

# Towards Standardization of Distributed Access Control

---

**Mario Lischka,  
Yukiko Endo,**

NEC Laboratories Europe

NEC Europe Ltd.

Heidelberg Germany

**Elena Torroglosa, Alejandro Pérez,  
Antonio G. Skarmeta**

Department of Information and Communications Engineering

University of Murcia

Murcia, Spain

SWIFT (Secure Widespread Identities for Federated Telecommunications) leverages identity technology as a key to integrate service and transport infrastructures for the benefit of users and the providers. It focuses on extending identity functions and federation to the network while addressing usability and privacy concerns [4][7]. In order to address these issues various types of policies have been identified in SWIFT

- **Privacy Release Policies.** Represent how to disclose privacy information based on who is requesting the information and how is going to use it. These policies applies to virtual identities, credentials, tokens, attributes, ...
- **Identity Management Policies.** Control how the identity management functions are performed, like the virtual identity lifecycle, credential lifecycle, how to export and import user's information, etc.
- **Business Agreement Policies.** Control the establishment of agreements at different levels (Organization, federation, etc.).
- **Resource Access Policies.** Indicate when to grant or deny access to protected resources, depending on the user information (VID, credentials, attributes...), target resource being accessed, action to be performed with the resource and environment variables.
- **Delegation Policies.** Indicate how users delegate privileges to other users, whatever this privileges are identity information (in form of attributes) or service access rights.

As the identity of a user is managed at various places (e.g. Attribute Provider) these policies are interrelated and the decision has to be coordinated. A single policy could not cope with the various aspects, such as integrating the access to a local access point based on the service level available to the requesting user. Therefore, we propose that an entity must be able to deduce its decision from arbitrary results of other entities. These entities are called *Authoritative Domains* and have specified a set of policies. The related evaluations are done at a dedicated policy decision point, which could be queried by other PDPs to deduce decisions. An important aspect is that the subject and the object of the derived requests may be changed according to the specification of a policy. In order to achieve this, we have developed a policy language which supports deduction of decisions, together with a distributed definition of the policies. This policy language is defined as an extension of the current draft of XACML 3.0 [6]

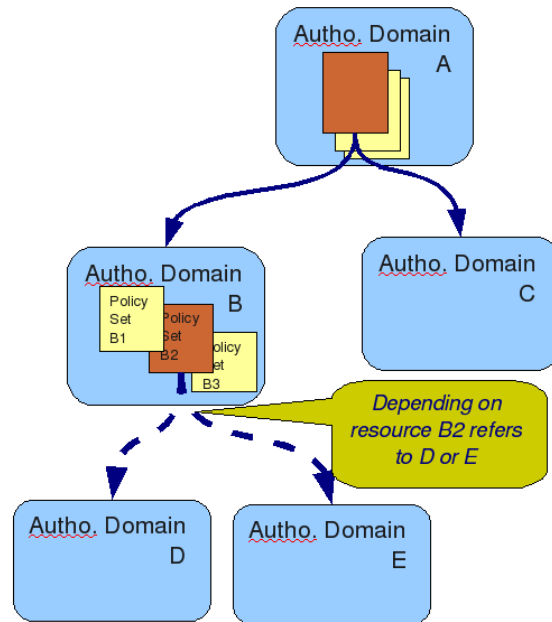
Anyway XACML is widely spread to specify access control policies, various tools are available to specify and model these policies<sup>1</sup>. Other approaches in this area are not applied to a standard [1][2][3].

## General Concept of Distributed Policies

The key idea of distributed policies is to combine the policy decision of another Authoritative Domain into those of the local domain. This will provide us

- **Abstraction:** details about other policy of other domains are not required
- **Independent:** definition of policies is done by the Authoritative Domain
- **Adaptive:** policies support dynamic references to other authoritative domains
- **Bridging:** translation of local attribute names and value space into those of referred ones
- **Transparency:** location of the referred domain with respect to end-points is not explicitly required inside a policy
- **Confidentiality:** internal details on the rules and the attributes leading to the decision can be kept confidential

Due to the proposed extensions to XACML other policies could refer to this set of policies without having to know the details of the policies given inside. As each of them might have their own notion of subject, resources and action an important aspect is the mapping of the latter between different sets of policies.



**Figure 1: Concept of Hierarchical Deduction**

Therefore, we must be able to transform the given request in a particular environment of domain A into one with a (slightly) different context of another authoritative domain B. Depending on the results of the domain B the result of domain A is deduced as sketched in Figure 1. It is important to notice that the domain might not be fixed, but instead be dynamically determined e.g. based on attributes of the original

<sup>1</sup> e.g. the XACML editor of University of Murcia (<http://sourceforge.net/projects/umu-xacmleditor/>)

request. An example is indicated in Figure 1, where policy set B2 is either deducting its decision on the results of the authoritative domain D or E, depending on the resource of the request to B. In the same way several domains may give a result on some request, but do not provide a result on others.

