

# A Guide to Authorization:

## A Discussion of New Best Practices Using Hybrid Role and Attribute Techniques

Patrick Parker, CEO EmpowerID



# CONTENTS

<b>INTRODUCTION .....</b>	<b>1</b>
<b>HOW APPLICATIONS MAKE DECISIONS .....</b>	<b>2</b>
<b>EXTERNALIZING DECISION MAKING .....</b>	<b>5</b>
External Authorization Components .....	6
Centralized Versus Decentralized External Authorization.....	7
<b>ROLE-BASED ACCESS CONTROL (RBAC) .....</b>	<b>8</b>
RBAC Authorization Model.....	8
The RBAC Anti-Pattern .....	9
“Resource”-Based RBAC.....	12
Role Explosion Challenge.....	14
RBAC Benefits.....	15
RBAC Weaknesses .....	16
<b>ATTRIBUTE-BASED ACCESS CONTROL (ABAC).....</b>	<b>16</b>
ABAC Benefits.....	17
ABAC Primary Benefits .....	17
ABAC Weaknesses .....	19
ABAC Challenges.....	22
RBAC Versus ABAC: Tradeoffs and Balance.....	23
<b>POLICY-BASED ACCESS CONTROL (PBAC) .....</b>	<b>24</b>
What is PBAC? .....	25
<b>CHALLENGES WITH EXTERNAL AUTHORIZATION .....</b>	<b>27</b>
<b>HOW ARE POLICIES ENFORCED? .....</b>	<b>28</b>
Self-Enforcement.....	29
Claims-Based Enforcement .....	30
Gateway Enforcement.....	31
Open Policy Agent (OPA).....	32
Provisioning Engine Enforcement .....	34
<b>EMPOWERID’S HYBRID ROLE AND ATTRIBUTE COMPLIANT ACCESS APPROACH .....</b>	<b>36</b>
What is Compliant Access Delivery .....	36
How Does Compliant Access Leverage Your Business Model? .....	37
Role-Based Access Control .....	39
Role Mining and Optimization.....	46
Compliant Access Self-Service (Exceptions Management).....	53
Compliant Access Recertification .....	57
Compliant Risk Management .....	58
<b>CONCLUSION .....</b>	<b>60</b>
<b>REFERENCES.....</b>	<b>60</b>

## TABLE OF FIGURES

Figure 1: Generic overview of authorization .....	2
Figure 2: Simple example of authorization .....	2
Figure 3: Application Decision Making Process .....	3
Figure 4: Authorization Decision Flow Within Application .....	4
Figure 5: Local Hard-Coded Security Checks Are Unauditable .....	4
Figure 6: Lack of Auditability Becomes a Large Problem Due to Number of Apps.....	5
Figure 7: Externalized/Centralizing Decisionmaking Process .....	6
Figure 8: Logical Components of External Authorization .....	7
Figure 9: The Three Possible Locations for Application Decision Making .....	7
Figure 10: Decoupling of Questions and Decision Making .....	8
Figure 11: Independence of Question Interfaces and PDP Policy Models.....	9
Figure 12: RBAC PDP Flow Determining Access to Bob’s X-Ray for Alice .....	9
Figure 13: Pseudocode example of simplistic “IsInRole” approach.....	10
Figure 14: Expanded Pseudocode example of the problem with “IsInRole” approach.....	11
Figure 15: Pseudocode example of roles without resources.....	12
Figure 16: Pseudocode example of roles with rights for resources .....	13
Figure 17: Role definition showing rights granted and scopes used in the assignment.....	14
Figure 18: Example of the Role Explosion Challenge.....	15
Figure 19: Role explosion due to the creation of many Teller Business Roles .....	15
Figure 20: NIST SP 800-162 ABAC Definition .....	17
Figure 21: Elaborate ABAC policy showing the potential of ABAC .....	18
Figure 22: Typical Presentation Slide Showing Audience Response.....	19
Figure 23: Sources of attribute data for a complex runtime ABAC policy evaluation .....	20
Figure 24: A Common Standard Auditor's Question.....	21
Figure 25: The Who, What, and How of ABAC Challenges .....	22
Figure 26: Examples of authoring tools and languages for XACML policies .....	23
Figure 27: Example of a raw XACML policy editor .....	23
Figure 28: Balance between the benefits and limitations of attribute and role approaches.....	24
Figure 29: Information Technology – Next Generation Access Control – Functional Architecture (NGAC-FA).....	25
Figure 30: Example of a PBAC Get Permissions Decision .....	26
Figure 31: Example of a Full PBAC Access Check .....	27
Figure 32: Will our critical apps use this new external decision engine? .....	27
Figure 33: Self-Enforcement – External Authz-Enabled Apps.....	29
Figure 34: Example of User-Managed Access (UMA) .....	30
Figure 35: Example of Claims-Based Enforcement for AWS.....	31
Figure 36: Example of Gateway Enforcement .....	32
Figure 37: Example of OPA Enforcement Model .....	33
Figure 38: Example of Cloud Native OPA Authorization.....	34
Figure 39: Example of Provisioning Engine Enforcement .....	35

Figure 40: EmpowerID’s Innovative Hybrid RBAC/ABAC/PBAC Model..... 36

Figure 41: Business Function Mapping to IT Roles and Rights in external Systems ..... 39

Figure 42: Conceptual Model of Function Mapping to Technical Access ..... 39

Figure 43: Illustration of the layers of an optimized best practice EmpowerID role implementation ..... 41

Figure 44: EmpowerID’s polyarchical RBAC approach showing people and their Business Roles and Locations ..... 42

Figure 45: Role Explosion Solution – one Teller Business Role scoped on assignment by Organizational Location ..... 43

Figure 46: IAM Acting as an Anti-Corruption Layer Insulating the Business Model from Technical Changes and Limitations ..... 44

Figure 47: Venn diagram of the 3 types of T-RBAC Management Roles and how they combine to enable task-based access..... 45

Figure 48: Logical illustration of the RBAC mapping concept..... 47

Figure 49: RBAC Mapper showing external role mappings ..... 48

Figure 50: Top-Down Analytical Role Mining Example – suggested match to convert the DE Sales group to a role-based assignment for the Sales in Munich Business Role and Location ..... 50

Figure 51: Role Visualizer showing examples of candidate functional and global roles ..... 51

Figure 52: EmpowerID's IT Shop ..... 54

Figure 53: Eligibility policy being applied to a Person ..... 55

Figure 54: Access Request Policy time-restriction settings ..... 56

Figure 55: Manager view of a direct report recertification viewing their Business Function access ..... 57

Figure 56: Risk Policies..... 58

Figure 57: Risk Violations..... 59

## INTRODUCTION

Authorization is the security mechanism by which systems and applications determine a user's privileges while using the system. In recent years, tremendous strides have been made around Authentication with the maturation of federated protocols and standards such as FIDO2 for secure passwordless and multi-factor authentication. Confirming a user's identity with a high degree of certainty and frequently reaffirming it is a crucial component of the "Zero Trust" security model.

