****

**Attribute-Based Access Control in a Microservices Architecture**

RELEASE DOCUMENT

Version 1.0

**REVISION HISTORY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Baseline Date** | **Change Description** | **Author** | **Reviewed By** | **Approved By** |
| **Name & Title** | | |
| 1.0 | 11/20/2020 | Baseline Version | Chetan Dravekar |  |  |

**TABLE OF CONTENT**

Contents

[Introduction 4](#_Toc69740775)

[What is role-based access control (RBAC)? 4](#_Toc69740776)

[What is attribute-based access control (ABAC)? 9](#_Toc69740777)

[ABAC Architecture 11](#_Toc69740778)

[Role-Based Access Control (RBAC) vs. Attribute-Based Access Control (ABAC) 12](#_Toc69740780)

[5 Identity Management Scenarios to Study 14](#_Toc69740781)

[How to Choose an Access Control Solution 15](#_Toc69740782)

[ABAC Microservice POC Architecture Diagram 16](#_Toc69740783)

[POC implementation UML SEQUENCES Diagram 17](#_Toc69740784)

[1. Sequence Diagram for Product Microservice Data Access 17](#_Toc69740785)

[1. Sequence Diagram for Customer Microservice Data Access 18](#_Toc69740786)

[REFERENCE DOCUMENTS 19](#_Toc69740789)

# Introduction

Access control is a fundamental element of the security infrastructure of any company. Every security officer wants to apply the principle of less privilege, [zero-trust](https://www.ekransystem.com/en/blog/zero-trust-security-model), segregation of duties, and other best practices without harming the company workflow.

There are several approaches to organizing an access management system. In this article, we analyze the two most popular access control models: role-based and attribute-based. We’ll talk out the pros and cons of each model, compare them, and see if it’s possible to combine them.

# What is role-based access control (RBAC)?

**Role-based access control** (RBAC) is an access control method based on defining employee roles and corresponding privileges within the organization. The idea of this model is that every employee is assigned a role. Every role has a collection of permissions and restrictions. An employee can access objects and execute operations only if their role in the system has the relevant permissions.

For example, a company’s accountant should be allowed to work with financial information but shouldn’t have access to client contact information or credit card data.

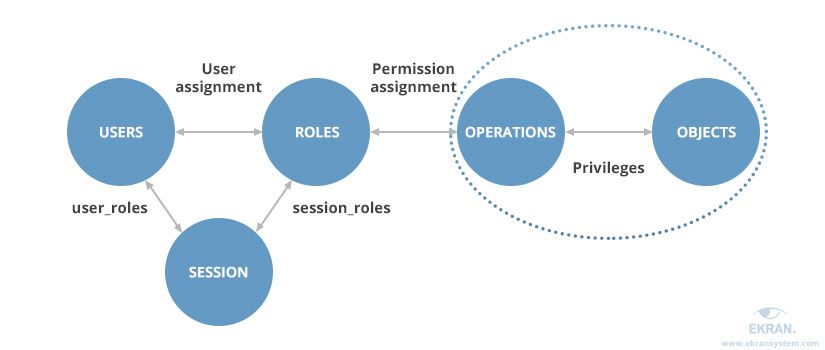
A user might be assigned to one or several roles. When a new employee comes to your company, it’s easy to assign a role to them. And when someone leaves the company, you don’t need to change the role parameters or a central policy.

Let’s consider the main components of the role-based approach to access control:

* ****User**** – an individual (with UID) with access to a system
* ****Role**** – a named job function (indicates the level of authority)
* ****Permission**** – equivalent to access rights
* ****Session**** – a mapping between a user and a set of roles to which the user is assigned in the context of a working time
* ****Object**** – a system resource that requires permission to access
* ****Operation**** – any action in the protected network

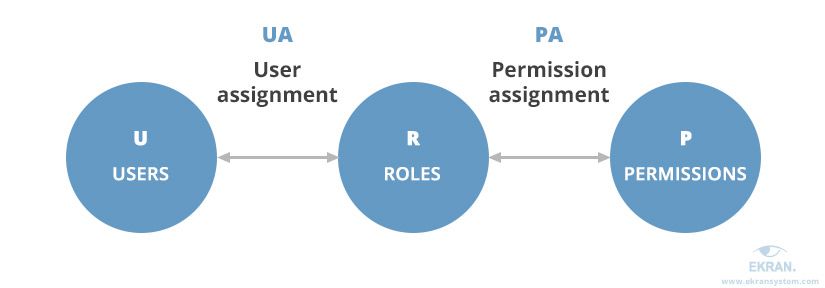
The basic rules of RBAC are:

* A user can execute an operation only if there is a role assigned to the subject.
* Identification and authentication are not considered operations.
* All user activities are carried out through operations.

