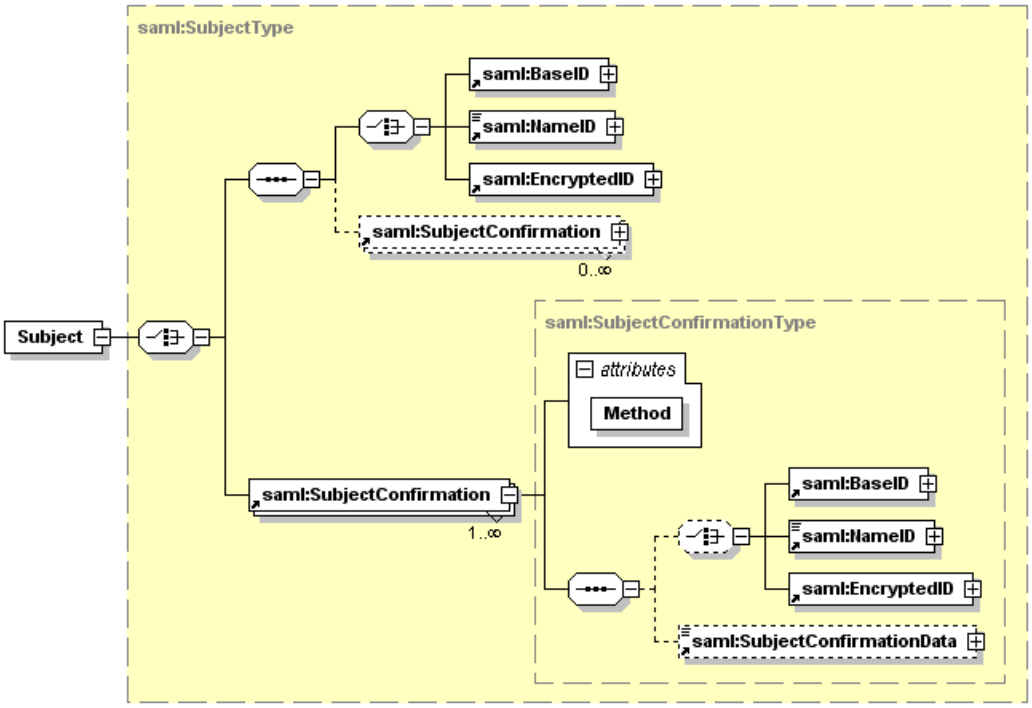

Figure 6. SAML Subject elements

3.3.2 SAML AuthnStatement and AttributeStatement format

The SAML AuthnStatement is used to convey authentication statement issued by an Identity Provider or an authentication service and has the following structure:

```
<!ELEMENT AuthnStatement (SubjectLocality?, AuthnContext)>
<!ATTLIST AuthnStatement
       AuthnInstant CDATA #REQUIRED
       SessionIndex CDATA #IMPLIED
       SessionNotOnOrAfter CDATA #IMPLIED
>
<!ELEMENT AuthnContext (((AuthnContextClassRef, (AuthnContextDecl |
AuthnContextDeclRef)?) | (AuthnContextDecl | AuthnContextDeclRef)),
AuthenticatingAuthority*)>
<!ELEMENT AuthnContextClassRef (#PCDATA)>
<!ELEMENT AuthnContextDecl (#PCDATA)>
<!ELEMENT AuthnContextDeclRef (#PCDATA)>
<!ELEMENT AuthenticatingAuthority (#PCDATA)>
```

The AuthnStatement has one mandatory attribute AuthnInstant that specifies the time at which the authentication took place, and two optional attributes SessionIndex that specifies the index of a particular session between the principal identified by the subject and the authenticating authority, and SessionNotOnOrAfter that specifies a time instant at which the session between the principal identified by the subject and the

The SubjectLocality specifies the DNS domain name and IP address for the system from which the assertion subject was apparently authenticated. SAML authority issuing this statement must be considered ended. The AuthnContext element specifies the context of an authentication event. The element can contain an authentication context class reference, an authentication context declaration or declaration reference, or both.

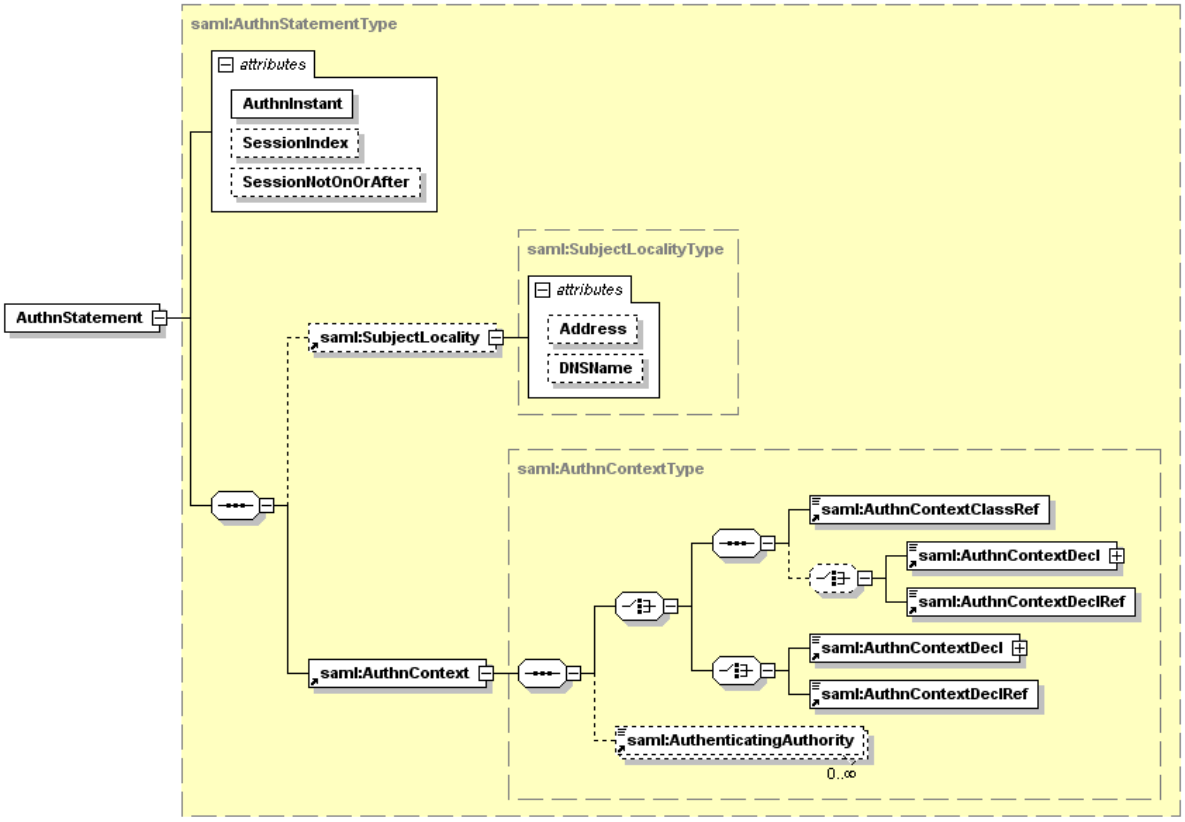

Figure 7. SAML AuthnStatement elements

Listing below provides an example of the authentication Assertion containing AuthnStatement element.

```
<Assertion xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
ID="e0fcd9f023440a05d540ba365e1ed1fe" IssueInstant="2004-12-29T17:14:24.085Z"
Version="2.0">
  <Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:X509SubjectName"
NameQualifier="cnl:subject:subject:AAAuthority">CN=Agent Smith, O=Matrix,
C=NL</Issuer>
  <Subject>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:emailAddress"
NameQualifier="cnl:subject:customer">WHO740@users.collaboratory.nl</NameID>
    <SubjectConfirmation>
      <ConfirmationMethod>email</ConfirmationMethod>
      <ConfirmationMethod>callback</ConfirmationMethod>
    </SubjectConfirmation>
  </Subject>
  <Conditions NotBefore="2004-12-28T23:00:00.000Z" NotOnOrAfter="2005-01-
29T21:22:22.000Z"/>
  <AuthnStatement AuthenticationInstant="2004-12-29T17:14:23.875Z"
AuthenticationMethod="AuthenticationMethod_X509_PublicKey">
    <SubjectLocality DNSAddress="dns.collaboratory.nl" IPAddress="192.30.180.22"/>
  </AuthnStatement>
</Assertion>
```

Figure 8. Example SAML 2.0 Authentication Assertion

The SAML AttributeStatement provides a format for communicating Subject's attributes issued by the Attribute Authority or Identity Provider. Figure 9 shows the structure of the SAML AttributeStatement element. It contains the following elements:

```
<!ELEMENT AttributeStatement (Attribute | EncryptedAttribute)+>
<!ELEMENT Attribute (AttributeValue*)>
<!ATTLIST Attribute
       Name CDATA #REQUIRED
       NameFormat CDATA #IMPLIED
       FriendlyName CDATA #IMPLIED
>
```

The AttributeStatement element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. Assertions containing AttributeStatement elements must contain a Subject element. The AttributeStatement element may contain either attribute reference/value or encrypted attribute.

The Attribute element contains The Attribute element is used within an attribute statement to express particular attributes and values associated with an assertion subject, it identifies an attribute by name and optionally includes its value(s). The Attribute element has a obligatory attribute Name that holds the name of attribute, and optional attributes the NameFormat representing the classification of the attribute name in URI format, and the FriendlyName providing a more human-readable form of the attribute's name, which may be useful in cases in which the actual Name is complex or opaque, such as an OID or a UUID.

