Access Control Systems
for Spatial Data Infrastructures and their Administration – GeoXACML and the Layered Administration Model

Jan Herrmann
Technische Universität München Faculty of Informatics
herrmanj@in.tum.de

ABSTRACT
Today sophisticated concepts, languages and frameworks exist, that allow implementing powerful fine grained access control systems for protecting Web Services and spatial data in SDIs. Especially rule based access control systems provide the capabilities to define and enforce expressive, fine grained access rights or restrictions respectively. Having powerful and complex access control policies in place introduces a new challenge. It is essential that the policy defining the semantics of an access control system can be easily and securely administrated. In this paper a very general and powerful administrative model, the Layered Administration Model (LAM), will be introduced. The LAM intends to support an easy, secure and tractable administration of complex spatial access control policies as found in SDIs.

Categories and Subject Descriptors
H.4 [Information Systems Applications]:

General Terms
Security, Standardization, Languages, Theory, Verification.

Keywords
Access control, Administration, Administration Model, Policy Analysis, Spatial access control, RBAC, XACML, GeoXACML, Layered Administration Model.

1. MOTIVATION
The value of Spatial Data Infrastructures (SDI) is widely recognized. People of various application domains require access to spatial information and services to solve their complex problems efficiently and with adequate quality. SDIs are usually built based on (OGC) Web Services, as they offer a flexible and efficient way to acquire, use, process, distribute and maintain spatial data. Within SDIs access to the services and the underlying data must be controlled. Hence access control systems must be present to ensure that sensitive services, operations and data are only accessible by authorized subjects. Further the availability of powerful access control systems is important to enhance the quantity of available data and services. At first sight this might be surprising as access control systems usually restrict the access options. While this is true, many organisations having valuable data or services for the community do not publish it at all if only a small subset is sensitive. In many cases the reason for this unsatisfying practise is the lack of trustworthy fine grained access control systems. Providing expressive and manageable access control systems within a SDI would permit these organisations to protect sensitive parts of their data and services and share the part of their data or services that might be of interest for others.

Today sophisticated concepts, languages and frameworks exist, that allow implementing powerful fine grained access control systems for protecting Web Services and spatial data in SDIs. Especially rule based access control systems provide the capabilities to define and enforce expressive, fine grained access rights or restrictions respectively.

Having powerful and complex access control policies in place introduces a new challenge. It is essential that the policy defining the semantics of an access control system can be easily and securely administrated. Administrators having to conduct the administrative tasks have to deal with the high complexity involved in the administration of these policies. There are many reasons for the complexity of the administration of access policies. For example real life IT infrastructures often have:

- many users and roles,
- hundreds of resource classes, each having many attributes and millions of instances and
- numerous services offering various operations on the data objects

Further application domains and legal regulations ask for fine-grained, content dependant, context dependant and spatial access rights. On top of this there are diverse dynamics that make the administration of a large number of complex access rules even more difficult. The data objects, the data model, the service interfaces, the users, the roles, the associated rights and so on can and do change over time and this entails various demanding administrative tasks on the existing policy. Given this situation, it is apparent that there is an urgent need to support administrators of rule bases access control systems for SDIs.
2. OUTLINE

In this paper a very general and powerful administrative model, the Layered Administration Model (LAM), will be introduced. The LAM intends to support an easy, secure and tractable administration of complex spatial access control policies as found in SDIs. Chapter 3 will provide background information on relevant base concepts. Chapter 4 starts by identifying components needed for the administration of access control policies and further requirements towards an administration framework. Subsequently the layered administration model will be introduced and discussed. Chapter 5 closes this paper with a brief conclusion and outlook.
3. BACKGROUND

3.1 Spatial Data Infrastructures based on (OGC) Web Services

This paper assumes that a SDI based on OGC Web Services (OWS) needs to be protected. SDIs are usually based on Web Services, as the Web Service concept allows for easy access, processing and manipulation of data. The characteristic property of spatial data is its spatial extend. Spatial objects called features have one or more spatial attributes representing e.g. the location of the feature. E.g. the location of a person can be represented as a GML point in the person’s data object. Services allowing spatial queries on the spatial data or inserting, analyzing and processing spatial data are called Geo Web Services. OGC Web Services are a set of well defined Geo Web Services allowing for interoperable use, integration and concatenation. Two popular OGC Web Services that will be used in the following examples is the Web Feature Service (WFS) [1] and the Web Map Service (WMS) [2]. A WFS allows for read, insert, update, delete operations on e.g. GML encoded features. A WMS responds to a getMap request with a Map in a raster format showing the requested subset of features.

3.2 Access Control for OGC Web Services

3.2.1 General Security Considerations

Security for existing Web Services should be achieved without or only with minor modifications to the service or client implementations. Therefore it should be the aim to separate the security aspects technically from the Web Service functionality as much as possible. This can be achieved by placing security functionalities as proxy components between the client and the service. Outsourcing security functionality from the services or client implementations comes with further advantages. For example the enforceable access rights become independent of the access control capabilities of the used Service or DBMS implementations. Security functionality should further be divided into separate services that address different security concerns such as authentication, authorization, audit, etc. Such an approach allows combining security services flexibly and reusing existing security concepts and implementations. Having individual security services instead of one all-in-one security component enhances flexibility and scalability and at the same time reduces complexity.

