

# Obligation Standardization

---

*David Chadwick, University of Kent, Mario Lischka, NEC Europe Ltd*

The OASIS XACML standard has become a recognized standard for the specification of access control policies, as well as a generic framework for access control. Whilst the XACML policy language is very flexible for specifying access policies, its current handling of obligations is very basic. The XACML standard (both v2 and v3) simply defines an obligation as an attribute assignment, thereby leaving the definitions of the syntax and semantics of the obligations to the application writers. We need to go much further than this. There is some very early work on so called "Obligation Families" within the XACML TC, which attempts to define additional semantics and processing of obligations, but this work has not progressed and is currently stalled. We have identified the following deficiencies in the current obligation specification:

- there is no overall conceptual model for obligations and their enforcement,
- there is currently no method to specify the exact timing between the actual user access and the enforcement of the obligation
- there is currently no way to specify the actual obligation language exchanged between a Policy Enforcement Point (PEP) and a Policy Decision Point (PDP)
- the location of the obligation enforcement can not be specified. In a distributed environment with multiple PDPs, the obligation policy needs to indicate which actor should enforce an obligation,
- there is no way of identifying or addressing conflicts between obligations.

In this paper we will present an outline solution for all five problems as extensions to XACML, or in the case of obligation language specification as an XML schema of its own which could be included in the regular XACML specification.

## Overview

1. We need a full conceptual model with standardised parameters in the XACML protocol for all the conceptual entities. We identify the following entities:

- the Subject which is to perform the obligation
- the Action which is to be performed
- the Target of the obligation, which can be different to the Subject e.g. a user (subject) must delete a file (target); the PEP (subject) must send an email to the Security Officer (target).
- Constraints on when the obligation is to be performed e.g. in two weeks time, or when event X has occurred.

- Exception or other secondary action to be taken if the obligation cannot be enforced.
2. We need a temporal positioning of the obligation with respect to the user’s action i.e.
- Before, With or After the user’s action. The outline for this was presented in [1]. The simple addition of a temporal attribute to an obligation element can indicate its temporal positioning.
3. We propose the dynamic obligation specification language (XOML) in the next section, which covers the following aspects. First, it allows us to define the actual obligation and its parameters including the relationship between obligations, especially conflicts between them. Second, potential conflicts are detected and partially solved at runtime based on the definition. Finally, we show how the introduced extensible obligation markup language (XOML) could be integrated into the XACML standard.

## Obligation Specification Language

In order to provide a dynamic obligation infrastructure, a generic specification language is required, which enables us to define the actual obligation and its parameters including their types. In addition, the relationship with other obligations, most importantly potential conflicts with them, should also be expressed through this generic specification language. This language is then used to define the concrete obligations used by a Policy Decision Point (PDP) and Policy Enforcement Point (PEP) while the actual policy and the response received to an access request will refer to this definition. It is assumed that both sides (PEP and PDP) agree on the set of obligations which can be enforced during initial configuration and set up.

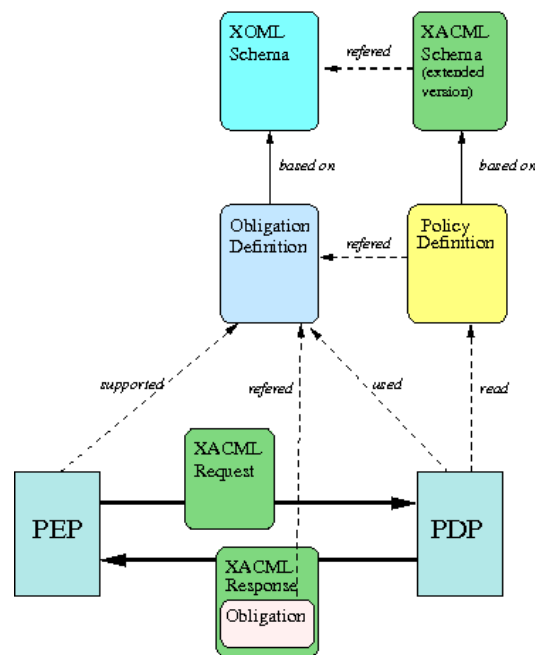


Figure 1: XOML Specification and Utilization

Figure 1 shows an overview of the usage of the specification. The policy used by the PDP refers to the potential obligation. This reference information is also contained in the response. Based on this reference

in the response, the PEP can easily verify if the contained obligation is part of the set supported according to the initial negotiation with the PDP. According to the specification of the supported obligations, the PDP and PEP are enabled to check for potential conflicts in the obligations provided as the result of a specific access request.