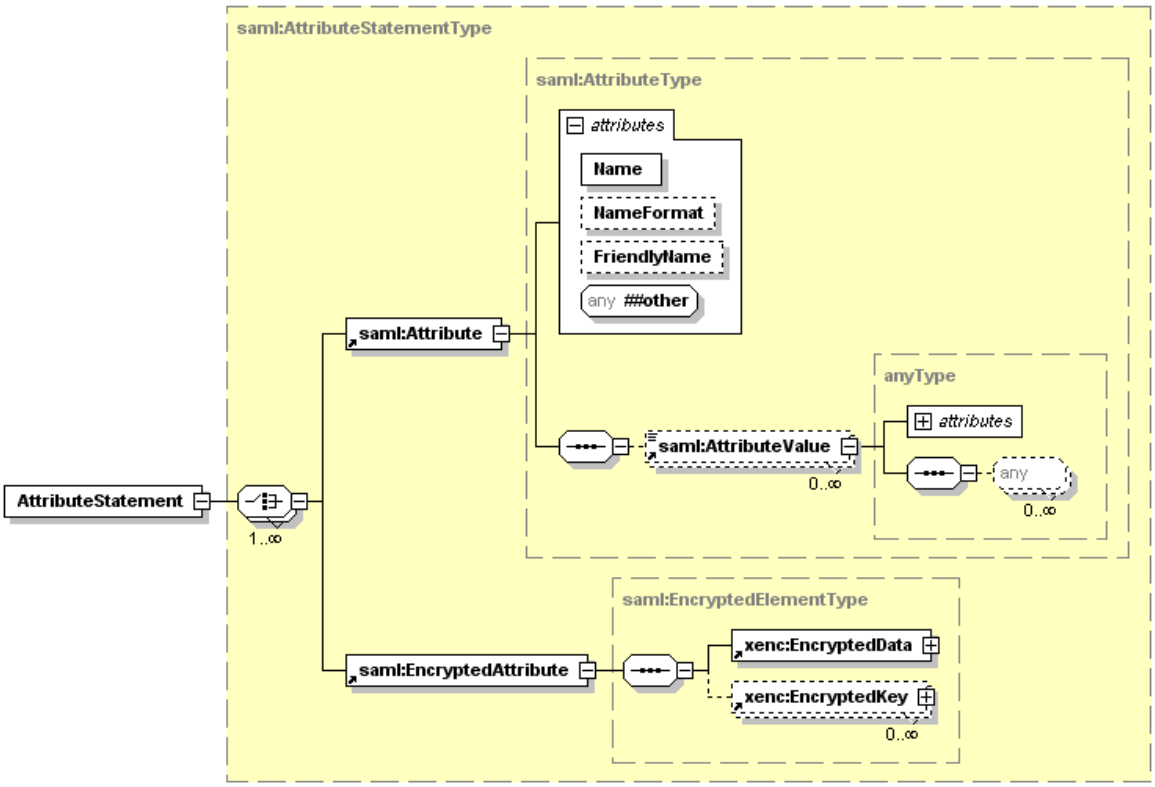

Figure 9. SAML AttributeStatement elements

Listing below provides an example of the authentication Assertion containing AuthnStatement element.

```
<Assertion xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
ID="b4d00e1500d2a10a43d3d2fb5a578028" IssueInstant="2004-12-29T17:17:24.164Z"
Version="2.0">
  <Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:X509SubjectName"
NameQualifier="cnl:subject:subject:AAAuthority">CN=Agent Smith, O=Matrix,
C=NL</Issuer>
  <Subject>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:emailAddress"
NameQualifier="cnl:subject:customer">HEIS007@staff.collaboratory.nl</NameID>
    <SubjectConfirmation>
      <ConfirmationMethod>email</ConfirmationMethod>
      <ConfirmationMethod>callback</ConfirmationMethod>
    </SubjectConfirmation>
  </Subject>
  <Conditions NotBefore="2004-12-28T23:00:00.000Z" NotOnOrAfter="2005-01-
29T21:22:22.000Z"/>
  <AttributeStatement>
    <Attribute xmlns:typens="urn:cnl" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
AttributeName="AttributeSubject" AttributeNamespace="urn:cnl">
      <AttributeValue
xsi:type="typens:subject">@cnl:subject:role:manager</AttributeValue>
      <AttributeValue xsi:type="typens:subject">cnl:subject:role</AttributeValue>
      <AttributeValue xsi:type="typens:subject">jobID</AttributeValue>
    </Attribute>
  </AttributeStatement>
</Assertion>
```

Figure 10. Example SAML 2.0 Attribute Assertion

4 XACML policy expression and messaging format

4.1 XACML overview

4.1.1 XACML Policy logical model

XACML (eXtensible Access Control Markup Language) is the OASIS standard that provides a well defined policy language with rich functionality to express complex rules for access control to different types of resources. XACML allows defining different application specific profiles. A number of special XACML profiles are discussed below.

The XACML policy logical model in a simple way can be presented as below. 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. The Target element actually defines a matching expression between (S-R-A-E) of the request and the policy:

```
Target (S, R, A, E) =>
      => Target (M(Sreq,Spol), M(Rreq,Rpol), M(Areq,Apol), M(Ereq,Epol))
```

where M – is a matching function between attributes provided in the request and embedded in the policy. It is important to mention that XACML allows only 2 variables matching functions in the Target element which however can be cascaded [4].

XACML policy may contain a number of rules which in its own turn may contain a number of conditions and a rule Target used for rules matching (or selection). The Conditions can use a wide range of functions defined in the XACML specification [4]. The following describes the structure of the Rule element:

```
Rule(Target (S, R, A, E),
     Cond (F(Sreq, Spol), F(Rreq, Rpol), F(Areq, Apol), F(Ereq, Epol)),
     Obligation)
```

where F – is a logical function between attributes provided in the request and embedded in the policy.

Additional flexibility for XACML policy rules definition is provided by the possibility to use the full functionality of the XPath expressions that can refer to the ResourceContent element of the XACML Request message.

The XACML policy can also specify the policy 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. Obligations are included into the policy definition and returned by PDP to PEP which in its own turn should take actions as prescribed in the Obligation instructions or statements. As an example, policy obligations may prescribe that some actions must be logged or user account must be changed or mapped to another account when accessing the resource.

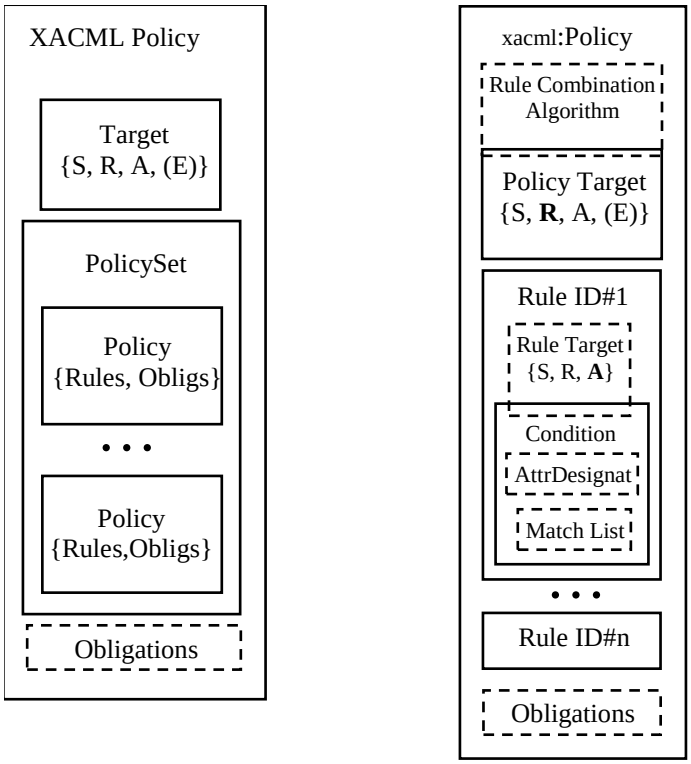

Figure 11. XACML policy model.

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. Few policies may be combined to form a single policy set that is applicable to the request. XACML specifies a number of policy and rule combination algorithms. The Response message may contain multiple Result elements, which are related to individual Resources.

Any of S-R-A-E elements allow for extensible “Attribute/AttributeValue” definition to support different attributes semantics and data types. Additionally, XACML allows for referencing internal and external XML documents elements by means of XPath functionality.

Two mechanisms can be used to bind the XACML policy to the resource: a Target element can contain any of S-R-A-E attributes and a policy identification attribute IDRef. XACML policy format provides few mechanisms to add and handle domain or session related context during the policy selection and request evaluation:

- Policy identification that is done based on the Target comprising of the Resource, Action, Subject, and optionally Environment elements.
- Attributes semantics and metadata can be namespace aware and used for attributes resolution during the request processing.

In complex authorisation scenarios the security context e.g. from the previous authorisation decision can be provided as an environment or resource attribute.

4.1.2 XACML Authorisation dataflow

The generic authorisation infrastructure consists of

- RBE (Rule Based Engine) as a central policy based decision making point,
- PEP (Policy Enforcement Point) providing Resource specific AuthZ decision request/response handling and policy defined obligations execution,

- PAP (Policy Authority Point) or Policy DB as a policy storage (in general, distributed),
- PIP (Policy Information Point) providing external policy context and attributes to the RBE including subject credentials and attributes verification
- RIP (Resource Information Point) that provides resource context.
- AA (Attribute Authority) that manages user attributes

To allow user access to the resource, Resource Agent requests via a Policy Enforcement Point (PEP) an authorisation decision from a Policy Decision Point (PDP) that evaluates the authorisation request against the policy defined for a particular job, resource and user attributes/roles. The access policy is defined by the resource owner and stored in the policy repository.

The PEP and PDP may also request specific user attributes or credentials from the Authentication service, or additional information from the Resource/Instrument.

Figure 12 illustrates an authorisation process dataflow when processing authorisation request by XACML compatible systems [4]. To get user access to the resource, Resource authorisation gateway requests via a Policy Enforcement Point (PEP) an authorisation decision from a Policy Decision Point (PDP) that evaluates the authorisation request against the policy defined for a particular action, resource and user attributes/roles. The access policy is defined by the resource owner and stored in the Policy Authority Point (PAP). The PEP and PDP may also request specific user attributes or credentials from the Authentication or Attribute Authority service, or additional information from the Resource/Instrument.