3.2.2 Requirements towards an Access Control System for OWS

Various projects and surveys identified the following requirements towards an access control system for SDIs based on Web Services:

- The definition of fine grained, positive and negative access control rights must be possible. Note that in the OWS context fine-grained means that the atomic access control object is an arbitrary XML node in the Web Service request or response.\(^1\)
- Depending on the service operations it must be supported to enforce access control on the Web Service request and/ or Web Service response (i.e. pre- and/or post-processing access control).
- It shall be possible to base the authorization decisions on the evaluation of other node values. In other words, content dependant access control rights must be declarable.
- In many cases protecting spatial data implies the need for spatial access rights. Spatial rights are a special class of content dependant rights that refer to the spatial characteristics of the data (e.g. allow access to data of buildings with a location within the US).
- Further the definition of (spatial) context dependant rights must be possible. Context refers to the state of the access control system at the time of the interception of the Web Service request or response (e.g. deny access in case of heavy server load).
- It should be supported to base the definition of rights on the RBAC model.
- Enable easy modification of interactions with partially insufficient rights.
- The architecture and components of the access control system, as well as the declaration of rights shall be based on standards. This allows for interoperable connectivity of components (e.g. PESes, PDSes, Policies), reuse of highly tested or verified software and will facilitate an easy implementation and integration of access control systems.
- Easy and sound administration of the access control policies must be possible.

An analysis of these requirements and a comparison with existing access control concepts reveals that the rule based access control approach is thanks to its flexibility and expressiveness very promising. XACML is following the rule based access control concept. Further XACML is a powerful, standardized and popular approach that can be used to build interoperable rule based access control systems.
3.2.3 XACML.

Space limitations do not allow for a detailed introduction in XACML. Here we will just mention its general characteristics. For more detailed information the reader is referred to e.g. [3..xacml spec][4..xacml tutorial].

3.2.3.1 Languages

XACML defines two separate but highly dependant XML encoded languages. The policy language is used to express access rights. The corresponding request/response language is used to state access control decision requests and responses. This language is further providing an abstraction layer towards the application context. At the root of all XACML policies is a PolicySet or Policy element. A PolicySet is a container that can hold other Policies or PolicySets, as well as references to related policies found in remote locations. A Policy element represents a single access control policy, expressed through a set of Rules. A XACML Rule is in fact a logic formula representing a set of conditions on the subject, resource, action and context information, that must be met in order to apply to a given access control decision request. XACML provides a lot of functions to compare values found in a request with those included in the rule. If all the conditions of a rule are met, then the rule applies and its effect is incorporated when deriving the authorization decision. In this paper policy objects like XACML’s Policy or PolicySet element are named Structural Policy Objects (SPO). Another feature XACML supports are obligations. An XACML obligation is a set of instructions specified in a XACML Rule, Policy or PolicySet that should be processed by the PES in conjunction with the enforcement of an authorization decision. Obligations allow for fine-grained pre-processing access control rules as they support the rewriting of Web Service requests.

3.2.3.2 Architecture of the Access Control System

Figure 1 shows the general architecture of a XACML based, modular and potentially distributed rule based access control system. Note that all entities in the figure could also occur more than once.

![Architecture of a XACML based Access Control System.](image)

The Policy Administration Service (PAS) is the component that allows one or multiple policy administrators to edit, maintain and analyze their XACML policy.

The Policy Enforcement Service (PES) is the entry point into the access control process. It can be characterized as a switch or filter that either forwards the intercepted WS-request from the client to the service (and the WS-response from the service to the client respectively) or replies with an adequate error message or modifies the intercepted WS-request or WS-response. The decision whether the WS-request or WS-response is to be forwarded, modified or blocked depends on the authorization decisions, received from the PDS as reply to authorization decision requests. Information usually included in such decision requests message is security assertion information like authentication information, transport protocol information and the Web Service request or response.

The Policy Decision Service (PDS) is the component that derives authorization decisions based on an authorization decision requests, received from PESs. The access control decision request is formed by XACML attributes and in the Web Service use case by the Web Service request or response. This data is referenced by rules through XACML AttributeSelectors or AttributeDesignators and than analyzed according to the logic defined in the rules. XACML offers a great variety of functions that can be used inside rules to express the intended authorization semantics. The effect of a rule (i.e. permit or deny) gets “fired” when its evaluation on a concrete decision request results in the fact that all its conditions evaluate to true. A PDS accesses the policies through a policy repository (implemented e.g. as a file or database) or requests the policies from a PAS.

Comparing the requirements listed above and the capabilities of XACML, it confirms that XACML fulfills most of them. Only two of them are not directly addressed. First, it is not possible to define the needed spatial rights. The next section introduces a standardized extension of XACML, that is additionally allowing to define spatial rules. Second XACML does not address how to maintain XACML policies. The administration service and administration model proposed in section 4 will address this open issue and show how XACML policies can be securely administrated by distributed administrators.
3.2.4 GeoXACML

This section gives a short introduction in the Geospatial eXtensible Access Control Markup Language (GeoXACML). More detailed information on GeoXACML can be found in [5] and [6].