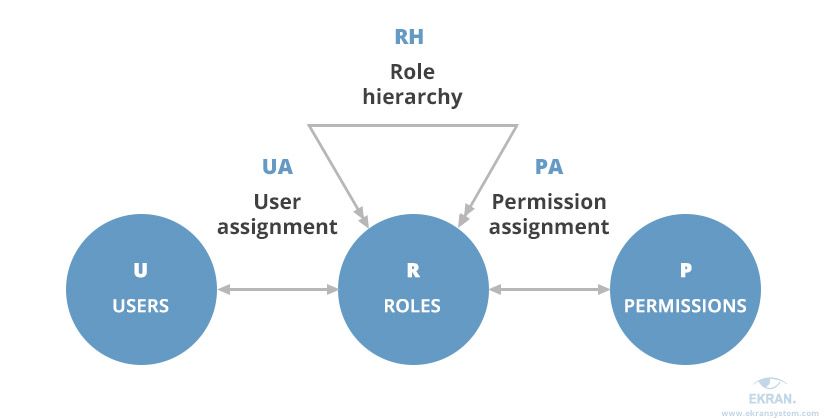


RBAC can be implemented on four levels, according to the [**NIST RBAC model**](https://csrc.nist.gov/CSRC/media/Publications/conference-paper/2000/07/26/the-nist-model-for-role-based-access-control-towards-a-unified-/documents/sandhu-ferraiolo-kuhn-00.pdf). Each subsequent level includes the properties of the previous. Let’s take a look at them:

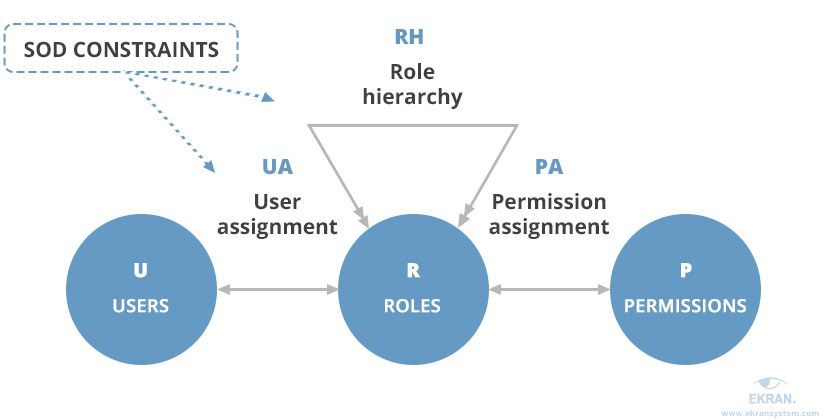
****1. Flat RBAC**** is an implementation of the basic functionality of the RBAC model. All users and permissions are assigned roles. Users obtain the permissions they need by acquiring these roles. There may be as many roles and permissions as the company needs. A single user can be assigned to multiple roles, and one role can be assigned to multiple users.



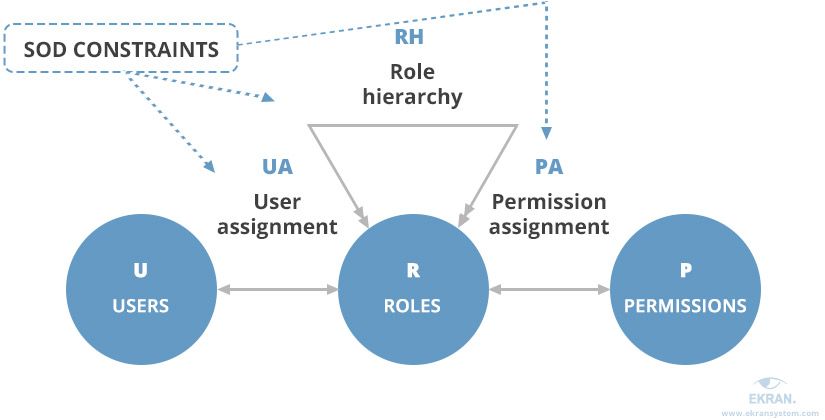
****2. Hierarchical RBAC****, as the name suggests, implements a hierarchy within the role structure. This hierarchy establishes the relationships between roles. Users with senior roles acquire permissions of all junior roles, which are assigned to their subordinates. The complexity of the hierarchy is defined by the needs of the company.