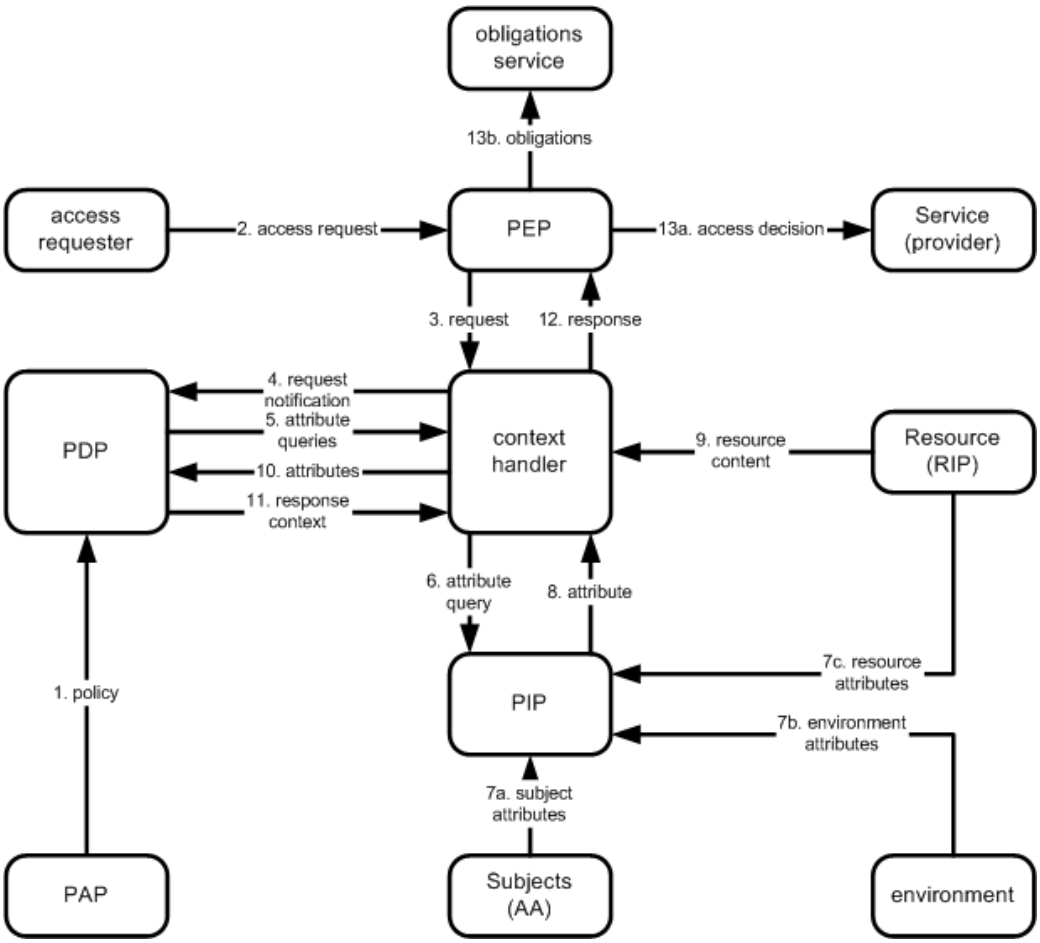

Figure 12. XACML dataflow model showing the major actors and sequences [4].

The following sequence explains what steps are suggested during the XACML authorisation request evaluation:

1. PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.
2. The access requester sends a request for access to the PEP.
3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.
4. The context handler constructs an XACML request context and sends it to the PDP.
5. The PDP requests any additional subject, resource, action and environment attributes from the context handler.
6. The context handler requests the attributes from a PIP.
7. The PIP obtains the requested attributes.
8. The PIP returns the requested attributes to the context handler.
9. Optionally, the context handler includes the resource in the context.
10. The context handler sends the requested attributes and (optionally) the resource to the PDP. The PDP evaluates the policy.
11. The PDP returns the response context (including the authorization decision) to the context handler.
12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.
13. If access is permitted, then the PEP permits access to the resource; otherwise, it denies access. The PEP fulfils the obligations, generally, for both cases of possible PDP solutions.

4.1.3 XACML 2.0 special profiles

XACML 2.0 RBAC profile [20]

XACML RBAC profile describes how to built Policies requiring multiple Subjects and roles combination to access a resource and perform an action. Multiple Subject elements in XACML allow flexibility when implementing hierarchical RBAC model for such cases when some actions require superior subject/role approval to perform a specific action. One or more `<Subject>` elements are allowed. A subject is an entity associated with the access request. For example, one subject might represent the human user that initiated the application from which the request was issued; another subject might represent the application's executable code responsible for creating the request; another subject might represent the machine on which the application was executing; and another subject might represent the entity that is to be the recipient of the resource.

XACML Multiple Resources profile [21]

The conditions under which multiple `<Resource>` elements are allowed are described in the XACML Profile for Multiple Resources. XACML Multiple Resources profile SHALL be interpreted as a request for access to all resources specified in the individual <Resource> elements. For each <Resource>element, one Individual Resource Request SHALL be created. This Individual Resource Request SHALL be identical to the original request context with one exception: only the one <Resource>element SHALL be present. If such a <Resource> element contains a "scope " attribute having any value other than "Immediate", then the Individual Resource Request SHALL be further processed according to the corresponding enumerated value of this attribute. This processing may

involve decomposing the one Individual Resource Request into other Individual Resource Requests before evaluation by the PDP.

XACML 2.0 Profile for Hierarchical Resources [22]

The hierarchical resource profile specifies how XACML can provide access control for a resource that is organized as a hierarchy, which examples are Examples include file systems, XML documents, and organizations. In this case resource is presented as set hierarchical nodes which are referred to as `resource-parent`, `resource-ancestor`, and `resource-ancestor-or-self`.

XACML 2.0 Privacy Policy Profile [23]

This profile provides standard attributes and a standard <Rule> element for enforcing the privacy protection principles, related to the purpose for which personally identifiable information is collected and used.

This specifies the following attributes:
`"urn:oasis:names:tc:xacml:2.0:resource:purpose"`

> This attribute indicates the purpose for which the data resource was collected. The owner of the resource SHOULD be informed and consent to the use of the resource for this purpose.

`"urn:oasis:names:tc:xacml:2.0:action:purpose"`

> This attribute indicates the purpose for which access to the data resource is requested.

XACML 2.0 XML Digital Signature Profile [24]

The profile provides a profile for use of the W3C XML-Signature Syntax and Processing Standard in providing authentication and integrity protection for XACML schema instances. The signature information must include a specification of the identity of the signer and a specification of the period during which the signed data object is to be considered valid.

Cross-Enterprise Security and Privacy Authorization (XSPA) Profile for Healthcare [25]

This profile is currently in public comments stage. The Cross-Enterprise Security and Privacy Authorization (XSPA) profile of XACML describes several mechanisms to authenticate, administer, and enforce authorization policies controlling access to protected information residing within or across enterprise boundaries. The policies being administered and enforced relate to security, privacy, and consent directives. This profile MAY be used in coordination with additional standards including Web Services Trust Language (WS-Trust) and Security Assertion Markup Language (SAML).

4.1.4 XACML 3.0 Specifications and profiles currently under review

XACML 3.0 specification [26] is currently under review at the final stage as a candidate specification. The new specification provides better definition of the PEP-PDP interaction, adds the Advice element that in contrary to the Obligation element is not obligatory for enforcing by PEP, extends both the Obligation and Advice elements content with the ObligationExpression, AdviceExpression and AttributeExpression elements.

In the new specification the Policy and the Request and Response are defined by common schema with the “xacml” namespace.

The structure of the XACML 3.0 Request was simplified and all attributes are now placed under the single Attributes element, that contains two elements Attribute and Content, and can be distinguished

by their AttributeId's. The Request reference multiple requests connected to the current one placed into the MultiRequests element.

The response is extended to contain elements AssociatedAdvice that hold returned by PDP policy advices, Status and PolicyIdentifierList.

The profiles of XACML 2.0 are updated and the following new profiles are proposed.

XACML 3.0 Administration and Delegation Profile [27]

The XACMLv3.0 Administrative and Delegation profile can indicate if the policy is issued by the trusted PolicyIssuer for the particular domain. In this case the PDP will rely on an already assigned or default PAP and established trust relations, otherwise when other entity is declared as a PolicyIssuer, the PDP should initiate checking administrative policy and delegation chain what is a suggested functionality of the PIP module.

XACML PDP Metadata (Working draft) [28]

XML PDP Metadata profile specifies an extensible schema for publishing information about PDP such as the version of XACML implemented, supported standard functions and combining algorithms, supported optional features, and the location of the PDP.

4.2 XACML2.0 policy datamodel

XACML provides a format for expressing policy for the generic Attribute Based Access Control (ABAC) model and defines a simple Request/Response messages format.

Decision request sent in a Request message provides context for policy-based decision. The complete policy applicable to a particular **decision request** may be composed of a number of individual **rules** or **policies**. Few policies may be combined to form the single policy applicable to the request.

XACML defines three top-level policy elements: `<Rule>`, `<Policy>` and `<PolicySet>` [4]. The <Rule> element contains a Boolean expression that can be evaluated in isolation, but that is not intended to be accessed in isolation by a **PDP**. So, it is not intended to form the basis of an **authorization decision** by itself. It is intended to exist in isolation only within an XACML **PAP**, where it may form the basic unit of management, and be re-used in multiple **policie**s.

The `<Policy>` element (see Figure 13) contains a set of `<Rule>` elements and a specified procedure for combining the results of their evaluation. It is the basic unit of **policy** used by the **PD**P, and so it is intended to form the basis of an **authorization decisio**n.

The `<PolicySet>` element contains a set of `<Policy>` or other `<PolicySet>` elements and a specified procedure for combining the results of their evaluation. It is the standard means for combining separate **policies** into a single combined **policy**.

XACML defines a number of Rule and Policy combining algorithms that define a procedure for arriving at an **authorization decision** given the individual results of evaluation of a set of **rule**s or **policie**s, in particular:

- Deny-overrides,
- Permit-overrides,
- First applicable,
- Only-one-applicable.

XACML Policies are bound to subject and resource attributes that are different from their identities. XACML allows multiple subjects and multi-valued attributes. XACML also allows policies based on resource content what means that authorisation decision may be based on content of the requested resource or its status.

Information security **policies** operate upon **attributes** of **subject**s, the **resource** and the **action** to be performed on the **resource** in order to arrive at an **authorization decisio**n. In the process of arriving at the **authorization decisio**n, **attributes** of many different types may have to be compared or computed. XACML includes a number of built-in functions and a method of adding non-standard functions. These functions may be nested to build arbitrarily complex expressions. This is achieved with the <Apply> element. The <Apply> element has an XML attribute called FunctionId that identifies the function to be applied to the contents of the element. Each standard function is defined for specific argument data-type combinations, and its return data-type is also specified.

Figure 14 shows the structure of Rule element. Policy is bound to the Target that is described by Subject, Resource and Action. Policy may contain a number of rules defined by multiple Rule elements.

A rule is the most elementary unit of policy. The main components of a rule are target, condition that are represented by subelements and effect which is included as an attribute of the Rule element.

The <Condition> element is a boolean function over **subject**, **resourc**e, **action** and **environment attributes** or functions of **attribute**s. If the <Condition> element evaluates to "True", then the enclosing <Rule> element is assigned its Effect value. The <Condition> element is of **ApplyType** complex type.