Identity is the "new perimeter" (or control plane) for our multi-cloud world—and a new approach was required. This new approach is dubbed "Zero Trust" and is focused on identity. Zero Trust is not a formal standard but rather a set of practices and an approach that arose out of the realization that the front line of the cyber battlefield is now being fought at the identity level. The Zero Trust paradigm accepts that our world is no longer comprised of simple internal/external = trust/no trust equations and, as such, mandates three fundamental principles: never trust, continuously verify, and always enforce least privilege. The never trust and always verify principles are handled by authentication. The last and equally important principle to always enforce least privilege fits squarely in the domain of Authorization.

The least privilege principle addresses authorization and states that an individual should have only the minimum access privileges required to perform a specific job or task and nothing more. The most apparent benefit of enforcing least privilege is that it reduces an organization's attack surface, making it harder for attackers to find privileged credentials and offer them fewer capabilities to perform malicious activities when using a compromised privileged account.

The central importance of authorization and the complexity in adequately managing the day-to-day activities to control who has access to what has dramatically increased as organizations have undergone digital transformations moving more of their business online and into the Cloud. Recent studies have shown that through 2023 over 99% of Cloud security failures will be the customer's responsibility. At least 75% of those failures will result from inadequate management of identities and their authorizations.

*Paul Mezzera, Gartner*

---

*"Cloud Management consoles, like those on Azure, AWS, and GCP, pose significant security risks to every company if not managed well. Over-privileges are a common fact and most organizations lack granular visibility into whether privileged users have unnecessary entitlements and provisions."*

*Martin Kuppinger, Founder and Principal Analyst at KuppingerCole*


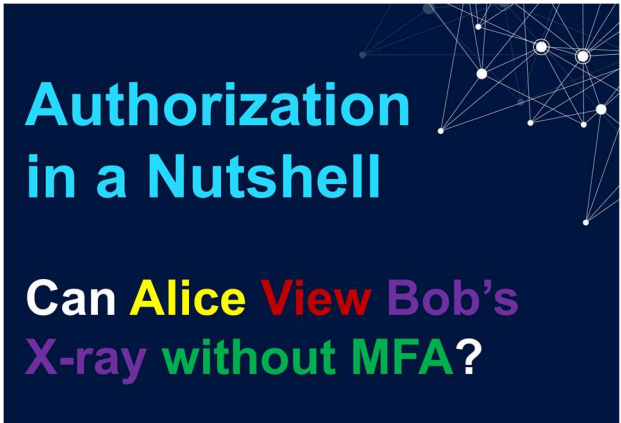
---

The responsibility to ensure adequate controls over an organization’s Cloud-based resources is primarily the customer's responsibility. In his now-famous response to the US Senate about the Capital One data breach, AWS’s CISO Stephen Schmitt stated that “even if a customer misconfigures a resource, if the customer properly implements “least privilege policy,” there is relatively little an actor has access to once they are authenticated — significantly diminishing the customer’s risk.”. This statement highlights the criticality of maintaining visibility and control over authorizations across this diverse multi-cloud landscape and enforcing least privilege as the cornerstone of a Zero Trust strategy.

In this white paper, we will look at how applications make decisions and externalizing decision-making. Then we will analyze and discuss the various attribute-based and role-based approaches for application authorization, including Role-Based Access Control (RBAC). We then discuss the RBAC versus Attribute-Based Access Control (ABAC) models and introduce a new approach known as Policy-Based Access Control (PBAC). Each model is discussed and illustrated with appropriate examples highlighting their strengths, weaknesses, and, where appropriate, applicable and inapplicable scenarios and use cases.

## HOW APPLICATIONS MAKE DECISIONS

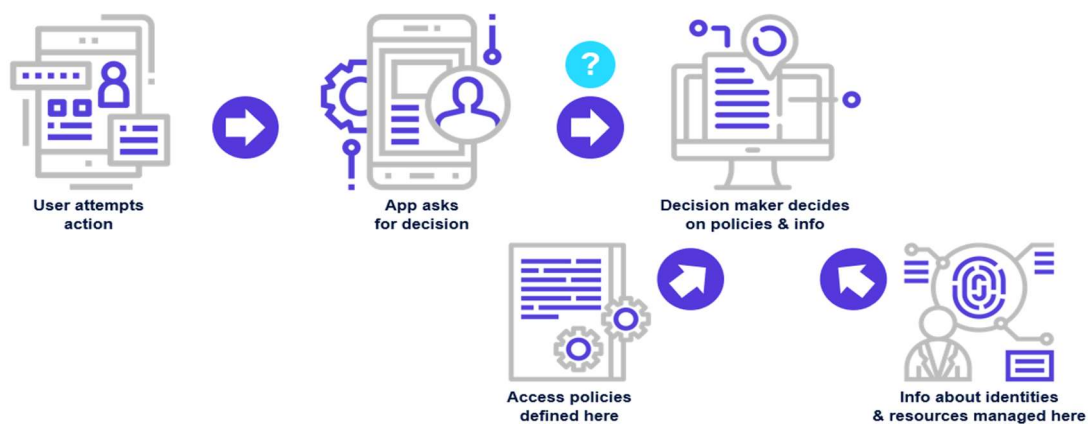
Authorization usually follows authentication to control what a user may see and do within the application. The question asked by modern applications is, “Can the subject perform the action to the resource given the context?”. More traditional applications ask the same question but without the ability to evaluate contextual information. A simple example of this is shown in Figures 1 and 2 below.

	
Figure 1: Generic overview of authorization	Figure 2: Simple example of authorization

It is beneficial to know how applications make decisions to allow or disallow users from either accessing data or performing their various functions to understand authorization. As shown in Figure 3 below, we have broken the process into easy-to-follow steps. The process itself starts when:

1. The user attempts to perform an action within the application. In this example, Dr. Alice is attempting to view patient Bob's X-Ray.
2. The application performs a check to decide whether to allow this action and will query its decision-making function accordingly. This check can be either internal or external.
3. The decision-maker then queries all access policies or definitions combined with information about the user's identity, the action they are trying to perform, and which data or protected function within the application they are trying to perform.
4. The decision is made once the evaluation is complete.

This is a generic description of the process within any application.



**Figure 3: Application Decision Making Process**

To date, most applications written will execute this authorization flow completely within the walls of the application. Typically, each application will have its own permissions, roles, and assignments defined within database tables. As shown in Figure 4 below, the application itself is self-contained and neither any centralized decisioning service nor access policy authoring happens outside the application.

## THIS CAN HAPPEN LOCALLY INSIDE THE APPLICATION



Figure 4: Authorization Decision Flow Within Application

The challenge created by a lack of centralization in application decision-making is that every application is its own security *island* (or *black box* – as shown in Figure 5, below). The result being that to see what is going on and who is allowed to do what, your security team is forced to audit each application. However, experience shows that this situation can be much worse if your applications contain hard-coded role checks. In such cases, it is likely that black boxes are often undocumented, and, having accumulated years of changes, the application developers are no longer with the company. Moreover, authorization decisions within the application are limited to simplistic policies involving only information and user attributes within the application itself.