****3. Constrained RBAC**** adds a [**separation of duties**](https://en.wikipedia.org/wiki/Separation_of_duties) (SOD) to a security system. SOD is a well-known security practice when a single duty is spread among several employees. It’s quite important for medium-sized businesses and large enterprises. Separation of duties guarantees that no work can introduce fraudulent changes to your system that no one else can audit and/or fix.



****4. Symmetric RBAC**** supports permission-role review as well as user-role review. It allows identification of the permissions assigned to existing roles (and vice versa). For example, by identifying permissions of a terminated employee, the administrator can revoke the employee’s permissions and then reassign the role to another user with the same or a different set of permissions.



Defining a role can be quite a challenge. You have to consider all the permissions a user needs to perform their duties and the position of this role in your hierarchy. If you assign too many permissions to a role, it will break the least privilege principle and may lead to privilege misuse.

Role-based access control is most commonly implemented ****in small and medium-sized enterprises****. Such organizations typically have simple workflows, a limited number of roles, and a pretty simple hierarchy, making it possible to effectively determine and describe user roles.

Once all the necessary roles are set up, this model doesn’t require a lot of maintenance and support from the IT department. Implementing RBAC can help you meet IT security requirements without much pain. On the other hand, creating a complex role system for a large enterprise may be challenging. The organization with thousands of employees can end up with a few thousand roles. This is known as role explosion, and it’s unavoidable for a big company.

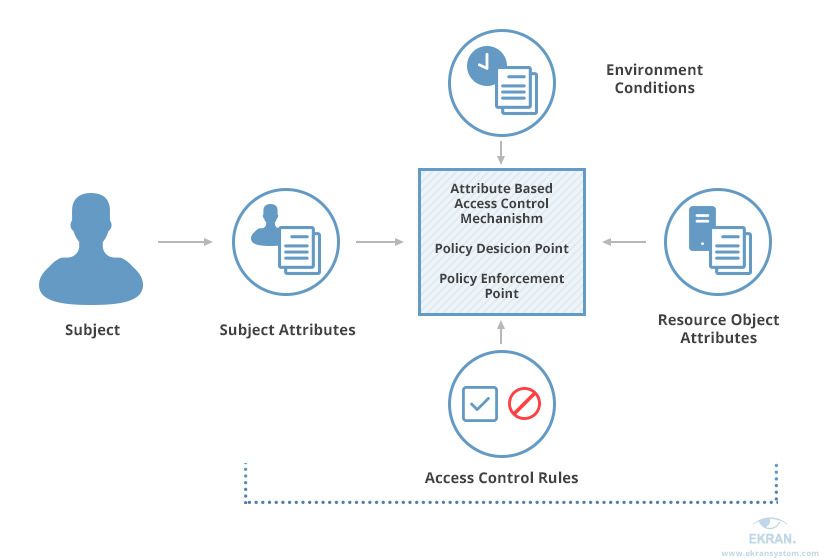
# What is attribute-based access control (ABAC)?

**Attribute-based access control** is a model that evolved from RBAC. This model is based on establishing a set of attributes for any element of your system. A central policy defines which combinations of user and object attributes are required to perform any action.