The <Apply> element denotes application of a function to its arguments, thus encoding a function call. The <Apply> element can be applied to any combination of <Apply>, <AttributeValue>, <SubjectAttributeDesignator>, <ResourceAttributeDesignator>, <ActionAttributeDesignator>, <EnvironmentAttributeDesignator> and <AttributeSelector> arguments.

XACML re-uses enumerated list of functions and operations defined in Xpath 2.0 [29] and XQuery 1.0 [30] used in the FunctionId attribute of the <Apply>/<Condition> element. Element Target contains matching specification for the attributes of the Subject, Resource and Action.

Example of the XACML policy is provided in Appendix.

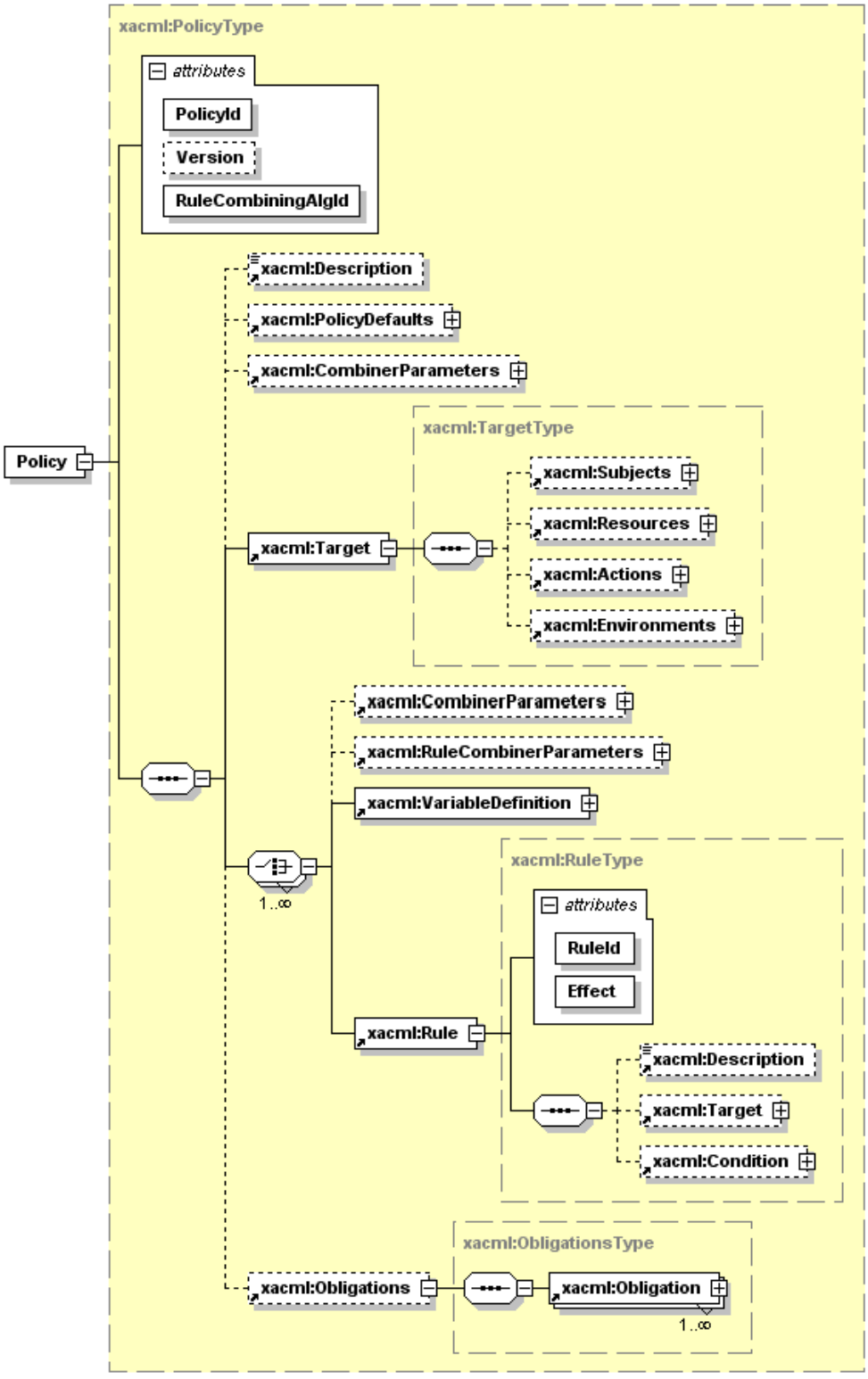

Figure 13. XACML Policy data model.

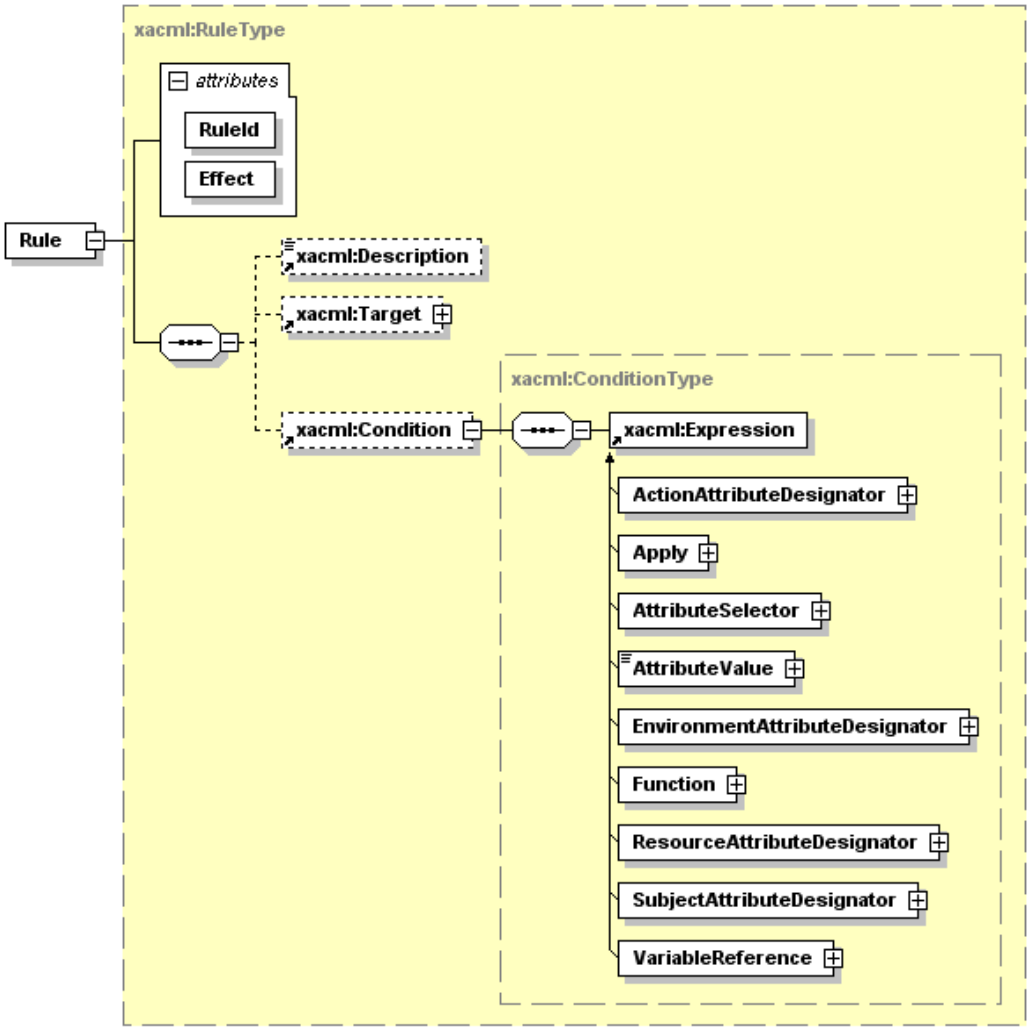

Figure 14. XACML Rule element.

4.3 XACML Request and Response messages

XACML defines format for the Request message that provides context for the policy-based decision. Request may contain multiple Subject elements and multiple attributes of the Subject, Resource and Action.

The request message consists of four mandatory elements Subject, Resource, Action, and Environment that may contain multiple attributes presented as AttributeId – AttributeValue pairs. The Resource attribute may also contain the ResourceContent element. The XACML2.0 speciation requires that all four elements are present but may be empty.

The Environment element provides a possibility to include a security context information such as a SessionId or an authorisation from the previous domain.