To overcome problems on decidability one of the requirements of our language will be that a deduction is only taken from a subordinated domain. These subordinated domains have to be explicitly defined.

## Concept and Architecture for Distributed Policies

A generic framework for access control is specified in XACML 3 [6]. Since it does not contain any major changes to this framework, one could argue that for a distributed decision request the context handler could query different (including non local) Policy Decision Points (PDP) for a single request. In addition, attributes from other Domains could be queried in the distributed policies in a similar manner to decisions. Integrating this with XPath expressions is already known in XACML, and resolving them with a special Policy Information Point (PIP) might be one solution, but would require additional assumptions on the XPath expressions. These assumptions would also lead to the changes how XPath expressions are handled and might raise inconsistencies in existing implementations.

Our solution implies that an entity must be able to compute its decision from results obtained from other entities. These entities are in domains which have specific sets of policies. A policy decision can therefore be divided into several queries which are evaluated at different PDPs. The process is recursive in that a remote policy can also contain references to remote decisions. An important aspect is that the subject and the objects of the remote requests can be modified according to a policy. For this purpose, we have developed a policy language which supports deduction of decisions, together with a distributed definition of policies. Our policy language is defined as an extension of the current draft of XACML 3.0, but can also be used together with XACML 2.0 with some minor modifications.

In order to have a clean separation between the proposed extensions to the XACML language and the framework, we have outlined two Figure 2, from a necessary deployment point of view. The new elements are the **Distributed Policy Information Point (DPIP)** and the **Distributed Policy Decision Point (DPDP)**. The DPIP is resolving attributes at different domains each time an element of *ReferredDesignator* or *ReferredSelector* is evaluated, the latter DPDP queries decision in case a *ReferredPolicy* has to be evaluated. In either case if one of these new elements is found during the evaluation of a policy, the proposed handling is in accordance with the one of an unknown attribute. Thus the PDP replies to the Context Handler (CH) with a XACML Context response including the "NeedMoreInfo" decision, specifying the required information. The "NeedMoreInfo" is a new response type (additionally to Permit, Deny, Indeterminate and NotApplicable). The required information is the target domain and either the attribute that should be deduced or the respective decision. The PDP maintains the status of the evaluation process to complete it after the CH provides the required information. Attributes and variables from the deduction are translated to the proper values in the XACML Context Response.

Based on the response from the PDP, the CH sends an authorization decision request to the DPDP respectively or an attribute query to the DPIP. The DPIP or DPDP discovers its proper corresponding contact point in the other domain, and sends the request. In case of an attribute query this might be another Identity Aggregator any other kind of Attribute service. The request for a distributed decision is sent to another Identity Management Platform<sup>2</sup> where the query is feed to the related context handler.

An authorization decision could be based on another one taken in a different domain. Furthermore, there are specific situations where additional information is needed to take the authorization decision. But the deduction process cannot be based only on the authorization decision from the other domain. Additionally some attributes are required from other domains, too. Therefore, the authorization decision received from

---

<sup>2</sup> In case of a legacy system the request has to be translated at the DPDP before it is send.

the target domain should include these distributed attributes. This information depends specifically on the related authorization decision and, therefore, it cannot be recovered in the usual way, before requesting the PDP. The use of distributed attributes implies that an agreement should be established previously among the implied domains in order to define the set of attributes that can be deduced. Thus each domain should be able to obtain and to return these attributes if necessary. Similar to the reference to other domains in order to get decisions on access request, we have to make sure that the no circular dependencies are occurring.

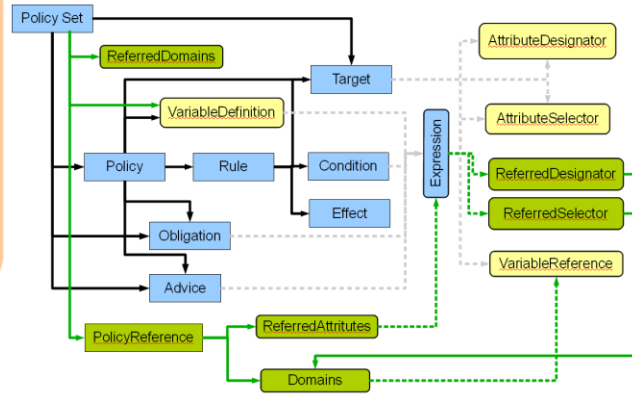
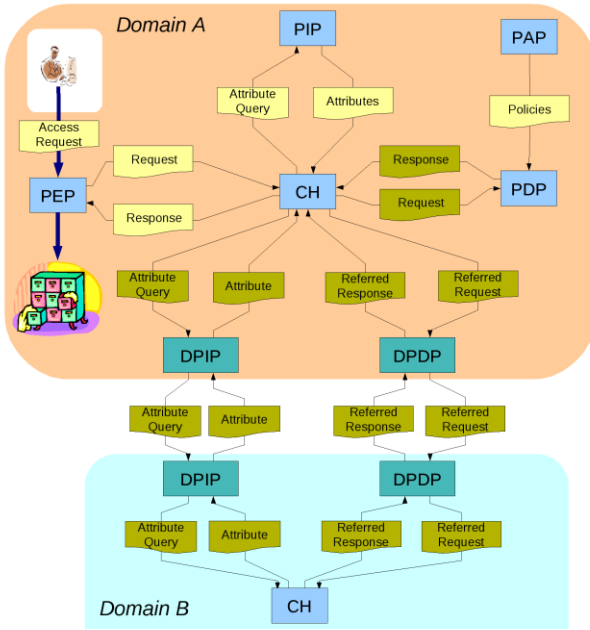