## YOUR APPLICATION SECURITY BECOMES A BLACK BOX



```

{
  if (User.IsRole('Accountant')) {
    User.IsRole('Finance_Head')
    User.IsRole('HRM_AP_Interna')
    User.IsRole('HRM_Processors')
  }
  else {
    Msg.Text = "You are not authorized to access invoice records."
  }
}

if (User.IsRole('Accountant')) {
  User.IsRole('Finance_Head')
  User.IsRole('HRM_AP_Interna')
  User.IsRole('HRM_Processors')
}
else {
  Msg.Text = "You are not authorized to access invoices."
}

if (User.IsRole('Finance_Head')) {
  User.IsRole('HRM_AP_Interna')
  User.IsRole('HRM_Processors')
}
else {
  Msg.Text = "You are not authorized to access invoice records."
}

if (User.IsRole('Accountant')) {
  User.IsRole('Finance_Head')
  User.IsRole('HRM_AP_Interna')
  User.IsRole('HRM_Processors')
}
else {
  Msg.Text = "You are not authorized to access invoices."
}

if (User.IsRole('Accountant')) {
  User.IsRole('Finance_Head')
  User.IsRole('HRM_AP_Interna')
  User.IsRole('HRM_Processors')
}
else {
  Msg.Text = "You are not authorized to access invoice records."
}

if (User.IsRole('Accountant')) {
  User.IsRole('Finance_Head')
  User.IsRole('HRM_AP_Interna')
  User.IsRole('HRM_Processors')
}
else {
  Msg.Text = "You are not authorized to access invoices."
}
    
```

APPLICATION ACCESS CONTROLS

Figure 5: Local Hard-Coded Security Checks Are Unauditable

Even more challenging today is the number of applications used by the average user is expanding rapidly. This proliferation of applications makes maintaining control, auditability, and visibility over who has access to what is extremely challenging. Furthermore, with organizations employing so many black-box



applications, the necessity to manually maintain and audit them regularly is proving daunting, labor-intensive, and expensive. Ultimately, at some point, it will become logistically unfeasible (Figure 6).

## AND YOU'LL HAVE MANY BLACK BOXES

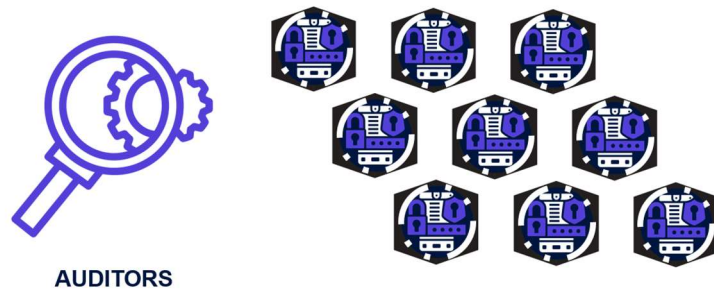


Figure 6: Lack of Auditability Becomes a Large Problem Due to Number of Apps

## EXTERNALIZING DECISION MAKING

It is easy to see the problem with hard coding application permissions locally within applications. Hardcoded permissions not only present a challenge to security teams but is also extra work for helpdesk staff. Consequently, it rapidly becomes clear that the solution for many security and audit challenges would be for application developers to externalize their applications' security decision-making. Doing so would allow centralized management of all application authorization policies and access assignments in one auditable system.

In the externalized decision-making model, as shown in Figure 7, and using our previous example of whether Dr. Alice could view Bob's x-ray, our application would ask an external source for a decision as to whether this action would be permitted. As policies and access assignments are defined and managed centrally, auditors have the dual ability to easily see who has access to do what across all applications and validate that this access is job appropriate for that person's role. This auditability makes everyone's tasks easier and lowers the administrative burden, makes the organization's life much smoother, and represents a considerable reduction in risk.

# EXTERNALIZING/CENTRALIZING DECISION MAKING

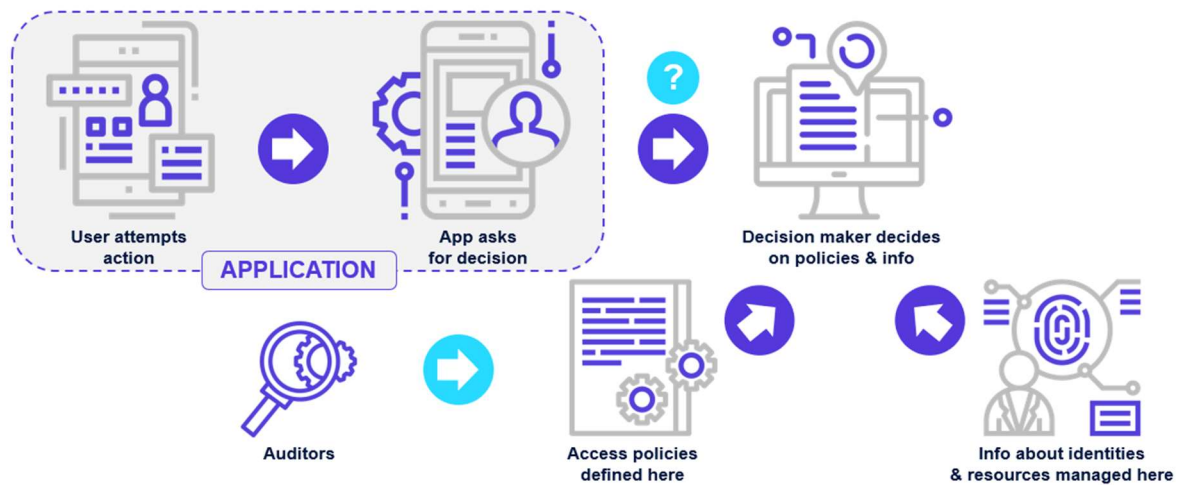


Figure 7: Externalized/Centralizing Decisionmaking Process

## External Authorization Components

External authorization is a well-established concept with a defined terminology for the various system components or actors involved in the policy authoring, decision making, and decision enforcement process. In conjunction with Figure 8, the most basic components of external authorization are as follows:

- **Policy Administration Point (PAP)** – is used to define and author the policies that are evaluated when making access decisions.
- **Policy Information Point (PIP)** – are the sources of information used in authoring the policies and making runtime decisions. PIPs can contain information about the user or subject, the resource, and the environmental conditions or context.
- **Policy Decision Point (PDP)** – receives the authorization question and then provides an answer based on either the policies contained in the PAP, the information provided by the PIPs, or is included in the request itself.
- **Policy Enforcement Point (PEP)** – enforces the PEP's decision and either permits or prohibits the user from performing the intended action against the resource in question.

As mentioned previously, for applications that do not support external authorization, all or most these functions are performed within the application itself. For example, the PDP and PEP functions are performed internally, whereas some applications rely on external directories for the PIP component.

## EXTERNAL AUTHORIZATION COMPONENTS



Figure 8: Logical Components of External Authorization

### Centralized Versus Decentralized External Authorization

As shown in Figure 9 below, externalized authorization does not require the centralization of authorization decisions. Though centralized decision-making has traditionally been the most common model for real-time externalized authorization, it is neither a requirement nor always the best fit. Indeed, a more distributed or decentralized model is a much more viable solution in some use cases. In others, when based on network topology or performance requirements, decentralized decision-making with multiple PDPs that are local to the applications or microservices they protect might be better. Distributed PDPs can still support the goals of centralized visibility, auditability, and management when combined with a centralized PAP model.