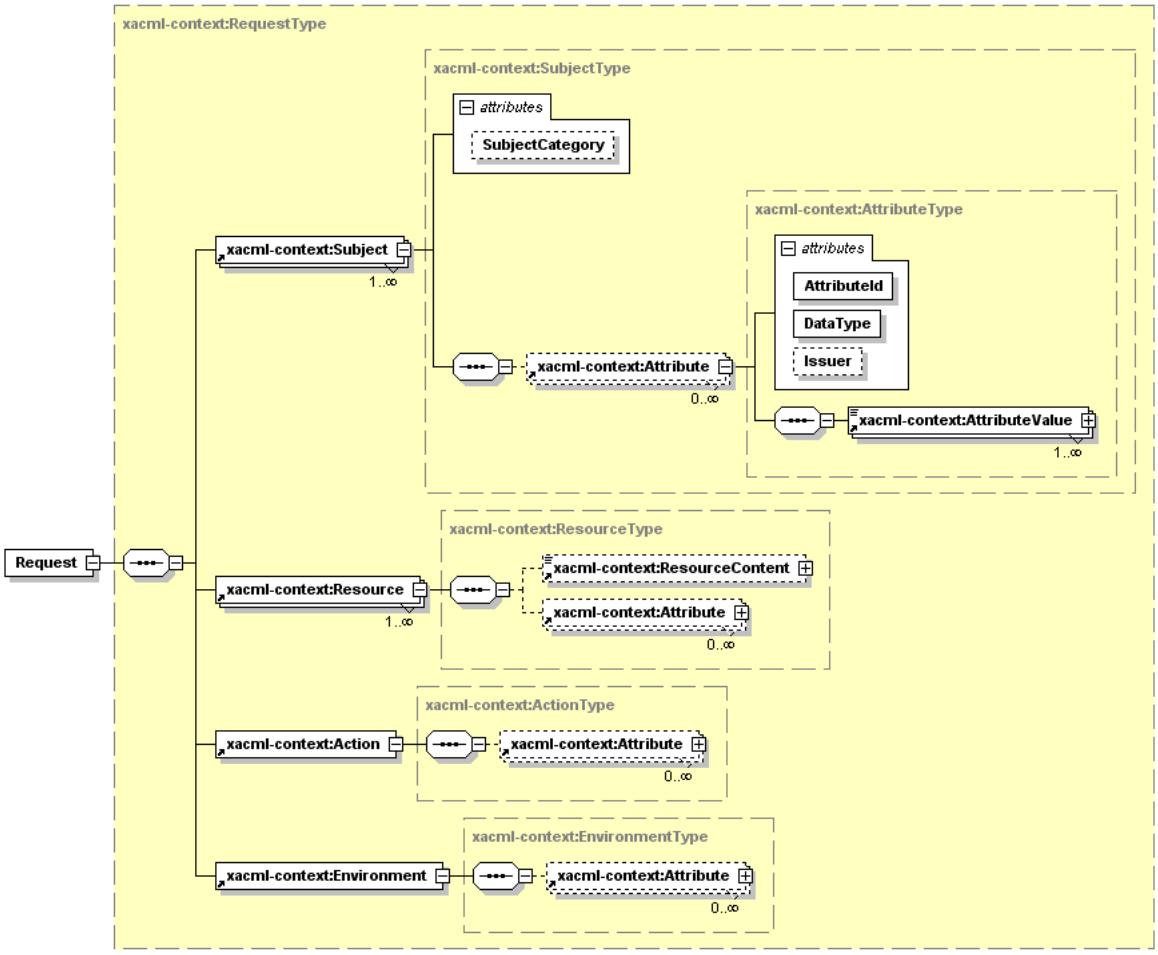

Figure 15. High-level elements of the XACML 2.0 Request.

Response message defined by XACML provides format for conveying Decision (“Deny” or “Permit”) and Status of the decision making process. The Response message format may contain multiple Result elements as defined by the request message and resource policy. The Result element contains a Decision element, which may contain either “Permit” or “Deny” or “Intermediate”. The Status element may contain a simple status code (e.g., “OK”, “request-info”, etc.) and additional status information in the StatusMessage and StatusDetail sub-elements.

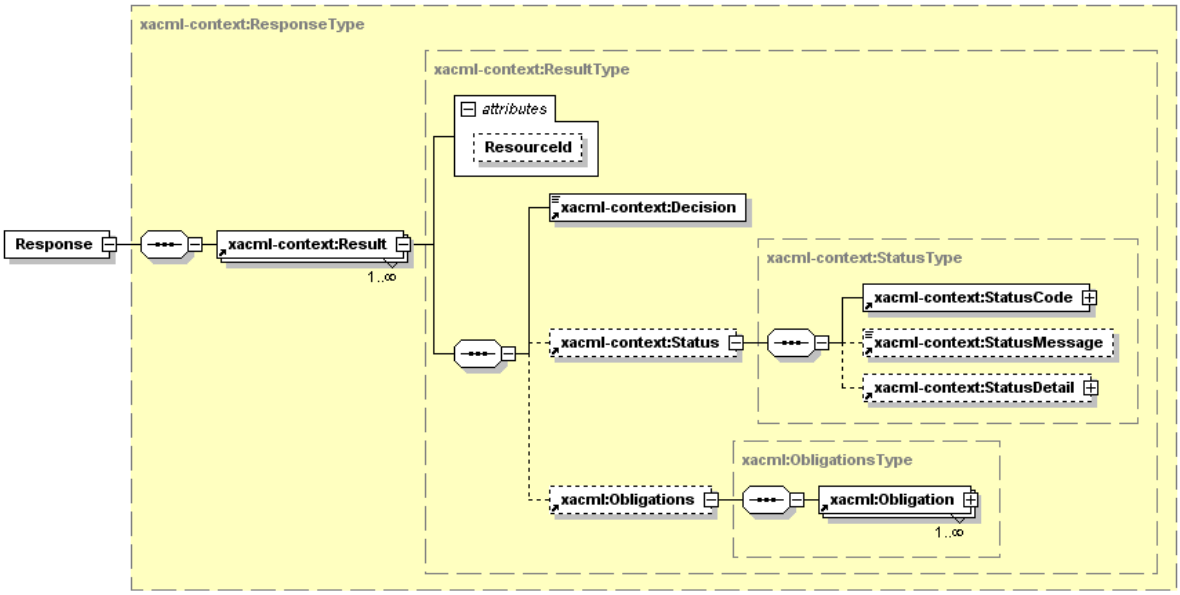

Figure 15. High-level elements of the XACML 2.0 Response.

A request message sent by a user or an application is an XML message sent with SOAP massage over HTTP protocol. Such a request contains information about the Requestor/Subject, Resource and requested Action. The response message contains a Decision result that may be either “Permit” or “Deny” for a final decision (or “Intermediate” for an intermediate communication).

5 SAML2.0 profile of XACML: SAML-XACML protocol and Authorisation assertions format

Although XACML defines XACML Request/Response messages format, it doesn’t provide any suggestions about using one or another transport container or protocol. Using XACML messages directly as authorisation assertions impose some security/integrity problems because they don’t have mechanisms to bind authority (trust) or express/imply security restrictions as they are provided by the such SAML elements as Issuer or Conditions.

SAML2.0 profile of XACML (SAML-XACML) combines well established SAML security assertions format [14] and reach functionality of the XACML policy format [3]. Such a solution provides a good combination between XACML policy expression and evaluation functionality and SAML security assertion management functionality. SAML-XACML profile is supported by the popular Open Source SAML implementation OpenSAML2.

The SAML2.0 profile of XACML defines the queries and assertions to support XACML based AuthZ services.

The XACMLAuthzDecisionQuery and XACMLPolicyQuery provide extension to the SAML protocol. The XACMLAuthzDecisionStatement and XACMLPolicyStatement provide extensions to the SAML assertions.

The XACMLAuthzDecisionQuery is introduced as additional query type for the SAML2.0 protocol. In contrary to the basic SAML2.0 queries, the XACMLAuthzDecisionQuery doesn’t contain the Subject element but used as container for the xacml-context:Request message.

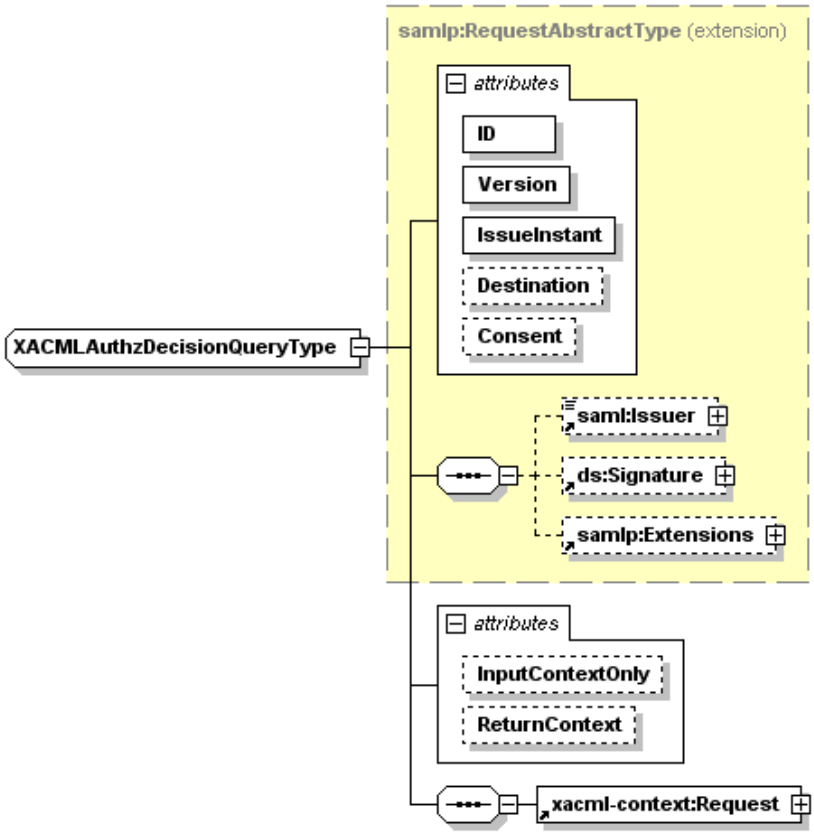

Figure 16. XACML2.0 XACMLAuthzDecisionQuery format.

The XACMLAuthzDecisionStatement provides a container for XACML Request and Response messages that actually hold all necessary information about the authorisation decision in a native XACML format. Figure below illustrates how the XACMLAuthzDecisionStatement is folded into the SAML assertion.

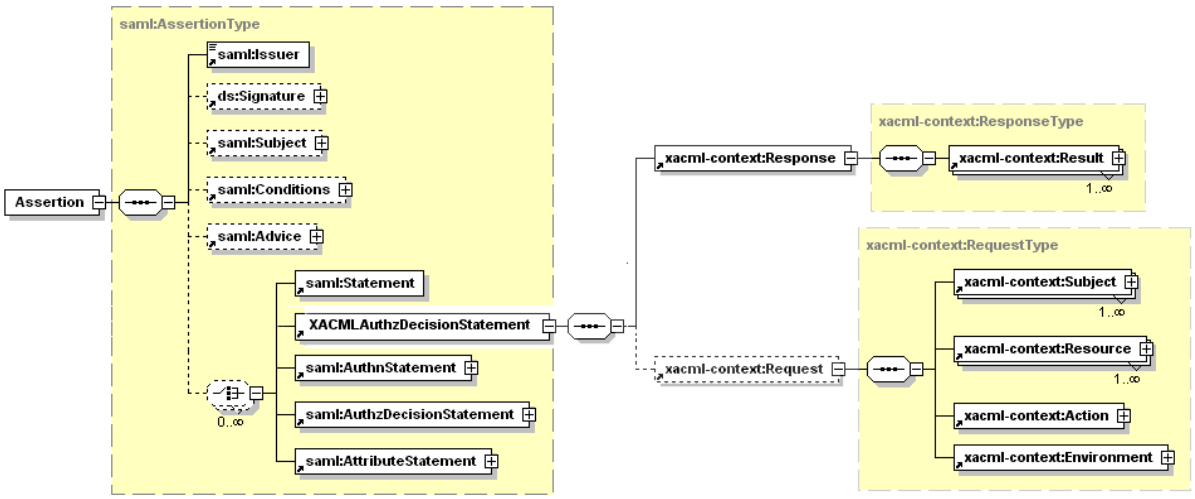

Figure 17. XACML2.0 Assertion containing XACMLAuthzDecisionStatement.