GeoXACML is an OGC standard and defines a geo-specific extension to the OASIS XACML standard. GeoXACML adds the capability to define spatial access control rules. The spatial authorization semantics can be expressed through spatial predicates and functions like within, disjoint, distance or boundary in the condition parts of GeoXACML rules. The arguments of these predicates are spatial attributes in OWS-requests or –responses (represented under <content> in the decision request) and the embedded spatial parameters in the rule itself. Spatial rules augment the capabilities of an access control system as they provide an expressive, native and completely new way to define stable authorization semantics. The spatial extension of XACML is based on a set of simple spatial data types and a comprehensive set of spatial authorization decision functions. GeoXACML’s geometry model is based on the Simple Features Specification [7]. Further GeoXACML allows using different encoding languages for the geometric data types. On top of the geometry model GeoXACML provides a large set of functions that can be used to express spatial rules (c.p. Table 1)
Table 1: Spatial Functions in GeoXACML

<table>
<thead>
<tr>
<th>Topological Functions</th>
<th>Constr. Geometric Functions</th>
<th>Miscellaneous Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Buffer</td>
<td>Distance</td>
</tr>
<tr>
<td>Disjoint</td>
<td>Boundary</td>
<td>IsWithinDistance</td>
</tr>
<tr>
<td>Touches</td>
<td>Union</td>
<td>Length</td>
</tr>
<tr>
<td>Crosses</td>
<td>Intersection</td>
<td>Area</td>
</tr>
<tr>
<td>Within</td>
<td>Difference</td>
<td>IsSimple</td>
</tr>
<tr>
<td>Contains</td>
<td>SymDiff.</td>
<td>IsClosed</td>
</tr>
<tr>
<td>Overlaps</td>
<td>Centroid</td>
<td>IsValid</td>
</tr>
<tr>
<td>Intersects</td>
<td>ConvexHull</td>
<td>Distance</td>
</tr>
</tbody>
</table>

3.2.5 XACML Profiles

XACML and GeoXACML are general purpose standards that can be used in various, highly different scenarios. In certain use cases a set of additional guidelines can be defined, that limit the way how to use the languages but not its expressiveness. These guidelines are packed into so called profiles. Thanks to those profiles, enhanced interoperability and simplified usability can be achieved. There is e.g. an RBAC profile of XACML describing how to use XACML to implement RBAC. The baseline of this paper is to control access in service oriented spatial data infrastructures based on OGC Web Services. Therefore the objects that need to be protected are dynamically generated XML documents - the Web Service request and/or Web Service response respectively. OASIS provides two profiles for the use of XACML with resources that are XML encoded – the Hierarchical resource profile of XACML and the Multiple resource profile of XACML. As the resources in the OGC use case are hierarchically organized nodes encoded in XML, both profiles are of relevance. See [8] and [9] for more detailed information on these two profiles and [11] for information how to apply these two profiles in the access control for OGC Web Services use case.

4. ADMINISTRATION OF ACCESS CONTROL POLICIES

4.1 Components needed for the Administration of Policies

The Administration of rule based access control policies for SDIs needs three central components. First of all one needs a Policy Administration Service through which the policies can be securely administrated and analysed. Second one needs further access control systems protecting the PASes. Third an administration model is needed bringing a clear structure into this set of services, users/administrators, roles, policies etc.

4.1.1 The Policy Administration Service

Through a PAS it must be possible to maintain access control policies. Hence a PAS shall support the following basic operations on policy objects: create, read, delete and update². Next to these basic operations, it is a crucial requirement, that a PAS provides methods to analyse, test and optimize the policies. Thanks to those functions the administrators can verify that a set of defined rules really implement the needed authorisations and restrictions. They thereby win in confidence that the access control system will permit exactly all intended (and only those) information flows between users and services. Further a PAS-Client should provide a graphical user interface (GUI) for the administrators. This GUI is needed to provide improved editing capabilities, to offer a clear view on already defined rules and most importantly to abstract from syntactical details of the access control policy language. Providing the policy administration tool as a service introduces all advantages related to service oriented architectures (e.g. simplifying the distributed administration of remote and possible dispersed policies).

It is important to highlight, that a PAS providing the mentioned capabilities will not solve all the difficulties that arise when administrating access control policies. As pointed out in the introduction, defining and maintaining access control rules is a highly complex task. Essential causes for the complexity of the administration of policies come directly from the application domain and are therefore not avoidable (e.g. quantity and complexity of rights, multiple administrators for the policy producing unintended side effects etc.). Hence one major problem area that is not mitigated by the PAS is, which rules need to be defined by whom, how should they be updated etc. In a first step all related parties have to specify informal or judicial descriptions of the needed authorisation semantics. Afterwards one or more persons have to implement the authorisation semantics according to the given descriptions. Further the administrators have to address the various changes that can occur over time in a coordinated manner. All these steps become particularly demanding when complex policies need to be enforced in large, dynamic and distributed environments. Neither the basic operations nor the analysis and test functions nor the

² For convenience reasons. Can be replaced by a delete request followed by an insert request.
GUI will address the related challenges. In the next section a second central component is introduced, that is urgently needed to achieve sound administration of access control policies.

4.1.2 The Access Control System for the Policy Administration Service

In this section the motivations for an access control system for the PAS itself are explained in detail. This second access control system allows to control under which conditions individual administrators can read, define, delete etc. certain rules. Thus it regulates the syntax and semantics of possible rules.

4.1.2.1 Partitioning Administrative Tasks/Rights

The access control system for the PAS allows to partition the administrative rights, needed to perform necessary administrative tasks, on a fine-grained level. There are three major arguments why one wants to separate the administrative rights and assign them to different administrative persons or roles.