### LOCUS OF DECISIONING

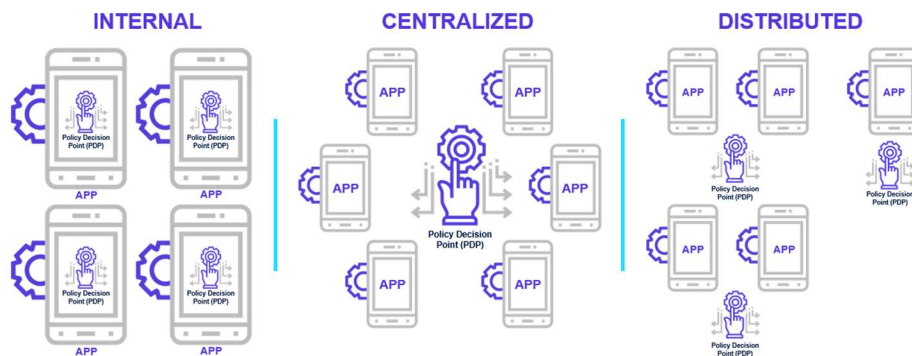


Figure 9: The Three Possible Locations for Application Decision Making

We look at the challenges with external authorization on page 27.

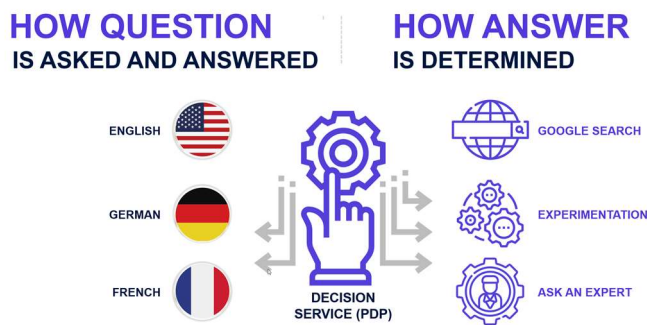
## ROLE-BASED ACCESS CONTROL (RBAC)

Role-Based Access Control (RBAC) is a security and authorization model for securing access to computer resources. Almost all enterprises use RBAC to secure their systems. Sometimes referred to as “Role-Based Security,” RBAC access is based on roles defined by the business using them. In the RBAC model, roles are created, and then sets of permissions for resources are assigned to the role. Users are then granted one or more roles to receive access to resources.

### RBAC Authorization Model

Though external authorization and real-time decisioning have become synonymous with attribute-based approaches, they are independent of the technology type of policies used. Since the central premise of real-time external authorization is that the application will call at runtime for a decision from an external PDP, it can support almost any model. How the PDP makes its decisions and how those policies are managed and defined does not require an attribute-based policy model.

If we were to use a real-world analogy to explain this decoupling, Figure 10, below, shows that the language in which a question is asked does not mandate the approach used to locate and determine the answer.



**Figure 10: Decoupling of Questions and Decision Making**

By extending this non-technical analogy to external authorization, our authorization service could expose various interfaces for calling applications to ask for authorization decisions from our PDP. The interfaces merely define the protocols and structures that support the different types of applications or scenarios required. On the back end, because our PDP is decoupled from the interface through which the question is being asked, it has no limitations on the number of authorization policy types used to determine the answer (Figure 11).

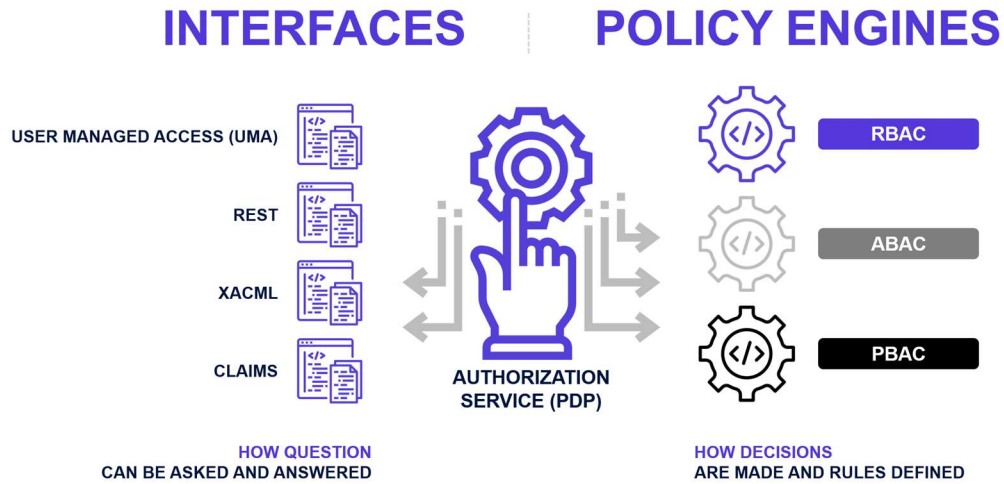


Figure 11: Independence of Question Interfaces and PDP Policy Models

In Figure 12 below, we continue our previous example where the application asks if Alice can view Bob’s X-Ray. Here, the decision is being made by a centralized RBAC system that leverages roles and assignments. The RBAC-based PDP checks if user ‘Alice’ has any roles in an assignment that grants her the View permission for Bob’s X-Ray. In this instance, because she is a member of the Doctors role and that role has a specific assignment granting the right to view Bob’s X-Ray, we can see the answer is yes.

## EXTERNAL AUTHORIZATION WITH RBAC

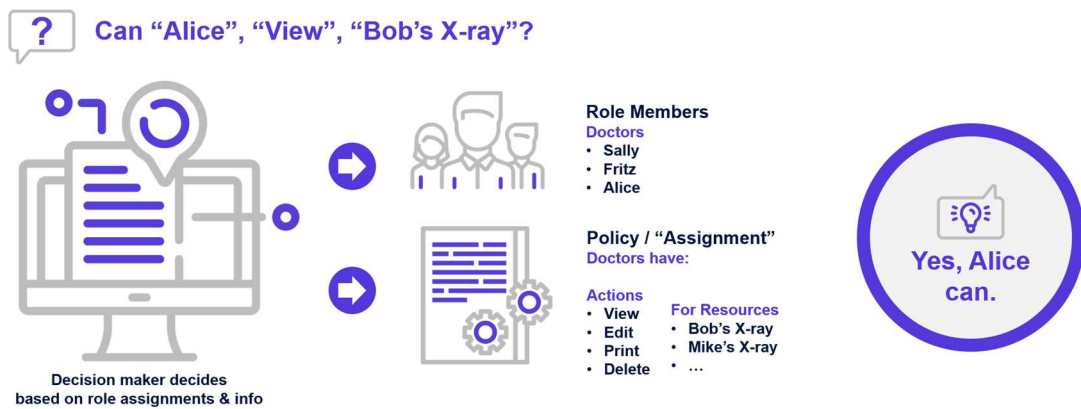


Figure 12: RBAC PDP Flow Determining Access to Bob’s X-Ray for Alice

### The RBAC Anti-Pattern

Much like ABAC’s Subject, Resource, and Action model, the RBAC model defines a Subject, a Role, and a Permission concept. In the RBAC model, a Permission is the combination of ABAC’s an “Action” for a “Resource,” e.g., Bob’sX-Ray.View.