6 Policy Obligations and Obligations handling

6.1 Obligations definition and expression in the XACML policy

The XACML policy can 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.

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.

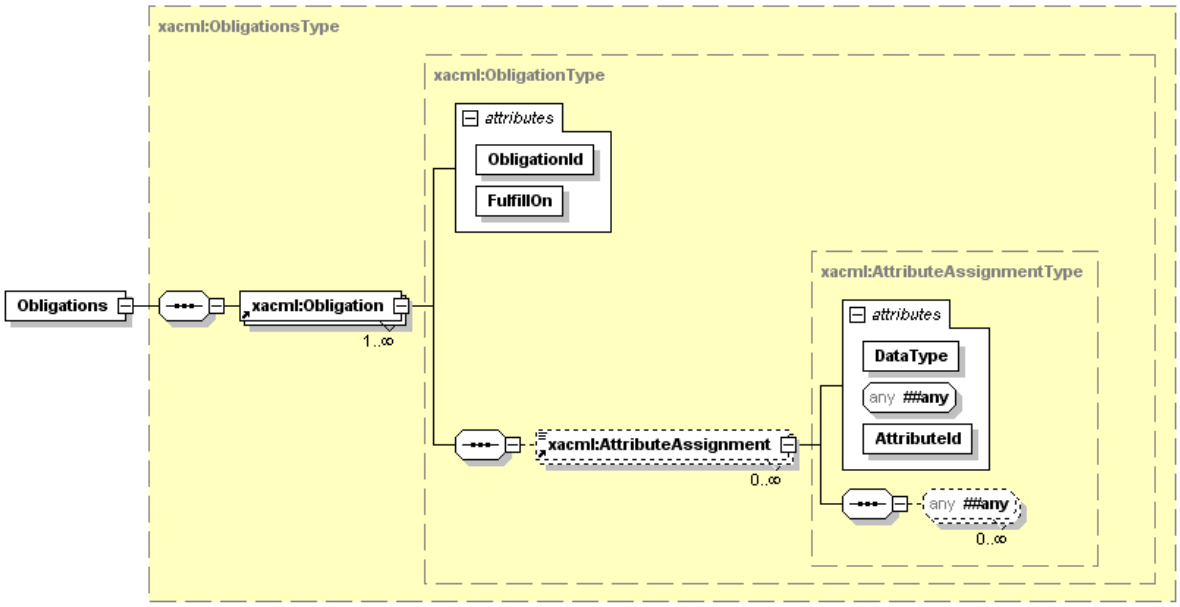

Figure 18. XACML2.0 Obligations element format.

6.2 OHRM Obligation Handling Reference Model (OHRM)

The idea of policy obligations is originated from the works by Sloman [31, 32] and Kudo [33]. The provisional authorisation model was proposed by Kudo and was further implemented in the IBM's XACL (XML Access Control Language) [34]. The provisional AuthZ architecture includes Provisional AuthZ Module (PAM) and Request Execution Module (REM). The provisional AuthZ means that PAM can authorise a request provided the requestor or system (actually REM) will take some security actions, defined as "provisional actions") prior to the request execution. Examples of such actions can be presenting additional credentials, signing privacy statements, logging events, etc.

The Obligations Handling Reference Model (OHRM) was proposed by authors [35] to support typical Grid and network resource provisioning scenarios that require account mapping and conditional authorisation decisions. 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 19 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 [4] 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.

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) [37] to add special Chronicle attribute to the Obligation element in XACML, but this idea has not been further discussed.

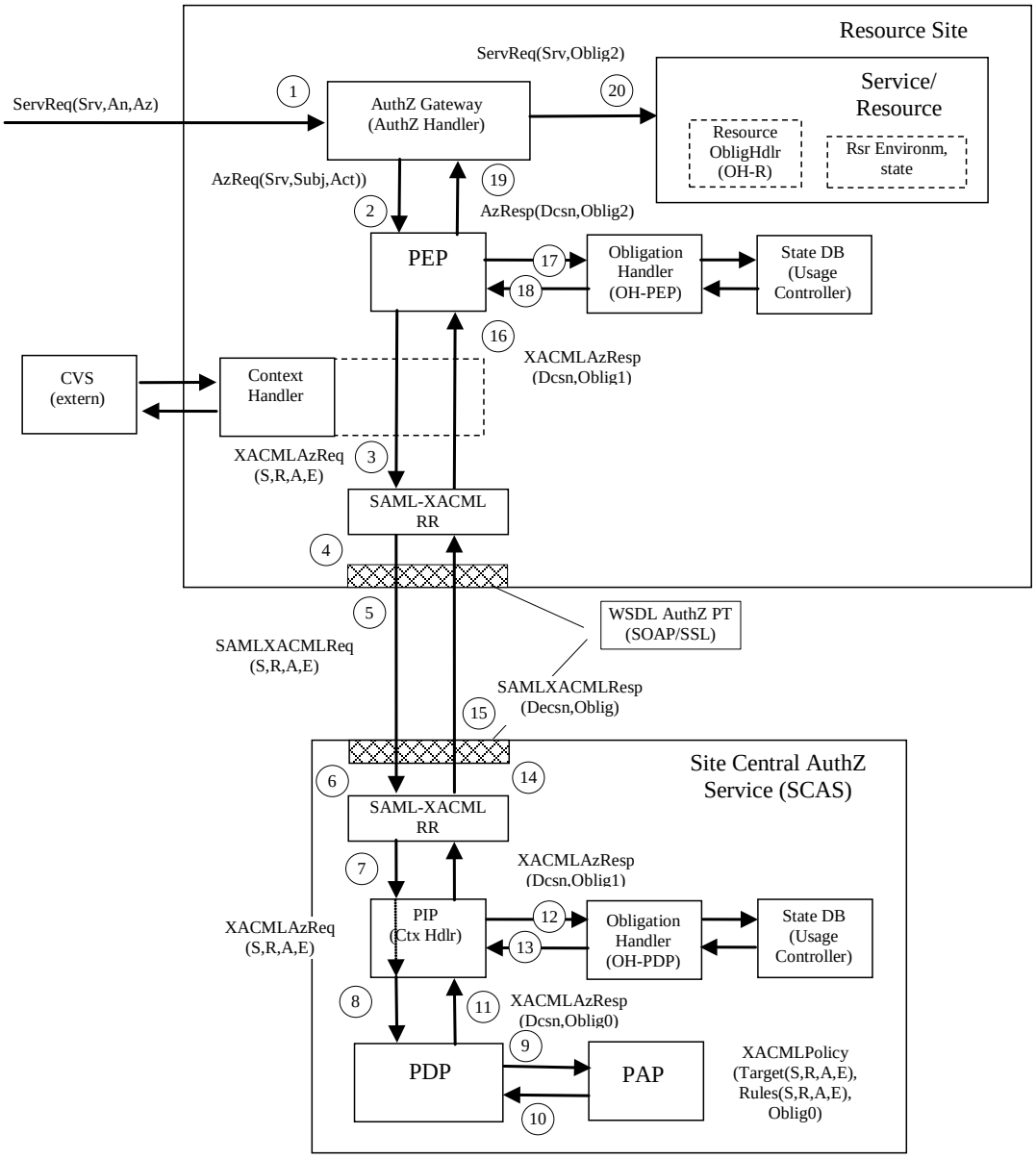

Figure 19. Generic Authorisation dataflow and Obligations handling in distributed AuthZ service.

6.2.1 Policy Obligations example

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 namespace “http://authz-interop.org/xacml” [37]).

```
<!-- 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:attribute: requesting-subject"
DataType="http://www.w3.org/2001/XMLSchema#string">
    &lt;SubjectAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-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>
```

7 XACML-NRP attributes and policy profile for Network Resource Provisioning

This section describes the XACML-NRP attributes and policy profile for Network Resource Provisioning [38] as an example of how application specific profile can be created including a number of practical suggestions about attributes expression and identification. The XACML-NRP profile refers to and uses experience of developing XACML profile for authorisation in Grid [37].

7.1 Use case and requirements

A policy framework and corresponding authorisation infrastructure to support NRP should meet the following requirements:

- Allow for multidomain access control policy definition and interdomain security context management.
- Support topology based policy conditions and rules
- Allow conditional AuthZ decision that should be evaluated in the next domain.
- Use open standards and AuthZ mechanisms that could be easy integrated into the network services.

In particular, the last requirement indirectly implies limitation on separating context and state management between stateless policy definition and PEP-PDP mechanisms to evaluate and manage inter-domains security context and conditional policy decision. In other words, policy should be stateless and security context and state management should be outsourced to a separate context handling functionality that support communications between PEP and PDP intercepting actions related to state information and modifying if necessary..

7.2 Attributes defintion in XACML-NRP

Network or resource related attributes

Network related attributes allow building policy depending on the network topology or other network characteristics.

Topology format should provide necessary information about the network resource to allow consistent policy evaluation, and vise versa the policy format may be defined by the network topology to which the policy is applied. Topology semantics will define the resource attributes semantics, and vice versa.

Network related attributes are considered as a part of the XACML Resource definition. The following resource/network related attributes can be specified and used for authorisation:

- Domain ID (network domain)
- Subdomain (or relationship)
- VLAN
- Node or TNA and TNA prefix, or
- Interface ID
- Device or resource-type
- Link ID
- Link parameters: average delay and maximum bandwidth
- ReservationEPR that may directly or indirectly define the resource federation or security/administrative domain
- Federation that defines a number of domains or nodes sharing common policy and attributes

As it was mentioned before, some more advanced scenarios may require that particular network route or path is provided, in this case the policy definition should allow decision on the specific features of path or on the path in total.

Subject related attributes

Subject related attributes allow building policy depending on the properties of the request Subject or user. The following subject related attributes can be specified:

- Subject ID
- Subject confirmation that contains AuthN assertion/token or other attribute confirming subject's ID by trusted AuthN authority
- Subject Role
- Subject Group
- Subject Federation (e.g., Virtual Organisation, or Shibboleth AAI federation) or domain
- Subject context that can provide additional information about the Subject other than Subject federation e.g. such as Session ID, or project/experiment name

Typically Subject attributes are provided as Subject credentials which depending on user client implementation and middleware may take a form of X.509 public key and attribute certificates (PKC, AC), SAML Authentication and Attribute assertions, proprietary AuthN system credentials.

