BEST PRACTICE
AUTHORIZATION CONCEPT

This document outlines best practices for structuring an authorization concept leveraging NOUMENA technology, focusing on functional rights and data access. An authorization concept is a key element of any solution in an enterprise environment, even more so if certifications such as ISO 27001, or licenses for financial service providers are required.

INTRODUCTION

A robust authorization concept is crucial for securing data and ensuring proper access controls within a system, guaranteeing data authenticity and process integrity. This document aims to provide guidance on designing an effective authorization strategy using NOUMENA's technology, emphasizing the separation of functional rights and data access.

Regulatory oversight highly appreciates our approach to authorization as it shows that it is put at the heart of the design of the solution. Regulated entities or entities following standards such as ISO 27001 need to be able to produce reports on who has what right.

For less strict regulated entities, looking at authorisation might make more sense to look at from a data perspective. Where it is more crucial to give access based on data rather than functionality.

KEY PRINCIPALS

- **Principle of Least Privilege:** Grant users only the minimum level of access necessary to perform their job functions.
- **Separation of Duties:** Divide responsibilities among different roles to prevent any single individual from having too much control.
- **Instance and Context Based Access Control:** Assign access to actions on particular instances including their process state, rather than global

STRUCTURING THE AUTHORIZATION CONCEPT

A well-designed rights and access scope model forms the foundation of any secure enterprise application. To ensure effective control, access scopes and user rights must be carefully aligned: broad access scopes may necessitate tightly restricted user rights, while narrow access scopes can allow for more permissive roles. Conversely, a combination of wide-ranging access scopes and elevated rights can result in unintended "super users," posing significant security and compliance risks.

FUNCTIONAL RIGHTS AND ROLES

Functional rights define what actions a user can perform within the system. These rights should be granular and tied to specific business functions. These rights are wired into protocols and cannot be changed without deployment. Roles on the other hand are bags of rights that can be assembled in an IAM solution such as KeyCloak or Azure Entra.

1. Define Functional Rights
Actions a user can or has to perform such as triggering a payment or approval of a request. Highly regulated entities tend to need a very granular approach. Defining this is crucial before you start implementing.

2. **Define Business Roles**
Based on the target operating model define required roles which need to be represented in the system. Examples

3. **Assign Rights to Roles**
Assign specific functional rights to each role. For example, „Customer Success" might have rights to „Review activities" and „Approve Onboarding", while the "Operator" has „Update Settings" or "Manual Overwrite" rights.

ACCESS SCOPE

Access scope controls who can view and interact with specific data / protocols. This requires identifying the data entities and their access levels.

1. Define Access levels
Define on what level data access will be granted. We differentiate conceptually three levels of access. For this purpose lets assume a CRM system with a Key Account Manager (KAM)

- **Individual record**
  A particular KAM enters a draft prospect into the system. No one else should have access at this point, but the record and related data should be stored and accessible by the particular KAM

- **Group of records**
  In a business with few very large clients a particular KAM might have only access to particular name based clients. On the other hand if the business has thousands of customers, KAMs might be able to access clients based on their region of origin, or subscription level. This is the approach taken by most applications we have seen in the past.

- **Global:** An edge case in our understanding would be a global key account manager that has access to all customers. Even though there might be cases where this is needed, e.g. for compliance or operational tasks, nonetheless this is an open door for least privilege principle violation. If this global approach is taken, the functional right should be extremely specific to counter weigh the global access. Eg. the Global KAM could be modeled to have only read rights, and regional KAMs would have full access to the customer.

2. Assign data scope-based attributes
Use user attributes or better IAM groups to assign users to a specific data scope.

PROTOCOL PARTIES

Considering both rights and data scope are represented as claims, they need to be combined in a party on a protocol to become useful. At initiation of a protocol parties are designed as entities that have an interest and therefore can access and interact with the protocol.