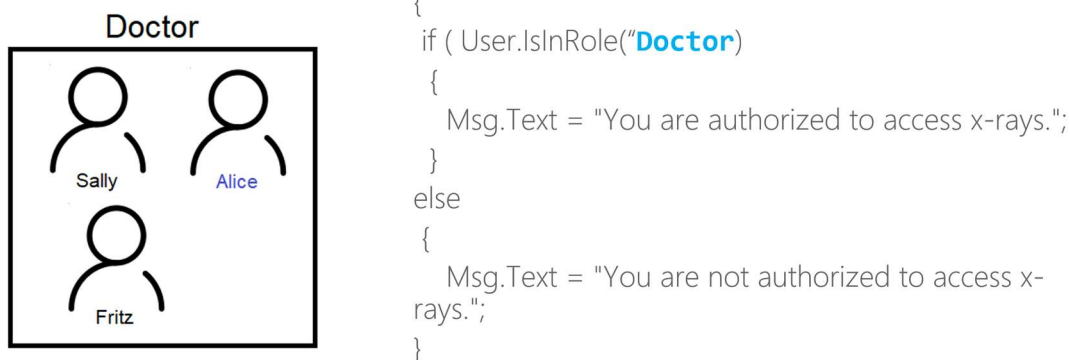
However, because of the way developers write them, most RBAC implementations limit themselves to simple checks for role membership without any concept of permission. This almost universally oversimplified implementation by application developers is RBAC's greatest weakness. Indeed, RBAC has been around since at least 1992. If you were to do a Google search for "RBAC" and read the resulting vendor developer documentation on how to implement it, you would find examples that implement what could be described as an RBAC "anti-pattern."

This anti-pattern implementation not only downgrades roles into being simply groups that maintain collections of users with the central role management system, but this also makes them unaware of the permissions they grant in the applications.

We illustrate this in Figure 13 below. Here, we have a role named Doctor with Sally, Alice, and Fritz as members. As a developer, I can write a role check code that says if the user is in the doctor's role, they are permitted to execute this feature or use this part of my application. Conversely, if they are not in the Doctor role, they will automatically be denied access and see the corresponding access denied message. This 'Hello world' example has been out there forever and is long overused and too simplified.

This oversimplification is so widespread that it has become synonymous with RBAC and even promoted as a standard practice by many vendors. In this model, application permissions are enforced solely internal to the application and hidden from the role management system. This makes it impossible to manage or audit the actual access being granted by roles and opens the possibility for security vulnerabilities and abuse.

## Hard Coded Role Checks Start Like This



**Figure 13: Pseudocode example of simplistic "IsInRole" approach**

Moreover, this approach is not only flawed but also creates additional challenges. For example, organizational security policies and the IT environment are dynamic and continually changing.

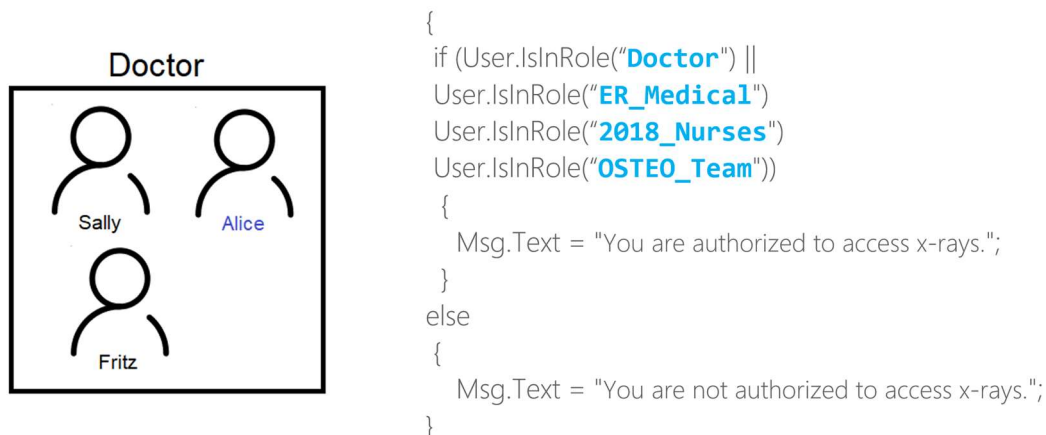
Extending our previous example, and as illustrated in Figure 14 below, the application developer receives a request to permit the ER Medical Team members to execute the same function as the Doctors. To implement this change, they would need to open the code to make this modification and then either

recompile or at least be required to modify, config files and redeploy the application. Subsequently (in our example, this occurs six months later), two additional requests for granting access arrive the first to add the 2018 nursing interns and the second to add the Osteopathic team.

Now that the 2018 intern season has long since passed, it is unlikely that a follow-on request will come to revoke their access, and the code will remain as-is. Such limitations are why organizations end up with hard-coded access rules that are unmanageable and eventually lead to no one understanding whom access was granted, why, and even if it or they are still applicable and relevant.

As we can see in the pseudocode example, in Figure 14, any time the rules for who may perform an action change, a new role needs to be created and added to the hardcoded list of roles checked by the application's *IsInRole* statements. Furthermore, when a role's permissions become more restrictive, all application code referencing them would need to be removed and recompiled. This process would also need to be repeated with each new policy change. Consequently, with even the smallest changes, the IT team would need to revisit the application and invest significant resources each time. Moreover, writing role checks into source code lacks the transparency necessary for maintaining a useful audit trail. As shown in the example code below, because this method lacks a clear external link between the X-Ray resource and the roles granting access to it, assignments like this cannot be audited. Furthermore, as organizations and personnel evolve, correct role and resource access become no more than tangled confusion.

## But Soon Become Like This



**Figure 14: Expanded Pseudocode example of the problem with "IsInRole" approach**

Given that the anti-pattern approach to RBAC is so widespread then, quite clearly, the *IsInRole* approach is not RBAC. "Roles" without Rights are not Roles; they are Groups. In our previous examples, the written logic to check if users were members of the Doctors role did not involve any type of permissions or application rights checks. Because the logic simply granted access to the doctor role members, it reduced it to a mere group with a membership check—and this is not truly an RBAC role with permissions. Such widespread misuse of the term RBAC for cases where group membership is checked has led many to state that roles are

just attributes like any other. Though this is an accurate statement, it is only because this is not RBAC. Having examined the 'Hello world' RBAC anti-pattern, we can now examine several examples where we add additional layers of sophistication to implement true RBAC.

In Figure 15 below, by defining the permission that role members would be granted for our X-Ray application, our Doctor role is now a proper RBAC role. The Doctor role now grants its members the Read, Edit, Print, and Delete permissions. Also noteworthy is that our developer's access check is more flexible and, though it demands that the user must have specific permission, it does not name a hard-coded role. The advantage now is that multiple configurable roles could deliver this permission to their members, granting greater flexibility as both organizations and permissions can evolve to match business changes. Despite this, our RBAC example is still limited by lacking granularity or context for the role members that receive the permissions. In this example, permissions are global to the application and not checked against a specific page, function, or data record within the application. The result is that, without any limits or context, any role member with a role granting the Delete permission will have this right anywhere within the application.