Figure 2: Extending XACML to XADML

Figure 2: Architecture for Distributed Policy Evaluation

### Extending XACML

The extensions towards a support of distributed decisions modeled in XACML should be as minimal as possible in order to ensure an easy adaption of existing interpreters as well as other tools. These new elements are indicated in Figure 3, by a green colour of boxes and lines, while blue indicates existing elements of XACML which are not directly related to attributes. Yellow is used to highlight existing XACML elements related to attributes. The definition of PolicySet has to be copied from the XACML specification and redefined in XADML due to the limitations of the extension mechanisms in XML, which only allows appending of new elements. The details of the new elements are sketched in the following subsections.

### Reference to Policies of other Domains

One of the key aspects of distributed policies is the reference to policies defined by another entity. Due to the potentially distributed definition of the policies, the dependencies among them have to be well observed, in order to ensure that a partial order exists. The referred domains used for deduction are specified for each policy. As indicated in Figure 3 by the dotted line from Domains to VariableReference, we allow variables, whose values are assigned during the evaluation of a policy, to specify the domain used for deduction.

The key point for extending XACML was to support deducting a policy decision based on the results of another policy. This kind of reference is done at the same level as a policy, due to two major reasons. First, the combining algorithms could be used to integrate the respective results including their obligations and advices in a well defined manner. As mentioned above some additional combining algorithms might be needed to handle indeterminate or non applicable references to other policies. Second, `PolicySets` could be used to structure these references into more complicated expressions. This is done through the new element `PolicyReference`, which is quite similar to the `Request` of XACML, but the list of potential attributes has to be extended by the domain to be queried and by a list of attributes. While the attributes used in XACML requests have to be given explicitly, support of dynamic values should be given, to avoid any limitation at this point.

## References to Distributed Attributes

When this kind of attributes are required to take an authorization decision a reference has to be made in a similar way as it is done with the decisions. The element `ReferredDesignator` is an extension of the `AttributeDesignator`, in addition it contains the domain which should provide the specific value. In the same manner `ReferredSelector` is an extension of `AttributeSelector`, whose XPath expression should be evaluated at the domain. After the deduction has been performed, the received distributed attribute can be used like a local attribute.

## Conclusion

This position paper has presented a new approach on deduction of policy decisions in a distributed environment. This approach is based on an extension of the well known XACML, enhancing it with the key feature of distributed policy decisions as well as attributes from different domains. This introduces a new level of abstraction to the XACML based policies. A more detailed description of the proposed extension has been presented in [4]. The work on distributed policies is being carried out within the SWIFT project [8].

## References

- [1] P. Bonatti, S. de Capitani di Vimercati, and P. Samarati. *A modular approach to composing access control policies*. In Proceedings of the 7th Conference on Computer and Communications Security (CCS-7), pages 164-173, Athens, Greece, November 1-4 2000. ACM
- [2] D. W. Chadwick and S. F. Lievens. *Enforcing "sticky" security policies throughout a distributed application*. In MidSec '08: Proceedings of the 2008 workshop on Middleware security, pages 1{6, New York, NY, USA, 2008. ACM.
- [3] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. *A Uni\_ed Framework for Enforcing Multiple Access Control Policies Security*. In Proceedings of the ACM SIGMOD International Conference on Management of Data, volume 26,2 of SIGMOD Record, pages 474{485, New York, May 13{15 1997. ACM Press.
- [4] M. Lischka, Y. Endo, M. Sanchez Cuenca, *Deductive Policies with XACML*, 2009 ACM Workshop on Secure Web Services, 13. November 2009, to be published
- [5] G. Lopez, Oscar Canovas, Joao Girao, and Antonio F.Gomez-Skarmeta. *A Swift Take on Identity Management*. IEEE Computer, pages 58{65, May 2009.
- [6] OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0*, April 2009. Committee Draft 1.
- [7] A. Sarma and J. Girao. *Identities in the future internet of things*. Wireless Personal Communications, 49(2), May 2009.
- [8] SWIFT (Secure Widespread Identities for Federated Telecommunications). <http://www.ist-swift.org>