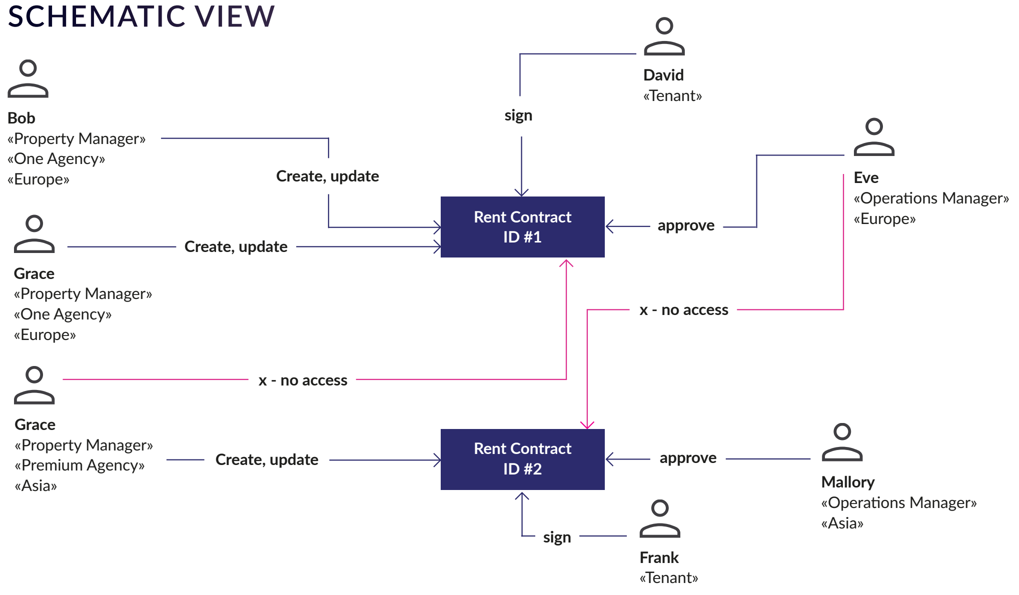
Considering two users in the Customer Success department Alice and Bob. Alice is responsible for Europe and Bob for the US. In the IAM they got the role Customer Success and Europe / US respectively. The Customer Success Role contains the right to approve customers after reviewing their KYC documentation.

This means they both can do the same functionally but on different customers. How this plays out will become more clear in the concrete example below.

CONCRETE EXAMPLE

Designing an authorization concept for a company called "Real Estate Wonder" which allows property managers to set up rent contracts for their tenants. In this system property managers set up contracts, tenants sign and operations managers approve the contract. Property Managers have a dedicated todo list that only they can edit. The business is organized in geographic regions, and real estate agencies only have access to their own data.

SCHEMATIC VIEW



USER SETUP

To simplify the example each role will consist only of one right with the same name. In reality a role would contain multiple rights.

User	Role / Rights	Data Scope
Bob	Property Manager	Europe, One Agency
Grace	Property Manager	Europe, One Agency
Alice	Property Manager	Asia, Premium
David	Tenant	No scope required
Frank	Tenant	No scope required
Eve	Operations Manager	Europe
Mallory	Operations Manager	Asia

This setup will be reflected in the chosen IAM. The data scope for the individual datascope needed tenants (David and Frank) which is only accessible by an individual user, does not need to be modeled, as the user_id is used for that.

USER SETUP

Protocol	Description	Parties	Consquences
Rent Contract ID #1	The rent contract David signs with One Agency	Tenant { right: **Tenant**, datascope: **David**}, Property Manager {right: **Property Manager,** datascope: **One Agency**} Operator {right: **Operations Manager**, datascope: **Europe**	David fulfils the "Tenant" requirements, Bob and Grace the "Property Manager" requirements and Eve the one for the "Operator". Everybody else has no access to this particular protocol. Eg. Alice who has the right "Property Manager" but doesn't have the datascope "One Agency" will be prohibited access.
Rent Contract ID #2	The rent contract Frank signs with Premium Agency	Tenant { right: **Tenant**, datascope: Frank}, Property Manager {right: **Property Manager,** datascope: **Premium Agency**} Operator {right: **Operations Manager**, datascope: **Asia**	Frank fulfils the "Tenant" requirements, Alice the "Property Manager" requirements and Mallory the one for the "Operator". Everybody else has no access to this particular protocol. Eg. Eve who has the right "Operations Manager" but doesn't have the datascope "Asia" will be prohibited access.

WHY IS THIS USEFUL?

This approach gives you three benefits: simplifies employee processes, allows de-coupling of user management (rights) and implementation of business functionality from the start, and

Employee onboarding, offboarding and changing their responsibilities is a non-trivial task and tends to cause friction and potential for data breaches. Concretely, this means for the four aspects

Onboarding
Lets assume an additional member joins the Operations Team in Asia, the only thing that needs to be done is granting him the user role "Operations Manager" and the data scope "Asia". This will give the user immediately the same access as Mallory.

Offboarding
The same mechanism can be leveraged for off boarding, if the user e.g. is offboarded in multiple steps, the given roles and data scopes can be revoked one by one (after handover) or all together, and at the actual exit date the user is deactivated.

Change of responsibilities
Lets assume an additional member joins the Operations Team in Asia, the only thing that needs to be done is granting him the user role "Operations Manager" and the data scope "Asia". This will give the user immediately the same access as Mallory.

Change of role responsibilities
if a role is enriched or further specialized individual rights can be added or removed from the role definition without interfering with the running system.

Using such a formal definition of authorization at the beginning of the implementation ensures authorization principles are implemented throughout the system at sprint 0. Which is a key success factor for operationalized applications, the likelihood of something being overseen or omitted at later stages of the development cycle is very high and costly.

The laid out approach allows any organization to implement an adaptive authorization model flexible in terms of what data is accessible to users and what they are able to do with it.