## Roles With Rights "No Resource"

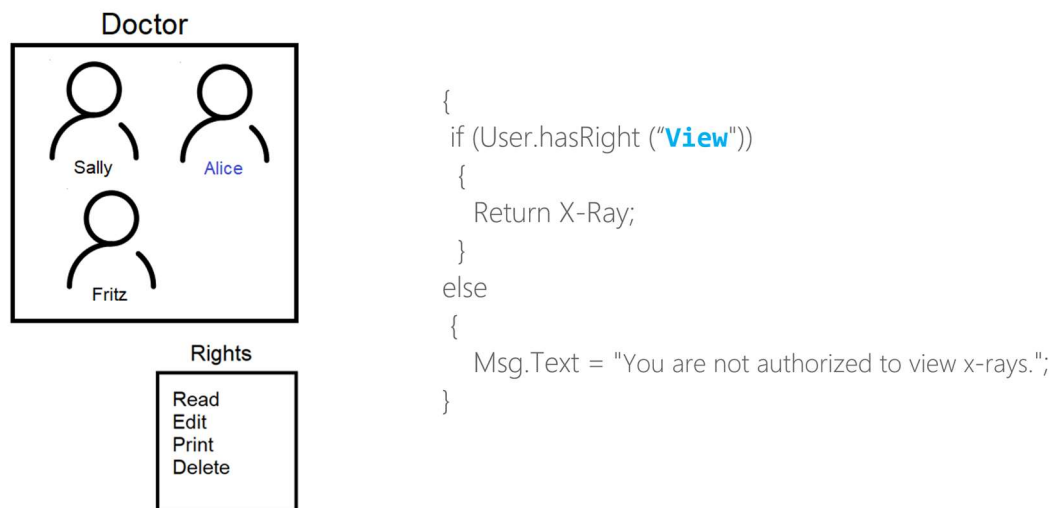


Figure 15: Pseudocode example of roles without resources

### "Resource"-Based RBAC

In our previous examples, there was a lack of contextuality or granularity in the assignment of permissions granted by our roles. However, we can extend our RBAC model by adding one more level of sophistication. In Figure 16 below, we've expanded our example by allowing the role to be associated with specific defined resources that grant members the correct permissions. Because of the addition of a more granular resource-level scoping of role access versus non-granular application-wide permissions, this variant of RBAC is sometimes referred to as "Resource"-Based RBAC.



The Resource"-Based RBAC model allows systems such as Microsoft SharePoint to grant users flexible roles that are scoped for specific sites, folders, or other secured objects. In these applications, the access check validates whether or not the user's role has both scope and right to manage a particular resource.

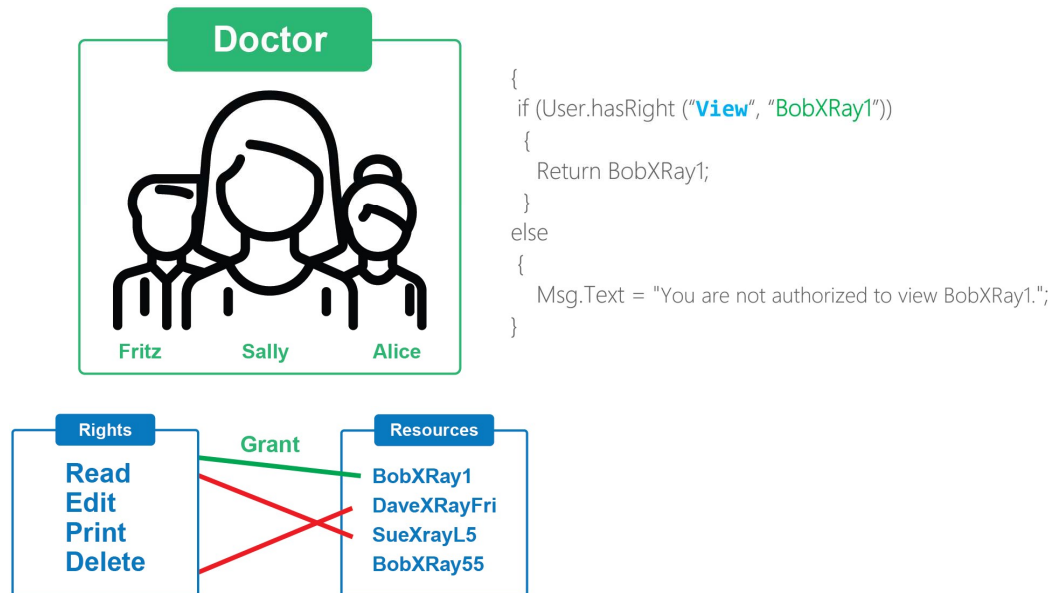


Figure 16: Pseudocode example of roles with rights for resources

In this Resource-based RBAC model, the concept of "Role Definitions" becomes quite important. Role definitions are "assignable" bundles of permissions. The role definitions themselves are not granted access to the resource but rather used in assignments to grant users, groups, or other security principals bundles of permissions for either specific or all resources within a path or "scope." As seen in Figure 17 below, the role definition defines permissions granted in assignments. These assignments grant the security principal the rights for the resources or scope below which the rights are conveyed.

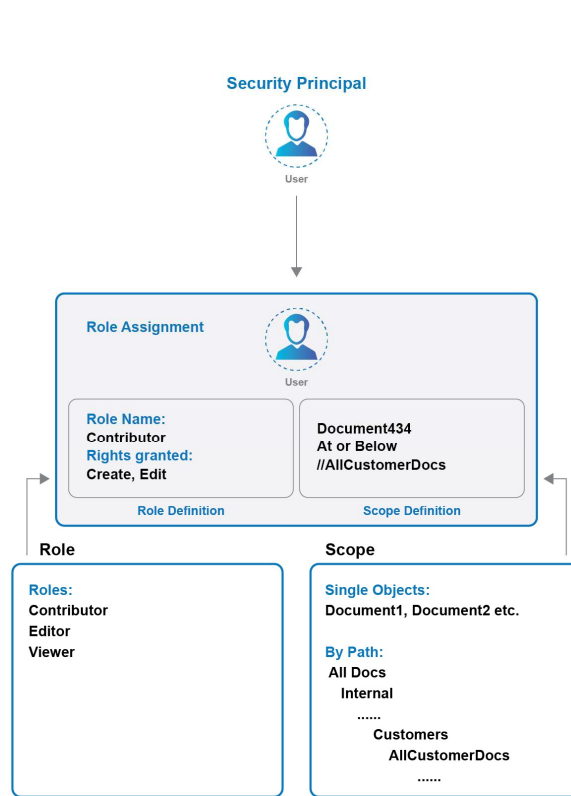


Figure 17: Role definition showing rights granted and scopes used in the assignment

### Role Explosion Challenge

Probably the most well-known problem with RBAC is “role explosion.” Organizations often end up with large numbers of roles to accommodate people performing the same job function within different geographical or functional areas within the company. Organizations are forced to duplicate and manage many very similar roles to accommodate the slight differences between these slight locational differences. This role duplication is known as “role explosion.” Unfortunately, organizations with an inflexible RBAC system will end up creating and managing thousands of roles, be forced to build roles for each simple access case, and their administration problem will multiply and compound. For example, as Figure 18 shows below, a simple organization with 20 job titles and 20 locations may require up to 400 separate roles.