Action and Environment related attributes

Action related attributes represent a limited number of the specific actions that requesting party can ask to initiate network resource reservation, access or management.

Environment related attributes allow providing additional information for policy definition and evaluation. There is no specific Environment attributes identified for the XACML-NRP profile but this may be a place to put security context related information from the previous domain.

7.3 Policy Obligations used in NRP

Policy obligation is one of the authorisation policy enforcement mechanisms that allows adding AuthZ decision enforcement components that can not be defined in the policy at the moment of making policy decision by the PDP, or may not be known to the PDP or policy administrator/writer

Suggested functionality that can be achieved with using obligations includes but not limited to:

- Intra-domain network/VLAN mapping for cross-domain connections, that can be used to map external/interdomain border links/TNA's to internal VLAN and sub-network
- Network identity and account mapping
- Type of service (or QoS) assigned to a specific request or policy decision
- Quota assignment
- Service combination with implied conditions (e.g., computing and storage resources)
- Usable resources/quota

The need of account mapping may exist in cases when domain based Network Resource Provisioning Systems (NRPS) have pre-installed/built-in pool accounts to which are different types or quality of service are assigned. In such situation authorised user need to use one of such accounts, e.g. "silver", "golden", "platinum". A number of different individual accounts of the same type may be limited, consequently a dynamically assigned account should be selected from the pool of available or free accounts. Such dynamic account assignment can not be specified in the typically stateless policy and cannot be done by PDP. However, the access control policy may contain instruction to PEP to do such mapping.

The following scenarios of enforcing obligations can be considered:

- Obligations are enforced by PEP at the time of receiving obligated AuthZ decision from PDP;
- Obligations are enforced at later time when the requestor accesses the resource or service
- Obligations are enforced before or after the resource or service delivered/accessed/consumed

Such functionality can be supported by ObligationHandlers that can be called either from the PEP or from the SAML-XACML interface modules that handles request/response messages. Although allowing simple solution/implementation, the first method will have problems when enforcing Obligations for later access/use of the reserved service. To allow Obligations enforcement at later time the AuthZ ticket or assertion can be used that contain all necessary information about the AuthZ request/response context. In this case AuthZ ticket must be properly secured with the XML signature and additionally encrypted. AuthZ ticket can use the SAML assertion containing XACMLAuthzDecisionStatement.

7.4 Attributes Expression conventions

This section provides suggestions and examples for the Resource and Subject attributes expression. The Action attributes can use either simple string format or enumerated URN or URL style similar to the Resource attributes. The proposed description is based on the current XACML-NRP and GAAA-TK library implementation in the Phosphorus project [39] and can be used as an example how to create a XACML profile for other application areas or use cases.

Resource attributes

In current implementation the Resource variable in the AuthZ request contains one attribute ResourceURI in the form of URI string that includes the network resource identifier and a list of parameter used for policy-based request evaluation. When sending a XACML Request to XACML PDP the input URI string is converted into the set of the Resource attributes (organised as a HashMap). The attribute names are taken from the XACML-NRP profile, such as “resource-id”, “resource-domain”, “resource-realm”, “resource-type”, “source”, “target”, etc.

The following ResourceURI formats are supported:

```
a) http://testbed.ist-phosphorus.eu/{domain}/{device | service}/{parameters}
```

For example, the following URI will be converted to the set of resource attributes

```
http://testbed.ist-phosphorus.eu/viola/harmony/source=10.7.12.2/target=10.3.17.3
resource-id = http://testbed.ist-phosphorus.eu/viola/harmony
resource-realm = http://testbed.ist-phosphorus.eu
resource-domain = viola
resource-type = harmony
source = 10.7.12.2
target = 10.3.17.3

b) http://testbed.ist-phosphorus.eu/resource-type/{resource-type-name}
```

Subject attributes

The Subject variable of the AuthZ request may contain the following attributes:

a) SubjectId (attribute identifier “subject-id”) – subject identifier in RFC822 (email) or X.521 (LDAP or X.509 Public Key Certificate) formats (must be the same as used in the SubjectConfirmatioData)

Example: WHO740@users.testbed.ist-phosphorus.eu

b) SubjectConfirmatioData (attribute identifier “subject-confdata”) – Authentication assertion or token provided by the trusted AuthN service (can be also SAML AuthN Assertion, X.509 or VOMS attribute certificate), or crypto-string provided local AuthN service.

c) SubjectRole (attribute identifier “subject-role”) – subject role, currently supporting single value.

Example: `admin`, or `researcher@project01`, or `admin@viola.testbed.ist-phosphorus.eu`

d) SubjectContext (attribute identifier “subject-context”) – this attribute is used for providing additional information about a user (and a resource) association like VO, project, experiment/job.

Example: `demo001`; or `VO-Phosphorus`

Potentially this attribute can be extended to provide instant reservation context for dynamically configured AuthZ service.

7.5 Policy identification and policy resolution

When evaluating AuthZ request the ContextHandler or PDP need to find/select an applicable policy. This is typically done based on the request parameters such as Resource or Subject attributes.

The policy selecti0n/finding comprises of two steps: policy resolution and policy retrieval. Policy resolution means extracting such information from the AuthZ request that can be used for further policy selection in the storage/repository. Based on this information, a repository request or query can be constructed to retrieve necessary policy.

Note, it is a SunXACML implementation convention that only one Policy or PolicySet should be supplied to PDP for evaluation, and only one component Policy must be selected if using PolicySet.

The following components of the XACML-NRP profile can be used for policy resolution:

a) resource ID and resource attributes;

b) subject attributes defining context in which the request should evaluated, e.g. project or VO (this information is typically a part of the subject attributes);

c) attributes and policy profile namespace, which can actually be a part of the resource ID if expressed in Fully Qualified Attribute Name format (FQAN format).

Depending on the policy storage/repository implementation, the following components can be used for policy identification:

a) policy file name and directory, if policy is stored as a file;

b) PolicyId attribute of the PolicySet or Policy element;

c) policy Target element that can include any of Subject, Resource, Action, Environment elements.

Although using basically different ways of storing policies, the first case and second identification methods can be based on similar approach to composing PolicyId attribute and (defining) policy file location path. When using third option, the policy repository should be capable to query policy database by the policy Target content.

It is suggested that the PolicyId or PolicySetId is created in the same way using typical for URL/URN style conventions:

```
PolicyId =
 <<url-namespace-prefix/>>testbed.ist-phosphorus.eu/viola/harmony/demo001/policy
PolicyId =
 <<urn-namespace-prefix:>>testbed.ist-phosphorus.eu:viola:harmony:demo001:policy
```

where

`<<namespace-prefix>>-` can be dropped;

```
namespace-prefix = http://authz-interop.org/nrp/xacml
or namespace-prefix = x-urn:authz-interop.org:nrp:xacml
```

Example URL style PolicyId expression:

```
PolicyId = http://authz-interop.org/nrp/xacml/ testbed.ist-
phosphorus.eu/phosphorus/demo001/policy
PolicyId = http://testbed.ist-phosphorus.eu/ viola/harmony/demo001/policy
PolicyId = http://testbed.ist-phosphorus.eu/ phosphorus/demo001/policy
```

8 Libraries and tools supporting SAML and XACML

8.1 OpenSAML Library and extensions

OpenSAML [40] is a set of open source C++ and Java libraries meant to support developers working with SAML. OpenSAML2 supports SAML 1.0, 1.1, and 2.0 specifications.

The OpenSAML framework has a number of extensions and profiles for various application areas and specific use cases developed by various development groups. Some of the extensions such as WS-Addressing, WS-Security, WS-Trust and SAML2 profile of XACML are integrated into the core OpenSAML2 library.

The SAML-XACML profile implementation in Globus Toolkit [41] and GAAA Toolkit (GAAA-TK) [42] uses OpenSAML2 library. Globus Toolkit implementation provides simple Obligations handling functionality as described in the XACML-Grid profile [38]. GAAA-TK implements OHRM that provides common Obligations handling model and flexibility for distributed authorisation infrastructure. Both implementations allow 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.

8.2 Sun’s XACML Java Library

Sun's XACML implementation [43] provides a reference Open Source XACML implementation and the most widely used. The library is also used as a basis for an ongoing Sun's project on Internet Authorization by the Internet Security Research Group.

The project provides complete support for all the mandatory features of XACML as well as a number of optional features. Specifically, there is full support for parsing both policy and request/response documents, determining applicability of policies, and evaluating requests against policies. All of the standard attribute types, functions, and combining algorithms are supported, and there are APIs for adding new functionality as needed. The library provides also APIs for writing new retrieval mechanisms used for finding things like policies and attributes.

Various external development projects and initiatives provide different XACML attributes and policy extension to support application specific attribute sets and policy models. The two mentioned above XACML-Grid and XACML-NRP profiles provide good examples.

9 References

[1] OASIS Reference Model for Service Oriented Architecture 1.0, Official Committee Specification, Aug. 2, 2006. [Online]. Available: http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

[2] Web Services Architecture. W3C Working Draft 8, August 2003. [Online]. Available: http://www.w3.org/TR/ws-arch/

[3] *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

[4] *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

[5] "Assessment of Access Control Systems", by Vincent C. Hu, David F.Ferraiolo, D. Rick Kuhn. Interagency Report 7316. [Online] Available: http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf

[6] Samarati, P., S.C. de Vimercati, Access Control: Policies, Models, and Mechanisms, in book "Foundations of Security Analysis and Design", LNCS, Springer Berlin/Heidelberg, 2001, Pages 137-196

[7] Sandhu, R. & Samarati, P., 1994. “Access Control: Principles and Practice”, IEEE Communication Magazine, September 1994, pp. 40-48.

[8] Sandhu, R., Coyne, E. J., Feinstein, H. L., Youman, C.E. 1996, "Role-Based Access Control Models", IEEE Computer, February 1996, pp. 38-47.

[9] Information Technology - Role Based Access Control, Document Number: ANSI/INCITS 359-2004, InterNational Committee for Information Technology Standards, 3 February 2004, 56 p.