Avoiding excessive demands:

Usually the sum of all tasks that need to be performed when defining and maintaining access control policies for large environments is too voluminous and complex that one administrator alone can securely handle the situation. If this is true in a certain use case, it is reasonable to apply a divide and conquer strategy. It should be the aim to assign only small and ideally disjunctive parts of the administrative tasks to individual administrators, so that each of them can handle the related complexity behind the assigned administrative rights/tasks. Significantly reducing the scope of individual administrators enhances the chances that each administrator will perform correct administrative actions. Such a partitioning of the administrative tasks should of course reflect the competences of the individual administrators. One could for example declare that a specific person working for the building authority will be responsible to define and maintain access control rules for building data in California. Another person working for the road building authority could be in charge to do the same job for data of streets within California and yet another person will control accesses to data of buildings located in NY.

Avoiding excessive accumulation of rights:

Various use cases require, that one person should not be allowed to posses too many administrative rights, even if this person is theoretically capable of handling the related complexity. An appropriate distribution of administrative rights can enforce this requirement. Ideally the partitioning should realize minimal rights per administrator.

Controlling the possession of particularly sensitive rights or the ownership of a critical combination of rights:

Closely related to the last argument is the requirement, that in some situations a certain combination of rights must be distributed over more than one administrator. Compared to the last argument, the focus here is more on the sort of a few rights and not on the quantity of rights.

Even if the motivations for partitioning the administrative tasks and related rights to individual administrators (or administrative roles respectively) differ, the conclusion is in any case the same. One always needs an appropriate way to define and enforce a fine-grained distribution of administrative rights. As we will see later a (Geo)XACML based access control system for the PAS itself can fulfil this requirement.

4.1.2.2 Controlling the mutual influence of cooperating administrators

In many situations the presence of more than one administrator aims at disjunctively distributing the administrative tasks among them. Nevertheless there are situations where this partitioning can and shall not be disjoint. For example overlapping fields of responsibility are needed, if two or more administrators from different domains have to cooperate to get the job done. Another example is the junior, senior administrator relation. Usually the scope of duties of the junior administrator is a subset of the responsibilities of the senior administrator. In those cooperation scenarios it is essential, that all involved parties know the possible influences of other parties on their activities. Again through an access control system for the PAS it is possible to control the mutual influence of the cooperating administrators.

4.1.2.3 Guaranteeing interoperability of policies

As mentioned in section 3 XACML and GeoXACML are very general-purpose standards. This flexibility can cause the loss of interoperability. XACML profiles reduce this flexibility by giving informally specified guidelines how to use the languages in certain situations. Additional there are usually application specific agreements that are informally defined on top of the profiles (e.g. what is the name of the role attribute, what are the names of existing roles, how to encode the authentication type). Thanks to the presence of an access control system for the PAS, all these informal regulations can be expressed in the rule language of the access control system for the PAS. This introduces a formal definition of these guidelines and agreements and further the generic rule enforcement mechanism will take care of the enforcement of the regulations. Thus access control systems for PASes will help establishing truly interoperable access control policies. All policies that were generated under the policies of the access control systems for the PASes will be interoperable as they conform to the guidelines establishing the interoperability.

---

3 The wording administrative tasks and administrative rights are closely related. Assigning administrative tasks implies that a user will need the corresponding rights. Granting administrative rights entitles him to perform the related tasks.
4.1.2.4 Guaranteeing efficient policy evaluation and analysis

Another motivation for an access control system for the PAS is the need to define and enforce certain guidelines, that rules in a policy must obey, to ensure a certain level of performance when enforcing or analyzing the policy afterwards.

4.1.2.5 Enablement of administrative rules

As we will see later, it must be possible to ensure a certain structure of the logical formulas representing the individual rules. This is a central requirement if one wants to allow for administrative rights on rules. In many situations it might for example be needed to enforce that the sub-formulas in a rule are, on the most outer level, combined through conjunctions. Only this structural restriction will ensure that an enforced predicate in a rule must evaluate to true if the rule matches. Assume you would e.g. just say, that a rule must have a predicate that tests whether the requestor has role A and at the same time say nothing about the placement of this predicate in the rule. In such a situation it is not clear whether the predicate that an administrative rule tries to ensure is actually always evaluated when the rule matches or whether it is wrapped by an OR statement. Obviously the OR wrapping will allow to bypass the forced term and therefore requiring that this predicate must be in the rule will be useless.

The arguments presented in the last five paragraphs have shown, that a flexible and expressive access control system for the PAS itself entails a lot of benefits for the administration of access control policies. We therefore conclude, that next to the PAS an access control system for the PAS is needed.
4.2 The Layered Administration Model

4.2.1 Introduction

As we have seen, we need a second access control system and policy to control the administration of access control policies for large SDIs. Pursuing this idea, it becomes apparent, that for the same motivations there is a need for a third access control system protecting the PAS used to administrate the rules protecting the PAS below. This layering of access control systems can be continued as far up as needed (c.p. 4.2.3). Figure 2 visualizes the resulting layers of related access control systems. It should be noted that all entities in figure 2 could occur more than once in a concrete architecture.

![Layers of related access control policies](image)

In the next sections a layer based administration model will be introduced. The reasons for defining an administration model for access control policies are manifold. One important reason for defining an administration model is to introduce various terms, definitions and abstractions that help simplifying explanations, analysis and discussions of complex issues related to the administration of access control policies. Just take the last paragraph for example and see how problematic it was to explain the fact that you have rules about rules and so on. Having a clear terminology greatly helps explaining the effects of defining arbitrary levels of rules about rules. An administration model can further be used as a reference model that helps a lot when designing and implementing an access control system for a service oriented infrastructure. Having a sound well analysed model significantly reduces the costs of building an administration system. Further a system based on the model will directly inherit the well studied characteristics of the model.