For more complex policies these obligation could be the result of the combination of different policies. As the combination algorithms specified in XACMLv3 do not consider any conflicts, our approach at least provides the potential for detecting them. In addition, we will identify those cases which allow automatic resolution strategies to be applied.

## Specification Language

While actual references to recompiled obligations are used in previous approaches such as PONDER , a language independent approach is presented in this paper. As XML has become the de-facto standard for such specifications, we choose to align our definition of a generic obligation specification with XACMLv3.

Thus the name and the parameters including their data types are specified independently of the implementation details in the PEP. A new XML schema called *extensible obligation markup language (XOML)* defines the details of the obligation specification. A concrete instantiation specifying a set of obligations based on this schema could be exchanged easily for negotiation purposes.

As shown in Figure 1, the related specification is realized as an extension of the existing XACML standard, and the Obligation Schema and Policy Schema are kept separated<sup>1</sup>. The concrete obligation definition is an independent part, and any policy definition has to refer to the obligation specification it is using. Each definition of the obligation contains a unique identifier for the obligation itself and a (potentially empty) list of parameters of the obligation, each with a locally unique name and the data type.

In order to detect actual conflicts in a set of obligations contained in the XACML response, it is necessary to define these conflicts in detail beforehand. As a first observation, these conflicts do not depend solely on the obligation function, but additionally on the current parameter values. Thus, depending on these parameter values, there might be no conflict or the type of conflict might differ. Regarding these types of conflicts or more generally speaking the **relations between obligations**, we identified the following:

- **Conflict:** two obligations are in opposition to each other (e.g. get a write lock on the same object for two different subjects).
- **Contradiction:** one obligation would annul the effect of the other, when executed in a specific order (e.g. acquire and release a lock)
- **Inclusion:** the objective of one obligation is included in the other e.g. get a read/write lock implies a read lock as well

---

<sup>1</sup> A combined specification is possible as well. An alternative is an independent Policy Schema which is not extending any previous definitions

- **Subordinated, Superordinated:** if executed in a specific order the two obligations could be executed in a meaningful way (e.g. releasing a lock on a specific object for one subject and then assigning it to another subject).

These conflicts arise through the combination of policy results. Thus instead of a static solution it would be more appropriate to investigate the policy combination algorithm further. But this is not the scope of this paper. Instead, we present some resolution algorithms. The first two relations could be resolved by assigning priorities to the obligations, which might not be appropriate for all application scenarios. The other two allow us to describe a (natural) solution of the problem.

As indicated above the relation between the obligations also depends on the actual value of the parameters. For easier detection of conflicts at run-time the following three matching categories of the parameters are introduced.

- **Direct:** the obligations are directly related independent of current parameter values
- **Partially:** depending on some parameter values of the obligation matching with the values of the same parameter in another obligation (e.g. two **readlock** obligations with the object, but different subjects).
- **Full:** all parameters values of the obligation have to match with their corresponding parameters of another obligation (e.g. **readlock** and **readunlock** whose parameter subject and object have the same values, respectively)

It is important to notice that there might be more than one relation between two obligations depending on the matching of the parameters.

## Run-Time

The PDP ensures during the loading of policy specification(s) that only obligations given in the definition are contained. The PEP checks for each access response whether it contains only those obligations that have been negotiated. The compatibility of the obligations including their parameters is checked based on the supported obligation specification. Independently, the PEP and PDP could generate a relationship model of all obligations contained in the obligation definition. Based on a conflict a unidirectional relationship is created and the relevant parameters together with the resulting conflicts are given. For a given reply the contained obligations are checked for a relationship based on the concrete parameter values. If one of the relationships **inclusion**, **subordinated** or **superordinated** is detected a straightforward resolution is applied. In the first case the PEP ignores the included obligation. In the second and/or third case an ordering of the execution takes place. In case of a **conflict** or **contradiction** the problem has to be escalated to the issuer of the related policies for further handling.

## Conclusion

With the aspects presented in this paper, XACML could be brought forward for a more robust and comprehensive infrastructure for the enforcement of obligations especially in distributed environments.

## References

[1] David W Chadwick, Linying Su, Romain Laborde. "Coordinating Access Control in Grid Services". *Concurrency and Computation: Practice and Experience*, Volume 20, Issue 9, Pages 1071-1094, 25 June 2008. Online version available from <http://www3.interscience.wiley.com/cgi-bin/abstract/117347062/ABSTRACT>