Let’s consider the **main components** of the ABAC model according to NIST:

* ****Attribute – a characteristic of any element in the network. An attribute can define:****
  + ****User**** characteristics – employee position, department, IP address, clearance level, etc.
  + ****Object**** characteristics – type, creator, sensitivity, required clearance level, etc.
  + Type of ****action**** – read, write, edit, copy, paste, etc.
  + ****Environment**** characteristics – time, day of the week, location, etc.
* ****Subject**** – any user or resource that can perform actions in the network; a subject is assigned attributes in order to define its clearance level
* ****Object**** – any data stored in the network; objects are assigned attributes in order to describe and identify them
* ****Operation**** – any action taken by any subject in the network
* ****Policy**** – a set of rules allowing or restricting any action in your information retrieval system; rules are “IF/THEN” statements based on attributes of any element (user, resource, environment)

Unlike in RBAC, in ABAC you can even use attributes that aren’t yet registered in the system but will appear during the work process.



This approach is ****suitable for a company of any size**** but is mostly used for large organizations. ABAC requires more time and effort than RBAC at the deployment and configuration stage, as security administrators need to define all attributes of the system. At first, you need to assign attributes to each system component manually.

But once you’ve created policies for most common job positions and resources in your company, you can simply copy them for every new user and resource. This is similar to how a role works in the RBAC model, but in the ABAC model, attributes can be modified for the needs of a particular user without creating a new role. Attributes make ABAC a more fine-grained access control model than RBAC.

# ABAC Architecture

# ABAC / XACML Architecture & Flow

This picture highlights how ABAC works: you have the notion of an interceptor or enforcement point (PEP) which intercepts the flow between the user and the app. This enforcement point will check whether the user can get access to whatever it is they want to get access to (data, an API call, a widget…). The idea is that the PEP is local to what you are protecting but the decision making is centralized and that is what will give you **consistent** authorization. You can have PEPs for SPA, for APIs… And they can enforce the same authorization policies consistently.

The PDP or Policy Decision Point is the one that processes the authorization requests and evaluates them against a set of policies you would have previously written. The language policies are written is typically [alfa](https://stackoverflow.com/questions/tagged/alfa" \o "show questions tagged 'alfa') or [xacml](https://stackoverflow.com/questions/tagged/xacml" \o "show questions tagged 'xacml').

The PIP (Policy Information Point) is an abstract representation of your data sources and user directories (AD, DB…) where you might store additional information about the users and resources. They can be useful to help make the right decision.

# Role-Based Access Control (RBAC) vs Attribute-Based Access Control (ABAC)

Let’s compare these two popular approaches — role-based access control vs attribute-based access control — to determine the pros and cons of each.

**RBAC pros and cons**

RBAC is the most popular approach to restricting access. The main advantage of this model is that companies no longer need to authorize or revoke access on an individual basis, bringing users together based on their roles instead. Establishing a set of roles in a small or medium-sized company isn’t challenging. On the other hand, setting up such a system at a large enterprise is no easy task.

There are several **limitations to the RBAC** model. You can’t set up a rule using parameters that are unknown to the system before a user starts working. Permissions can be assigned only to user roles, not to objects and operations. Also, using RBAC, you can restrict access to certain actions in your system but not to certain data.

**ABAC pros and cons**

The key **benefit of ABAC** is that it grants access based not on the user role but on the attributes of each system component. This way, you can describe a business rule of any complexity. Even if you need to make certain data only accessible during work hours, it can be easily done with one simple policy. On top of that, ABAC rules can evaluate attributes of subjects and resources that are yet to be inventoried by the authorization system.

As for **ABAC limitations**, this type of system is hard to configure due to the way policies must be specified and maintained. It’s difficult to perform a before the fact audit and determine the permissions available to a specific user. It could be impossible to determine risk exposure for any given employee position.

Gartner **predicts** that **70% of all organizations will use ABAC by 2020**

To sum up, let’s compare the key characteristics of RBAC vs ABAC:

|  |  |  |
| --- | --- | --- |
| Characteristic | RBAC | ABAC |
| Flexibility | **checked-icon**  (For small and medium-sized organizations) | checked-icon |
| Scalability | minus-icon | checked-icon |
| Simplicity | Easy to establish roles and permissions for a small company, hard to maintain the system for a big company | Hard to establish all the policies at the start, easy to maintain and support |
| Support for simple rules | checked-icon | checked-icon |
| Support for complex rules | checked-icon | checked-icon |
| Support for rules with dynamic parameters | minus-icon | checked-icon |
| Customizing user permissions | **minus-icon**  (Every customization requires creating a new role) | checked-icon |
| Granularity | Low | High |

**Combining RBAC and ABAC**

Companies often start with implementing a flat RBAC. This model is easier to set up and maintain. As organizations grow and manage more sensitive data, they realize the need for a more complex access control system. RBAC and ABAC can be used together, with RBAC doing the rough work and ABAC complementing it with finer filtering.