A Simple Organization with Only 20 Job Titles and 20 Locations

		Locations / Departments / Cost Centers					
		New York	London	Sydney	Germany	Japan	...→20
Job Titles	Teller	TLR_NY	TLR_LN	TLR_SY	TLR_DE	TLR_JP	
	CSR	CSR_NY	CSR_LN	CSR_SY	CSR_DE	CSR_JP	
	Developer	Dev_NY	Dev_LN	Dev_SY	Dev_DE	Dev_JP	
	Accts Payable	AccPay_NY	AccPay_LN	ACC_SY	AccPay_DE	AccPay_JP	
	...→20						

Requires up to 400 separate roles to be managed

Figure 18: Example of the Role Explosion Challenge

To further illustrate this problem, we will consider how both models address managing resource access needs for an employee familiar with banking institutions—the Teller. Most bank tellers perform and access similar functions and systems to perform their daily duties. Using RBAC to manage their access sounds relatively straightforward: First, you create a “Teller” role, then you assign all tellers to the Teller role, and finally, you assign the access need to the Teller role. At this point, everything appears excellent because only one role is needed. However, what if the banking institution has branches located in multiple cities, regions, and even countries? Although each Teller needs access to a shared pool of resources, they would also need access outside the shared pool determined by their location, i.e., tellers in New York would require access specific to New York, but not to London or Sydney. Tellers in London would need access to resources in London, but not New York or Sydney. Likewise, tellers in Sydney would require access to resources in Sydney, but not New York or London. In this case, using the same Teller role for all tellers is problematic because doing so would create a “super role”: a situation where giving each teller access to resources beyond their scope violates not only an organization’s risk policies but also the concept of Compliant Access.

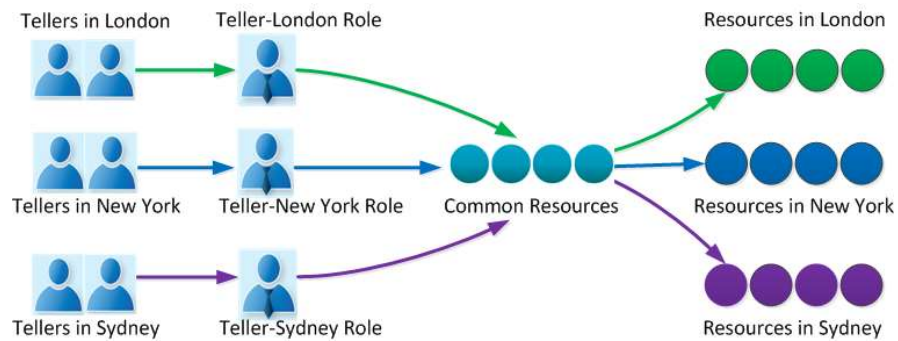


Figure 19: Role explosion due to the creation of many Teller Business Roles

**RBAC Benefits**

In the RBAC model, since access is not directly assigned to users but bundled into assignments made to roles, the correct assumption is that controlling and maintaining access is easier. Moreover, because roles and access management can be centralized, it is evident who has a role and access the role grants. There are

fewer assignments to be managed, which decreases the cost of security management and compliance auditing. Furthermore, according to a 2010 NIST study [1], correct RBAC implementation and efficient provisioning can also reduce employee downtime resulting in significant ROI.

### RBAC'S Primary Benefits:

1. RBAC is deterministic. An RBAC approach makes it easy to know who has access to what at any moment in time.
2. RBAC is more direct and easier to visualize. Security admins can visualize the actors and resources they will affect when creating or modifying a policy.
3. RBAC is inherently auditable. With RBAC assignments, as the consequences of that access are visible, it is simple for business owners to certify or attest to access granted. This visibility contrasts with ABAC where a "before the fact audit" is impossible and the effects of a rule are difficult to ascertain.
4. RBAC can be simpler than ABAC. For example, with RBAC, bundles of access can be directly assigned to a user. To do this in ABAC requires the creation of a new rule.

### RBAC Weaknesses

Unfortunately, the precise nature of the RBAC model can also be considered the source of many of its weaknesses:

- RBAC requires advanced knowledge of the Subjects and Resources and, typically, does not support making on-the-fly contextual decisions.
- An RBAC-only approach can result in an enormous number of roles to accomplish fine-grained authorization.
- Resource owners must know something about the roles and their intended purpose to grant access to those roles accurately.
- Resources must be organized into collections to facilitate delegation.
- Given a substantial number of roles and collections of resources, a correspondingly large number of delegations would need to be created and managed.

## ATTRIBUTE-BASED ACCESS CONTROL (ABAC)

Attribute-Based Access Control (ABAC): An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

*"The policies that can be implemented in an ABAC model are limited only to the degree imposed by the computational language and the richness of the available attributes." [2]*

Essentially, the more decision data at your disposal at runtime, the more sophisticated your ABAC policies can be.

ABAC relies on user attributes for authorization decisions. ABAC policies are rules that evaluate access based upon the following four sets of attributes:

- *Subject* – the attributes concerning the person or actor being evaluated.
- *Resource* – the attributes of the target or object being affected.
- *Action* – describe the action to be performed on the *Resource*
- *Environment* – includes attributes such as the time of the day, IP subnet, and others that do not relate to either the *Subject* or the *Resource*.

Figure 20 the NIST SP 800-162 ABAC Definition:

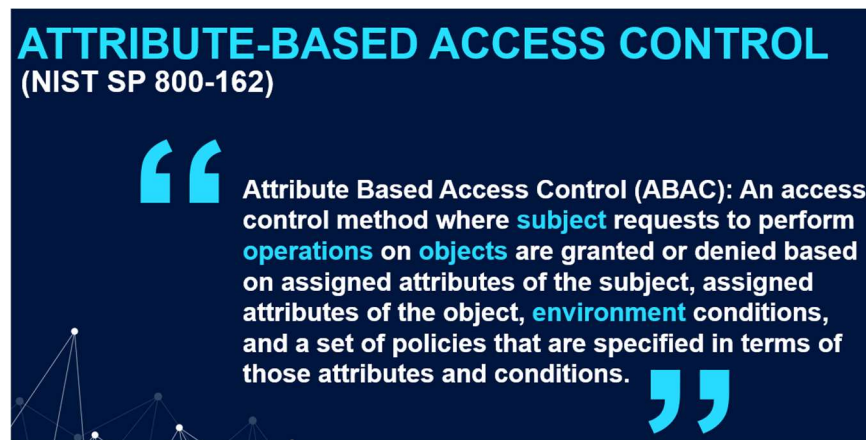


Figure 20: NIST SP 800-162 ABAC Definition

### ABAC Benefits

The key advantage of ABAC is that it does not allow application developers to hardcode a static list of roles and oversimplify their authorization source code. Rather, ABAC forces them to centralize all authorization decisions and call out at runtime to decide based on the *Subject*, *Resource*, *Action*, and *Environment* request attributes. Another key benefit is ABAC's sometimes simpler nature. This simplicity can make it easier to understand how a rule grants access to a resource when dealing with a small number of rules. In contrast, RBAC does seem foreign to many users, and, especially during the early phase of its adoption, the levels of abstraction can be challenging for an IT team.

