# 8.1 Authorization

This pattern describes who is authorized to access specific resources in a system, in an environment in which we have resources whose access needs to be controlled. It indicates, for each active entity that can access resources, which resources it can access, and how it can access them.

## *Example*

In a medical information system we keep sensitive information about patients. Unrestricted disclosure of this data would violate the privacy of the patients, while unrestricted modification could jeopardize the health of the patients.

## *Context*

Any environment in which we have resources whose access needs to be controlled.

## *Problem*

We need to have a way to control access to resources, including information. The first step is to declare who is authorized to access resources in specific ways. Otherwise, any active entity (user, process) could access any resource and we could have confidentiality and integrity problems.

How do we describe who is authorized to access specific resources in a system?

The solution to this problem must balance the following forces:

- The authorization structure must be independent of the type of resources. For example, it should describe access by users to conceptual entities, access by programs to operating system resources, and so on, in a uniform way.
- The authorization structure should be flexible enough to accommodate different types of subjects, objects, and rights.
- It should be easy to modify the rights of a subject in response to changes in their duties or responsibilities.

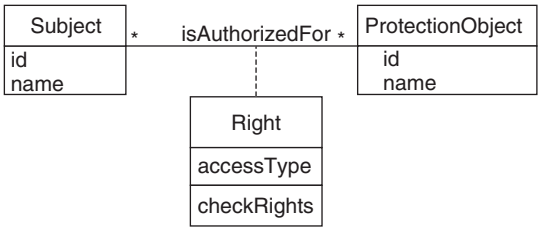
## *Solution*

Indicate, for each active entity that can access resources, which resources it can access and how.

## Structure

The `Subject()` class describes an active entity that attempts to access a resource (Protection Object) in some way. The `ProtectionObject()` class represents the resource to be protected. The association between the subject and the object defines an authorization, from which the pattern gets its name. The association class `Right()` describes the access type (for example, read, write) the subject is allowed to perform on the corresponding object. Through this class one can check the rights that a subject has on some object, or who is allowed to access a given object.

The figure below shows the elements of an authorization in form of a class diagram.




Class model for AUTHORIZATION (245)

## Implementation

An organization, according to its policies, should define all the required accesses to resources. The most common policy is need-to-know, in which active entities receive access rights according to their needs.

This pattern is abstract and there are many implementations: the two most common approaches are Access Control Lists and Capabilities [Pfl03]. Access Control Lists (ACLs) are kept with the objects to indicate who is authorized to access them, while Capabilities are assigned to processes to define their execution rights. Access types should be application oriented.

## Example Resolved

A hospital using an authorization system can define rules that allow only doctors or nurses to modify patient records, and only medical personnel to read patient records. This approach allows only qualified personnel to read and modify records.

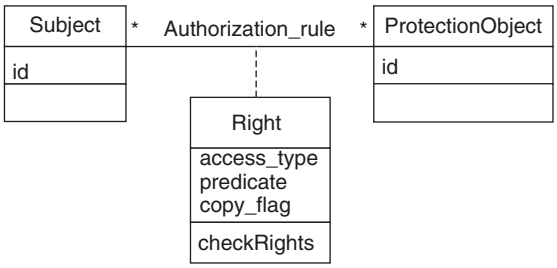
## Variant

The full access matrix model usually described in textbooks also includes:

- Predicates or guards, which may restrict the use of the authorization according to specific conditions

- Delegation of some of the authorizations by their holders to other subjects through the use of a Boolean 'copy' flag

The next figure extends AUTHORIZATION (245) to include those aspects. Right now includes not only the type of access allowed, but also a predicate that must be true for the authorization to hold, and a copy flag that can be true or false, indicating whether or not the right can be transferred. `CheckRights` is an operation to determine the rights of a subject or to find who has the rights to access a given object.




Extended AUTHORIZATION (245)

## Known Uses

This pattern defines the most basic type of authorization rule, on which most more complex access-control models are based. It is based on the concept of access matrix, a fundamental security model ([Pfl03] and [Sum97]). Its first object-oriented form appeared in [Fer93]. Subsequently, it has appeared in several other papers and products ([Ess97] and [KBZ01]). It is the basis for the access control systems of most commercial products, such as Unix, Windows, Oracle, and many others. PACKET FILTER FIREWALL (405) implements a variety of this pattern in which the subjects and objects are defined by Internet addresses.

## Consequences

The following benefits may be expected from applying this patter:

- The pattern applies to any type of resource. Subjects can be executing processes, users, roles, user groups. Protection objects can be transactions, data, memory areas, I/O devices, files, or other resources. Access types are individually definable and can be application-specific in addition to the usual read and write.
- It is convenient to add or remove authorizations.
- Some systems separate administrative authorizations from user authorizations for further security, on the principle of separation of duties [Woo79].

- The request may not need to specify the exact object in the rule: the object may be implied by an existing protected object [Fer75]. Subjects and access types may also be implied. This improves flexibility at the cost of some extra processing time to deduce the specific rule needed.

The following potential liabilities may arise from applying this pattern:

- If there are many users or many objects, a large number of rules must be written.
- It may be hard for the security administrator to realize why a given subject needs a right, or the implications of a new rule.
- Defining authorization rules is not enough, we also need an enforcement mechanism.

## See Also

ROLE-BASED ACCESS CONTROL (249) is a specialization of this pattern. REFERENCE MONITOR (256) complements this pattern by defining how to enforce the defined rights.