4.2.2 Formal Definition of the Layered Administration Model

In this section we define a rule, role and layer based administration model e.g. applicable when administrating spatial access control policies used to protect Web Services in SDIs. As the name reveals one central characteristic of the Layered Administration Model is to introduce layers to structure the entities involved in the administration of policies. Upfront it is worth mentioning that the definition of the LAM is independent of the language used to define the rights and independent of the specifics of the application domain.

Definition: Layer $x$ (0.1)

The abstract word Layer expresses the togetherness of a set of entities. A Layer is defined by a tuple consisting of four sets:

Layer $l_x := (C_x, O_x, R_x, S, CO)$

$C_x = \{c_x | c_x \text{ is a layer } x \text{ class (L}_x\text{-Class)}\}$

$O_x = \{o_x | o_x \text{ is a level } x \text{ object (L}_x\text{-Object); i.e. an instance of a L}_x\text{-Class}\}$

$R_x = \{r_x | r_x \text{ is a level } x \text{ role (L}_x\text{-Role)}\}$

$S = \{s | s \text{ is a service instance}\}$

$CO = \{co | co \text{ is a container instance}\}$

It is very important to note that elements in $C_x$, $O_x$ and $R_x$ always belong to exactly one layer. Only services and containers can belong to more than one layer.
Definition: Lx-Class (0.2)

On each layer there exists a well defined set of Lx-Classes. The Lx-Classes describe the objects on layer x that are used or generated by subjects. The definitions 0.8 to 0.11 specify the sets of Lx-Classes on each layer (i.e. L0-Classes, L1-Classes etc.). The super class of all Lx-Classes of any layer is called Layerclass.

Definition: Lx-Object (0.3)

Lx-Objects are instances of Lx-Classes and are created, queried, updated, deleted, processed etc. by roles or services on that layer. Lx-Objects can e.g. be actual parameters in service requests (e.g. insert operation) or form parts of the service response (e.g. response to a read query).

Definition: Lx-Filter (0.4)

Lx-Filters are a special sort of Lx-Objects. In case of read, delete and update operations the service request will contain one or more Lx-Filter objects, selecting a specific set of Lx-Objects that should be read, deleted or updated respectively. The LAM does not allow for filters selecting objects from different layers. Thus a filter always refers to a set of Lx-Objects and is therefore called Lx-Filter.

Definition: Lx-Role (0.5)

Actions on Lx-Objects are always initiated through subjects that have a Lx-Role activated. Further through Lx-Roles one can only carry out actions on Lx-Objects. To facilitate the administration of rule based systems the LAM is strictly role based. It is not possible to define rights that refer to users directly. Further Lx-Roles are bound to layer x and thus activating a Lx-Role implies that the administrative capabilities are restricted to layer x.

Definition: Container (0.6)

All Lx-Objects reside in containers. Containers can e.g. be a data base table, a file or an XML element. Depending on the implementation of a container, it might be possible to have nested containers. In some cases a service might abstract from the container used to store the objects. Containers can be reused on multiple layers and can therefore hold objects from multiple layers. Hence in general they can not be termed Lx-containers. Nevertheless if in a concrete use case a container instance is only used to store e.g. L1-objects or other L1-containers (in case of nested containers), than this container can be named L1-container.

Definition: Service (0.7)

Services are activated to perform actions on certain objects. In the LAM it is possible that the same service instance can be used on multiple layers. Such a service offers either operations that can work on objects from different layers (only through individual calls on objects of each layer separately) or offers layer specific operations. In case a service can only be used on exactly one layer, the service can be named Lx-Service.

Definition: L0-Class (0.8)

The definition of layer 0 is in its core based on the definition of a set of L0-Classes. L0-Classes are classes that are not classes of the policy languages, used to express the access rights. L0-Classes in the geospatial problem domain are for example Building, Street or PointOfInterest. L0-Classes do not describe policy objects like for example a XACML Rule or an access control matrix.

L0-Class ∈ {all classes except classes of the policy languages}

Definition: L1-Class (0.9)

L1-Classes are all classes of the policy languages used to define the access rights referring to layer 0 entities (e.g. L0-Roles, L0-Objects, L0-Filters). The concrete set of L1-Classes is dependant on the used policy language(s). If you use e.g. XACML or GeoXACML to define your access rules for layer 0 than the following L1-Classes are available:

- L1-XACML-Rule,
- L1-XACML-StructuralPolicyObjects (L1-SPO) like:
  - L1-XACML-Policy,
  - L1-XACML-PolicySet,
  - L1-XACML-PolicyReference and
  - L1-XACML-PolicySetReference.

Semantic of L1-objects:

Layer 1 objects are instances of L1-Classes and refer to entities on layer 0. A L1-rule (i.e. a specific L1-Object) for example can express that only certain building objects (L0-Objects) can be accessed by a user with an activated L0-Role through a WFS read operation (i.e. GetFeature request). The meaning of a L1-Object always refers to the layer 0 entities.

Definition: L2-Class (0.10)
L2-Classes are all classes of the policy languages, used to define the access rights referring to layer 1 entities (e.g. L1-Roles, L1-Objects, L1-Filters). Again, the set of available L2-Classes is dependant on the used policy language (e.g. L2-XACML-Rule, L2-XACML-Policy or L2-XACML-PolicySet).