[10] ISO/IEC 10181-3:1996 Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Access control framework. – Available in “OSG Authorisation API”. - http://www.opengroup.org/online-pubs?DOC=9690999199&FORM=PDF

[11] 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

[12] RFC2903 Laat de, C., G. Gross, L. Gommans, J. Vollbrecht, D. Spence, "Generic AAA Architecture,” Experimental RFC 2903, Internet Engineering Task Force, August 2000. ftp://ftp.isi.edu/in-notes/rfc2903.txt

[13] 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

[14] SAML 2.0 Profile of XACML 2.0, Version 2.0. OASIS Standard, 1February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

[15] Security Assertion Markup Language (SAML) 2.0 Technical Overview, Working Draft 21, 21 February 2007. [Online]. Available from: http://www.oasis-open.org/committees/download.php/22553/sstc-saml-tech-overview-2%200-draft-13.pdf

[16] Shibboleth Attribute Authority Service. [Online]. Available from: http://shibboleth.internet2.edu/

[17] The Liberty Alliance Project. [Online]. Available from: http://www.projectliberty.org/

[18] Liberty Alliance ID-WSF 1.1 Specifications. [Online]. Available from: http://www.projectliberty.org/resource_center/specifications/liberty_alliance_id_wsf_1_1_specifications

[19] Web Services Security: SAML Token Profile 1.1, OASIS Standard, 1 February 2006. [Online]. Available from: http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[20] *Core and hierarchical role based access control (RBAC) profile of XACML v2.0*, OASIS Standard, 1 February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf

[21] Multiple resource profile of XACML 2.0, OASIS Standard, 1 February 2005, available from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-mult-profile-spec-os.pdf

[22] Hierarchical resource profile of XACML 2.0, OASIS Standard, 1 February 2005, available from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-hier-profile-spec-os.pdf

[23] Privacy policy profile of XACML v2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-privacy_profile-spec-os.pdf

[24] XML Digital Signature profile of XACML v2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-dsig-profile-spec-os.pdf

[25] Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare Version 1.0, Public Review Draft 02, 29-Apr-09. http://www.oasis-open.org/committees/download.php/32444/xspa-xacml-profile-pr02dad_v3.doc

[26] eXtensible Access Control Markup Language (XACML) Version 3.0, CD-1, 16-Apr-09. http://www.oasis-open.org/committees/download.php/32425/XACML-3.0-cd-1-updated-2009-May-07.zip

[27] XACML v3.0 Administration and Delegation Profile Version 1.0, CD-1, 16-Apr-09. http://www.oasis-open.org/committees/download.php/32425/XACML-3.0-cd-1-updated-2009-May-07.zip

[28] XACML PDP Metadata Version 1.0, OASIS Working Draft, 24 February 2008. http://www.oasis-open.org/committees/download.php/27316/xacml-3.0-metadata-v1-wd-01.zip

[29] XQuery 1.0 and XPath 2.0 Functions and Operators W3C Recommendation, 23 Jan. 2007. See http://www.w3.org/TR/xpath-functions/.

[30] XQuery 1.0: An XML Query Language. W3C Recommendation 23 January 2007. http://www.w3.org/TR/xquery/

[31] Sloman, M. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.

[32] Bettini C., S. Jajodia, X. S. Wang, D. Wijesekera, "Provisions and Obligations in Policy Management and Security Applications", Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[33] Kudo M and Hada S, XML document security based on provisional authorization, Proceedings of the Seventh ACM Conference on Computer and Communications Security, Nov 2000, Athens, Greece, pp 87-96.

[34] XML Access Control Language (XACL)- http://xml.coverpages.org/xacl.html

[35] Demchenko, Y., C. de Laat, O. Koeroo, H. Sagehaug, Extending XACML Authorisation Model to Support Policy Obligations Handling in Distributed Applications, Proceedings of the 6th International Workshop on Middleware for Grid Computing (MGC 2008), December 1, 2008, Leuven, Belgium. ISBN:978-1-60558-365-5.

[36] Demchenko, Y., C. M. Cristea, de Laat, XACML Policy profile for multidomain Network Resource Provisioning and supporting Authorisation Infrastructure, IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY 2009), July 20-22, 2009, London, UK.

[37] Open Grid Forum OGSA Authorisation Working Group. [Online] Available: https://forge.gridforum.org/projects/ogsa-authz

[38] An XACML Attribute and Obligation Profile for Authorization Interoperability in Grids. [Online] Available: https://edms.cern.ch/document/929867/1

[39] PHOSPHORUS Project. [Online] Available: http://www.ist-phosphorus.eu/

[40] OpenSAML library. [Online] Available: https://spaces.internet2.edu/display/OpenSAML/Home/

[41] Globus Toolkit. [Online] Available: http://www.globus.org/toolkit/

[42] Generic AAA Toolkit pluggable Java library. [Online] Available: http://www.phosphorus.pl/software.php?id=gaaa_tk

[43] Sun's XACML Implementation. [Online] Available: http://sunxacml.sourceforge.net/

Appendix A. Examples XACML Policy and Request/Response Messages

The examples in this section illustrate how Obligations can be expressed in the XACML policy format models an idea to communicate PEP Obligations handling capability to the PDP in the Environment element. However, to make it possible to select the applicable policy based on returned Obligations, we need to put explicit values of the ObligationId's into the policy Environment matching expression.

a) Policy example that permits access to the resource "VO-EGEE/CE01" for subjects that have Virtual Organisation EGEE (VO-EGEE) membership provided as "subject-vo" Subject attribute. The policy also imply obligation to map-the Subject identity "subject-id" attribute to the pool account defined by UID and GUD.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
      xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
      xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
      xmlns:md="http://www.medico.com/schemas/record"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
        access_control-xacml-2.0-policy-schema-os.xsd"
      PolicyId="urn:oasis:names:tc:xacml:2.0:policy:example841:policy"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
  <Description>Example - Policy: Obligation ID negotiation - as PEP type in the
Resource attribute</Description>
  <PolicyDefaults>
    <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
  </PolicyDefaults>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        VO-EGEE</AttributeValue>
        <SubjectAttributeDesignator
SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject:-category:access-subject"
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-vo"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">http://nikhef.nl/VO-EGEE/CE01
        </AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#URI"/>
       </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">GLite-
CE</AttributeValue>
```

```
       <ResourceAttributeDesignator AttributeId="urn:some:path:peptype"
DataType="http://www.w3.org/2001/XMLSchema#String"/>
</ResourceMatch>
      </Resource>
    </Resources>
    </Target>
    <Rule RuleId="urn:oasis:names:tc:xacml:2.0:policy:example841:rule"
Effect="Permit">
      <Description>
        User with role "researcher"  from "VO-EGEE" can access Resource
"http://nikhef.nl/VO-EGEE/CE01".
      </Description>
      <Target>
      <Actions>
          <Action>
            <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
               <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">SubmitJob</AttributeValue>
               <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
      <Condition>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
           <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">researcher</AttributeValue>
        <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-role"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="EGEEAttributeIssuer"/>
        </Apply>
       </Condition>
    </Rule>
    <Obligations>
      <Obligation ObligationId="urn:oasis:names:tc:xacml:2.0:scas-
policy:example007:policy:obligation.UID" FulfillOn="Permit">
        <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:1.0:example:attribute:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string">
             &lt;SubjectAttributeDesignator
               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
               DataType="http://www.w3.org/2001/XMLSchema#string"/&gt;
          </AttributeAssignment>
<!-- This is actual account attribute/name to which it should be mapped -->
          <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:poolaccount"
DataType="http://www.w3.org/2001/XMLSchema#string">
            &lt;PoolAccountDesignator

AttributeId="http://glite.egee.org/JRA1/Authz/XACML/obligation/poolaccount”
              DataType="http://www.w3.org/2001/XMLSchema#string"/&gt;
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">egee-
pool-next-available</AttributeValue>
       </AttributeAssignment>
     </Obligation>
   <Obligation ObligationId="urn:oasis:names:tc:xacml:2.0:scas-
policy:example841:policy:obligation.GID" FulfillOn="Permit">
```

```
     <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:1.0:policy:subject:subject-group"
DataType="http://www.w3.org/2001/XMLSchema#string">
      GID-researchers
     </AttributeAssignment>
   </Obligation>
  </Obligations>
</Policy>
```

b) Request that contains additional Resource/Attribute information “peptype” that indicates specific PEP type/functionality.

```
<?xml version="1.0" encoding="UTF-8"?>
<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
        access_control-xacml-2.0-context-schema-os.xsd">
  <!-- Example - Request: Supported Obligations negotiation - PEP type is included
in the Resource attribute -->
  <Subject>
    <Attribute
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="http://www.w3.org/2001/XMLSchema#string">
       <AttributeValue>Wim Huizinga</AttributeValue>
    </Attribute>
    <Attribute
          AttributeId="urn:oasis:names:tc:xacml:2.0:subject:subject-vo"
          DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>VO-EGEE</AttributeValue>
    </Attribute>
    <Attribute
          AttributeId="urn:oasis:names:tc:xacml:2.0:subject:subject-role"
          DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>researcher</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute
         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
         DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>http://nikhef.nl/VO-EGEE/CE01</AttributeValue>
    </Attribute>
    <Attribute
         AttributeId="urn:some:path:peptype"
         DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>GLite-CE</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>SubmitJob</AttributeValue>
    </Attribute>
  </Action>
  <Environment/>
```

```
</Request>
```

c) Response message that contains obligations to map to a pool account that must be enforced by PEP or resource itself.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os access_control-
xacml-2.0-context-schema-os.xsd">
  <Result ResourceId=" http://nikhef.nl/VO-EGEE/CE01">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
    <xacml:Obligations>
      <xacml:Obligation ObligationId="urn:oasis:names:tc:xacml:2.0:scas-
policy:example007:policy:obligation.UID" FulfillOn="Permit">
        <xacml:AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:1.0:example:attribute:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string">
            &lt;SubjectAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/&gt;
          </xacml:AttributeAssignment>
<!-- This is actual account attribute/name to which it should be mapped -->
          <xacml:AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:poolaccount"
DataType="http://www.w3.org/2001/XMLSchema#string">
           &lt;PoolAccountDesignator

AttributeId="http://glite.egee.org/JRA1/Authz/XACML/obligation/poolaccount”
             DataType="http://www.w3.org/2001/XMLSchema#string"/&gt;
         <xacml:AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">egee-
pool01</xacml:AttributeValue>
       </xacml:AttributeAssignment>
     </xacml:Obligation>
   <xacml:Obligation ObligationId="urn:oasis:names:tc:xacml:2.0:scas-
policy:example007:policy:obligation.GID" FulfillOn="Permit">
     <xacml:AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:1.0:policy:subject:subject-group"
DataType="http://www.w3.org/2001/XMLSchema#string">GID-
researchers</xacml:AttributeAssignment>
   </xacml:Obligation>
  </xacml:Obligations>
  </Result>
</Response>
```