This access model is also known as **RBAC-A**. There are **three RBAC-A approaches** that handle relationships between roles and attributes:

* **Attribute-centric**. A role becomes the name of one of the user attributes. It resembles a job title. The “role” attribute in such a model is used to mark a set of attributes required for a certain position.
* **Role-centric**. Attributes are added to constrain roles. In such a model, attributes can reduce permissions available to a user. This approach strengthens the security of your data.
* **Dynamic roles**. Attributes such as time of day are used to determine the subject’s role. In some cases, a user’s role can be fully determined by dynamic attributes.

In addition, there’s a new method called **next generation access control (NGAC)** that’s currently being developed by **NIST**. It’s based on ABAC but implements a more refined approach to policies. For example, NGAC supports several types of policies simultaneously, including ones that are applied both in the local environment and in the network.

# 5 Identity Management Scenarios to Study

These examples can help you understand when RBAC systems are best and when ABAC systems might work better. We've also included an example of using both together, as sometimes that works well too.

Companies should consider the question of RBAC vs. ABAC when dealing with:

**1. Small workgroups.**RBAC is best. Defining work by role is simple when the company is small and the files are few.

If you work within a construction company with just 15 employees, an RBAC system should be efficient and easy to set up.

**2. Geographically diverse workgroups.**ABAC is a good choice**.**You can define access by employee type, location, and business hours. You could only allow access during business hours for the specific time zone of a branch.

**3. Time-defined workgroups.**ABAC is preferred**.**Some sensitive documents or systems shouldn't be accessible outside of office hours. An ABAC system allows for time-based rules.

**4. Simply structured workgroups.**RBAC is best**.**Your company is large, but access is defined by the jobs people do.

For example, a doctor's office would allow read/write scheduling access to receptionists, but those employees don't need to see medical test results or billing information. An RBAC system works well here.

**5. Creative enterprises.**ABAC is ideal because creative companies often use their files in unique ways. Sometimes, everyone needs to see certain documents; other times, only a few people do. Access needs change by the document, not by the roles.

For example, the creative staff within your company, including artists and writers, create files that are easily distributed by other employees. But those employees, including billing departments and account executives, might need to see those files. And the marketing team might want to share them.

The complexity of who should see these documents, and how they are handled, is best accomplished with ABAC.

Many times neither RBAC nor ABAC will be the perfect solution to cover all the use cases you need. That’s why most organizations use a hybrid system, where high-level access is accomplished through RBAC and then fine-grained controls within that structure are accomplished through ABAC.

For example, you might use the RBAC system to hide sensitive servers from new employees. Then, you might use ABAC systems to control how people alter those documents once they do have access.

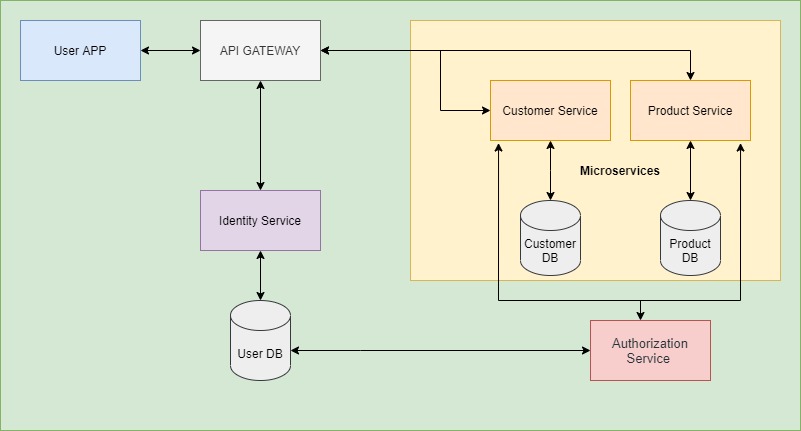
Researchers say blending RBAC and ABAC can help administrators get the best of both systems. RBAC offers leak-tight protection of sensitive files, while ABAC allows for dynamic behavior. Blending them [combines the strengths of both](https://dl.acm.org/doi/abs/10.1145/3316615.3316667).