Semantic of L2-Objects:

L2-Objects refer to L1-Entities. Remember that L1-Objects are administrated and analysed through L1-Roles using the interfaces of a PAS available on layer 1. Therefore L2-Objects control the allowed PAS requests and responses of L1-Roles on layer 1. These requests contain L1-Objects or L1-Filters. Note the semantic difference between L1-Objects and L2-Objects. L1-Objects refer to L0-Roles and L0-Objects. L2-Objects on the other side control the actions that users, having activated a L1-role, can perform through a PAS on L1-Objects. As L1-Objects themselves refer to entities on layer 0 an L2-Object can have an indirect reference to L0-Entities. To clarify this, let’s have a look at a little example. A L2-Rule could e.g. permit, that an administrator with an activated L1-role L1_A could insert L1-Rules through a PAS. This L2-Rule could further require that insert-able L1-Rules must fulfil certain guidelines. Assume for example that all L1-Rules, insert-able by L1_A must control read access to building data (L0-Objects) over a specific WFS. Note that L2-Rules have a L1-specific part and a L0-specific part.

Definition: Lx-Classes of higher layers (0.11)

In the previous section we pointed out, that an arbitrary number of layers can be modelled depending on the needs of the use case. According to the definition of L1-Classes or L2-Classes one can define L3-Classes, L4-Classes etc. Each set of Lx-Classes is dependant on the used policy languages. Policy objects that are instances of Lx-Classes (x>0) are called Lx-Objects and these always directly refer to the entities of the layer x-1. This means, that they only directly control the allowed actions of Lx-1-Roles on Lx-1-Objects. Nevertheless Lx-Objects can have an influence down to level 0 as the direct influence of a Lx-Rule on Lx-1-Objects can set guidelines for the objects below (c.p. 4.2.3).

Definition: Lx-Policy (0.12)

A Lx-Policy (x>0) is a well structured tree or graph of Lx-Objects defining the semantics of a Lx-Access-Control-System. Note that the PES and PDS, are seen as passive services on layer x-1 (x>0) while the policy used by the PDS on layer x-1 is a Lx-Policy. Further note, that it is possible to have more than one Lx-Policy on layer x.

Definition: MultiLayerPolicyCombinerObject (0.13)

In certain situations there might be the need to combine Lx-policies from different layers into an overall policy. The object needed to combine policies from different layers is called a MultiLayerPolicyCombinerObject (MLPCO). This object is not part of any layer. It is simply a special box for Lx-Policies of different layers. This PolicyCombinerObject is provided as base structure at the beginning of the administration process by the Master-Administrator only.

Definition: Layered Administration Model (0.14)

The Layered Administration Model (LAM) structures the administration of access control policies problem domain into an arbitrary number of ordered layers. Layer 0 groups the application domain specific entities, while the layers above represent interrelated levels of access control policies or systems respectively. Lx-Rules refer to entities on layer x-1 (e.g. Lx-1-Objects or Lx-1-Roles). Lx-Roles can only act on Lx-Objects.

Figure 5 in Appendix A visualises the entities in the LAM and some of its relations. The focus of the graphic is to show the grouping effect of layers. Note that this diagram is provided just for explanation purposes and does not represent the needed classes from an implementation point of view.

Closing the formal definition of the LAM it should be mentioned that the LAM is based on and includes the model of the application domain and the access control model (see Fig. 3).

![Figure 3: Relation between models.](image-url)
4.2.3 Characteristics and Explanations

In the last section we tried to keep the formal definition of the LAM as short as possible. In this section additional explanations and characteristics are presented.

4.2.3.1 Benefits

The LAM supports multiple layers of rules about rules. Thanks to a L2-Policy e.g. it is possible to regulate the state of and changes to a L1-Policy. An appropriately defined L2-Policy hence allows to:

- define a partitioning of administrative tasks and rights,
- control the mutual influence of cooperating administrators on level 1,
- define guidelines enhancing interoperability,
- define guidelines enhancing the performance when using or analysing L1-Policies,
- explicitly describe delegation of L1-Rights (L3-Rules in turn control the possible delegations)
- ...

4.2.3.2 Horizontal Partitioning

Using a powerful fine-grained policy language to define L2-Rules, allows amongst others partitioning the administration problem on level 1. This partitioning is called horizontal partitioning as it refers to one layer. Having a large application domain on Level 0 with complex access control requirements might imply that a lot of partitions on level 1 will be needed to solve the administration related complexities and challenges on level 1. This further entails, that many L2-Rules will be needed, that in turn are hard to administrate as well. So there is a need to partition the administration of L2-Rules through L3-Rules. This should be continued as long as the rules on a certain level can be easily administrated.

4.2.3.3 Vertical Partitioning

Having multiple layers of policies separates the administrative tasks in a vertical way. Layer 3 rules e.g. impose conditions on possible L2-rules. In case the L3-Rule refers to L2-Rules, that in turn are referring to the insertRule(L1-Rule) operations, than the L3-Rule might indirectly add constrains on possible L1-Rules. In general, administrators on the highest layer n start defining some first guidelines that certain rules on the layer n-1 below have to fulfill. These Ln-Rules set the administrative frame for the layers below. By defining Ln-Rules one indirectly imposes conditions on possible actions on all layers below. Further the rules on layer n-1 (fulfilling the rules of layer n) extend and refine the existing guidelines of the layer n rules. In this way the guidelines are refined on each layer in a controlled way. The final result is a set of L1 rules that fulfil all the guidelines expressed by all the administrators on all levels above. These L1-Rules enforce the access semantics on the services and data of the application domain (L0) as intended by the administrators of all layers above. LAM’s capability to distribute the administration problem on multiple layers is called vertical partitioning. Vertical partitioning can be very useful in real life infrastructures. In SDIs or large global companies for example different levels of organisational units (e.g. federal level, state level, county level) express access semantics. In many cases the highest level does not know the details of the level below and can only specify broad guidelines that the next organisational layer has to fulfill. The layers of the LAM naturally model these administrative chains and the needed indirection. Next to these organisational requirements rules from higher layers support an easier administration of rules at a lower level. The vertical partitioning simplifies the horizontal partitioning, as the degree of freedom when defining Lx-rules is constantly reduced. Reduced options (e.g. for access control, interoperability or performance reasons) when defining and maintaining rules simplifies the related tasks. Think e.g. of an extreme example where an L1 administrator just has the choice to activate/insert completely pre-specified L1-Rules or L1-Rules that conform to a very specific template. In this case the L2-Rules dictate the content of definable L1-Rules by a specific administrator with no or hardly any degree of freedom.

One common strategy when defining Lx-Rules is to allow inserting Lx-1-Rules under two sorts of restrictions. First you control the structure of the formula representing the Lx-1-Rule (e.g. on the most outer level of the formula all sub-formulas are combined by AND.) Second you specify how individual sub-formulas have to look like. These conditions in the Lx-Rule can be very specify forcing a sub-formula (as described in the Lx-Rule) to be one-to-one in the Lx-1-Rule or can only impose loose conditions that the sub-formula must adhere to. In either case the effect is, that a rule on a lower level might consist of a set of maybe redundant or even contradictory sub-formulas. Sophisticated analysis and optimization functions inside the PAS allow resolving the redundancies and producing simplified formulas. Additionally semantic contradictions can be indentified and reported automatically.

4.2.3.4 Note on the structure of rules

Each Lx-Rule has a characteristic structure. A L3-Rule for example can have a L2 specific part, a L1 specific part and L0 specific part. For example a L3-Rule could define that...

L2-specific part:
- ...a L2-Rule L2_A can insert E2-Rules if the L2-Rule fulfils the following conditions:...

L1-specific part:
- ...the L2-Rule must refer...
to the L1-Roles L1_A or L1_B,

to insertRule operations,

and all insert-able L1-Rules must in turn fulfil the following conditions:

L0-specific part:

- the L1-Rule must refer to
- the L0-Role L1_staff,
- Service A (e.g. WFS) operations and
- Building data objects within the US.

4.2.3.5 Anchoring rules on layer x

The LAM requires that each Policy Object (a rule or a structural policy object) is anchored on a specific layer. A rule is a Lx-Rule (x>1), if the rule refers to Lx-1-Entities. In case the Lx-Rule allows a Lx-1-Role to read or delete Lx-1-Objects the rule is semantically anchored on Lx if the objects are real Lx-1-Objects. This test can rely on the fact that only valid Lx-1-Objects can exist. Thus checking the layer attribute of the Lx-1-Objects will directly reveal if the object subject to the read or delete right is a Lx-1-Object. The fact that only objects and only those that are properly semantically anchored on Lx-1 have a layer attribute equal x-1 makes this test save.

In case the Lx-Rule allows a Lx-1-Role to insert Lx-1-Object objects, it must be checked whether the insert able objects are true Lx-1-Objects. Then the test goes into a recursion and has to check whether the Lx-1-Role (as constrained in the Lx-Rule) is a proper Lx-1-Object. This recursive step terminates directly if the Lx-1-Role refers to read or delete operations on Lx-2 labelled objects, as, according to the LAM, all Lx-2 labelled objects have to be well anchored. In case the Lx-1-Role refers to a insert(Lx-2-Role) operation the next recursive step follows. This recursion reaches at latest its break condition if the level of the rule under test equals 1. In this case the rule must refer to L0 entities. This means that the objects the rule refers to should not be Rule, SPO or Lx-Filter (with x>0) objects as Rule, SPO and these Filters are no L0-Classes.

Note that the test, ensuring that a object is a well anchored Lx-Object, is implemented by one or more rules on Layer x+1. The algorithm presents a general guideline how to define properly anchored Lx-Rules.

4.2.3.6 Anchoring SPO objects on layer x

Guaranteeing that only well anchored Lx-SPO exist in the policies is not as strait forward as in case of rule objects. SPO Objects are usually used to group the rules in order to improve the performance of rule evaluation (and maybe analysis). In many cases SPOs are generated automatically from existing rules, as there is in general no need to define a SPO structure by hand. In this process certain predicates of similar rules are unioned and simplified and than serve as the predicates of e.g. a XACML Policy. It can not be guaranteed that the conditions in a SPO are sufficiently complete such that an algorithm can decide that they are semantically well anchored on a layer. The solution to this problem is a simple structuring requirement of the Lx-Policy. The constraint is that all SPO under the Lx-Policy root (the Lx-root-SPO – anchored on layer x by definition) shall contain predicates referring to the role attribute. Thus all children of the Lx-root-SPO are well anchored on layer x if the roles the SPO refers to are all Lx-Role names. All other SPO or rules are descendants of these now well anchored Lx-SPO objects. From there it is save to label descendant SPOs as Lx-SPOs, if they are child of a Lx-SPO object. Remark that this anchoring of Lx-SPOs is based on the placement of the SPO in the policy tree only. It leaves options open in which predicates are included in these Lx-SPO’s, that refer to entities that are not on Layer x-1. While this is not intended this causes no security risk. Doing so will channel wrong decision requests to the SPO and in the end maybe to the Lx-rules. This is no risk as all Lx-Rules are well anchored on level x and will not be applicable to these wrongly channelled decision requests. Note that these unintended situations never occur if the SPO’s are calculated automatically from the rule predicates. They only occur when defining the SPO explicitly. Anyway it is no security problem when one defines incorrect Lx-SPOs.