# How to Choose an Access Control Solution

When it comes to security, it’s crucial to plan and monitor your access control processes carefully. Use a robust access management tool to help you set up your access control, and regularly review your setup to make sure it still fits your organizational needs.

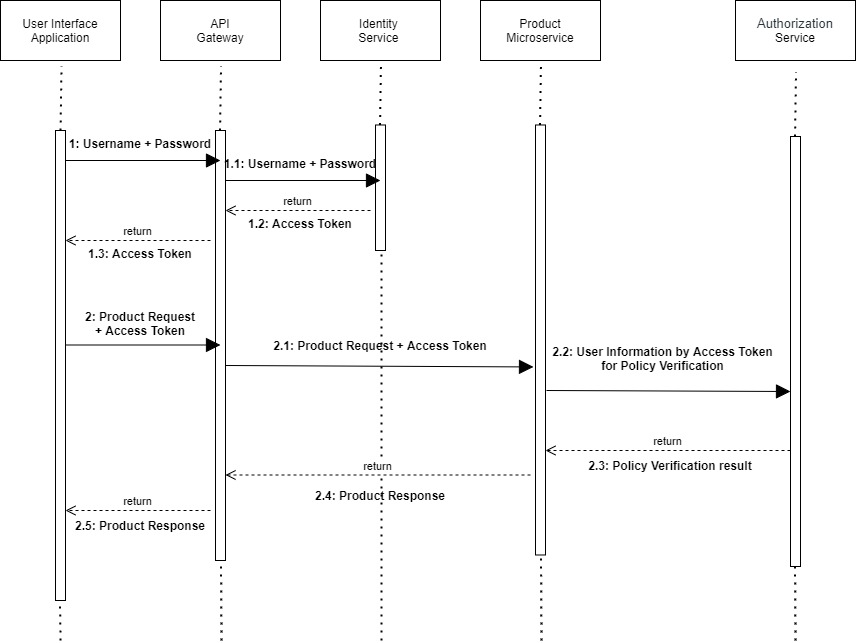
Whether you invest in Access Rights Manager make sure the tool you choose can set up a protocol and mechanism to ensure users have the correct access to what they need to do their jobs, and nothing more.

# ABAC Microservice POC Architecture Diagram



# POC implementation UML SEQUENCES Diagram

# Sequence Diagram for Product Microservice Data Access



# Sequence Diagram for Customer Microservice Data Access

# C:\Users\chetan.dravekar\Desktop\POC\UML Customer Sequences Diagram.jpg

# 

# REFERENCE DOCUMENTS

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Document Title** | **Document ID** |
| 1 | Attribute-Based Access Control (ABAC) | * <https://en.wikipedia.org/wiki/Attribute-based_access_control> |
| 2 | Role-based Access Control (RBAC) | * [https://elang2.github.io/myblog/posts/2018-09-29-Role-Based-Access-Control-MicroServices.html#:~:text=RBAC%20stands%20for%20Role%20Based,users%2C%20vendors%20and%20customers%20efficient.](https://elang2.github.io/myblog/posts/2018-09-29-Role-Based-Access-Control-MicroServices.html%23:~:text=RBAC%20stands%20for%20Role%20Based,users%2C%20vendors%20and%20customers%20efficient.) * <https://www.hebergementwebs.com/new/designing-rbac-role-based-access-control-for-microservices> |
| 3 | How this is different from Role-based Access Control (RBAC) | * <https://www.ekransystem.com/en/blog/rbac-vs-abac> * [https://www.dnsstuff.com/rbac-vs-abac-access-control#:~:text=Attribute%2DBased%20Access%20Control%20(ABAC),-Role%2Dbased%20access&text=The%20primary%20difference%20between%20RBAC,%2C%20environment%2C%20or%20resource%20attributes.](https://www.dnsstuff.com/rbac-vs-abac-access-control%23:~:text=Attribute%2DBased%20Access%20Control%20(ABAC),-Role%2Dbased%20access&text=The%20primary%20difference%20between%20RBAC,%2C%20environment%2C%20or%20resource%20attributes.) |

# 

# 