4.2.3.7 Dependency of rights

In many scenarios there are relations between the roles of one layer. Assume e.g. a L2-Role L2_insert role to insert L1-Rules. Bob using this L2_insert role, is able to insert L1-Rules. In order to see the rules he is defining, it will in general be useful, if Bob can see the defined rules. Further it is practical if he is able to update and delete his defined rules. The dependency of rights/roles referring to read-, insert-, update- and delete-operations is shown in Figure 4. The subset a e.g. insert-role refers to (e.g. permit insert of building data in state CA through WFS XY), is the same as the dependant roles refer to.

![Diagram](image-url)  

Figure 4: Dependencies of Lx-Roles.
These dependencies of roles can e.g. be implemented through new Lx-roles inheriting the rights assigned to the existing roles like L2_update, L2_delete, L2_read. In the case just described all roles belong to one layer and can thus be easily combined. One could either combine the rights through a new Lx-role on layer x inheriting the rights of other Lx-roles. Alternatively one could let an existing Lx-role directly inherit from other Lx-roles.
4.3 Implementation of the LAM in (Geo)XACML

We have implemented an administration infrastructure based on PASes and the LAM. The PAS can be reused on all layers above layer 0. It provides a set of basic functions (e.g., insert(L3-Rule)) as well as analysis and optimization functions. We are using (Geo)XACML as the policy language on all layers. This allows defining fine-grained, complex, spatial and even context dependant rights on each layer. Space limitations don’t allow going into the details but some interesting points should nevertheless be mentioned.

Every XACML Object (i.e., Rule and SPO like Policy, PolicySet, RPS, PPS, PolicySetReference, PolicyReference) has a special XML attribute, named layer, in its root element encoding the layer of an object. Lx+1-Rules guarantee that all XACML Lx-Objects with a layer attribute equal x semantically refer to Lx-1-Entities only.

According to the LAM, each Lx-Object must be anchored on a specific layer. Below it will be analyzed how objects of the different classes can be related to a certain layer.

- An XACML RPS is an Lx-RPS if it refers to a set of Ex-1-roles.
- A XACML Policy(Set)Reference is an Lx-Policy(Set)Reference if its father is an Lx-PS (more precisely an Lx-RPS, Lx-PPS, or an descendant Lx-PS of an Lx-PPS) and if it is pointing to a Lx-PS (or to an Lx-P in case of a PolicyReference).
- A XACML PPS is a Lx-PPS if it is referenced through a Lx-PolicySetReference inside a Lx-RPS. One option to ensure this is through a PAS providing a high-level-PAS operation insert-RPS/PPS-Pair. This encapsulation always implies that only valid references between RPS and PPS can be defined. As a RPS can be anchored as described above, the corresponding PPS is automatically anchored. Another option to ensure the validity of references between some SPOs on one layer is to use a specific naming schema for these objects.
- A XACML PS is a Lx-PS if it is child of a Lx-PS or Lx-PPS.
- A XACML P is a Lx-P if it is child of a Lx-PS or Lx-PPS.
- A XACML Rule is an Lx-Rule if it is child of a Lx-P (this actually implies that it exists in a Lx-PPS and therefore refers to the Lx-1 role(s) of the corresponding Lx-RPS) and fulfils the conditions described in (4.2.3.5).

Note that further conditions apply towards the above elements according to the XACML specification, the XACML RBAC profile, the multiple and hierarchical resource profile etc. They have been omitted here as they do not play a significant role in the anchoring of RPS, PPS etc. on a certain layer.

A very useful feature of our PAS is the capability to automatically generate a performance optimized Lx-Policy out of an Lx-Policy following the structuring described above. The algorithm behind the exportOptimizedPolicy method is that the whole policy is restructured in such a way that a performance optimized policy evaluation and analysis can be guaranteed. Next to the restructuring the individual rules are simplified. Naturally all transformations of the policy are strictly preserving the authorization semantics.

5. CONCLUSION AND OUTLOOK

In this paper we presented the Layered Administration Model. The model describes the use of multiple layers of dependant access control systems to simplify the administration of access control policies. Space limitations did by far not allow a detailed summary of the research results. A more in-depth presentation of the findings will be given in [10]. In this document the LAM will also be discussed in the context of related work (e.g., relation between LAM and ARBAC and how to implement ARBAC on top of the LAM). Next to this focus on administrations models, the analysis of spatial access control rules is highlighted in the thesis. Various reasoning tasks are analysed and extended. An especially interesting area that is addressed, is the use of spatial logics and spatial reasoning on spatial access control rules.

Next to the administration topic and the focus on analysis of policies, new enhancements of the XACML and the GeoXACML standard, that are in-between mostly integrated in the to-be-standardised future 3.0 version of XACML and its profiles are introduced.

6. REFERENCES
[3] XACML spec
[4] XACML intro
[7] SFS
[8] Hierarchical resource profile of XACML 2.0
[9] Multiple resource profile of XACML 2.0
Figure 5: LAM's Entities and their relations.