One other added advantage of ABAC is its flexibility. With ABAC, as long as the necessary data is available, almost anything can be represented as a rule-based query. For example, a rule evaluated at runtime in a login session can use contextual information—even information passed in via SAML claims or JWT tokens. In contrast, when delivering the role membership for a user to the application, a standard RBAC engine would not evaluate this type of information.

### ABAC Primary Benefits

1. ABAC enforces centralized management of authorization policies.

2. ABAC makes it easy to specify access rules as simple queries.
3. ABAC rules can be extraordinarily fine-grained and contextual.
4. ABAC rules can evaluate attributes of Subjects and Resources that are not inventoried by the authorization system.
5. ABAC rules need less maintenance and overhead because they do not require the creation or maintenance of the structure on which an RBAC model depends, e.g., roles and resource locations.

Figure 21, below shows an example of an actual ABAC policy implemented in an EmpowerID demonstration. Here, our application is asking if Alice can View Bob’s X-Ray. Our ABAC decision engine will base its decision on an elaborate ABAC policy (labeled Policies in the figure). The first policy check, Rule1, is that the action being taken is ‘View.’ The next extended requirement for Rule1 is that the company is not in ‘Emergency Mode,’ i.e., in the middle of a crisis. However, to further extend the rule flexibility during a crisis, some roles may be granted additional permissions to enable those who would typically not be permitted to perform those activities to assist. The next check is only to allow ‘View’ if the person is currently on the ‘Local Network.’ (Some sensitive activities might not be allowed if the person is outside the corporate network.) The next check confirms that the user performed a Multi-Factor Authentication (‘MFA’) with a Level of Assurance (‘LOA’) of at least ‘2’. This type of authentication refers to a stronger authentication method, like a physical token or mobile push. The policy also checks to see if the user is a member of the Doctors role and that their status is not currently set to out of office. Finally, the checks ensure that the X-Ray has neither been flagged as ‘Confidential’ nor the Subject has a relationship of the type Attending Physician.

## EXTERNAL AUTHORIZATION WITH ABAC

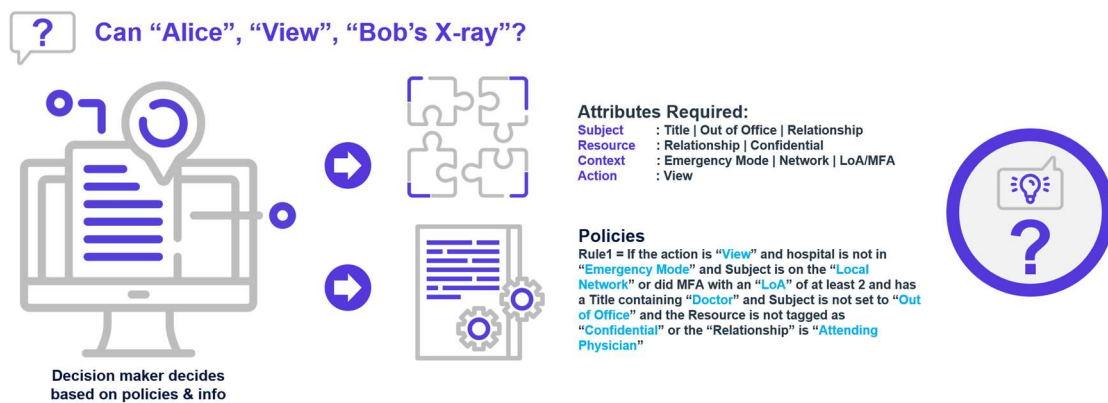


Figure 21: Elaborate ABAC policy showing the potential of ABAC



**Figure 22: Typical Presentation Slide Showing Audience Response**

It is at this point in an ABAC demo where at least one person in the audience thinks, "Wow! RBAC is dead ABAC is the best thing I've ever seen. Clearly, all systems will convert to ABAC, and we're counting the days until RBAC is dead." In fact, "Gartner predicted that 70% of all organizations would use ABAC by 2020" [3]. This is typically the high point, followed closely before the descent into the trough of disillusionment when they realize that ABAC is not a magic cure for all situations. Indeed, in the sense that 70% of organizations probably use ABAC in one way or another, Gartner's statement has a semblance of truth. However, the evidence is that RBAC remains the much more dominant means of enterprise authorization.

### **ABAC Weaknesses**

The first challenge we encounter with implementing policies like this is the information needs to be obtained and evaluated at runtime. In our policy example, if we needed to know if the user's nationality was Swiss, then this information would likely reside in either the corporate Active Directory or HR system (Figure 23). Moreover, if we needed to know if the company was in *Emergency Mode* or not, this essential information might be difficult to obtain in a live corporate environment. Likewise, if we needed to ascertain their *out-of-office* status, this would reside in a corporate email system such as Microsoft 365. Furthermore, to attain information for network login sessions or the MFA status would require a query to an Identity Provider such as Ping Federate.

## WHERE DO I FIND THIS INFORMATION “AVAILABLE ATTRIBUTES”



**Figure 23: Sources of attribute data for a complex runtime ABAC policy evaluation**

It is easy to see that the information required to make a decision at runtime will be scattered across many different systems. With regards to your organization, if your ABAC system were to make a live call to retrieve this information from these systems, there are two prime questions:

- how long would it take?
- would this delay be acceptable to your organization/its users?

For end-user applications, where an additional delay of even half a second can be considered unacceptable, such a lag is not only unacceptable but also presents an insurmountable challenge. (This also assumes that all these systems present an API that can be called to retrieve this data, which many do not.) It quickly becomes apparent that our extremely powerful but complex demo policy would be an impossibility in a real-world environment and is a puzzle that organizations need to solve.

Indeed, the challenge of where information about the resources can be found is discussed in the UMA docs from [Kantara](#). In section 1.3.2, they say the Resource server is, first, responsible for figuring out which unique resource a user's request maps to, and then, second, to pass this identifier along to the PDP. Though the PDP would have an explicit resource ID to use in its policy evaluation, it might not have the resource attributes required to evaluate a complex ruleset. UMA was designed from the perspective of users "sharing" access to their specific digital resources. [4]

With ABAC, a challenge is that the Just-In-Time (JIT) evaluations of its rules often results in a disconnect or lack of an auditable link between the rule-based policies and the resources they protect. With access rules defined in text-based policies, ABAC policies are inherently non-relational. This model contrasts significantly with the relational nature of RBAC, where users are related to roles and which, in turn, relate to permissions for resources. This lack of a relational concept is at the core of many of ABAC's weaknesses.

In terms of auditing, though answering the following standard questions should be straightforward, the reality is otherwise:



1. As an application owner, how would I see which users have access to my application and how each was granted access?
2. How will application owners be trained to read and interpret the rules and how to research all the source information?
3. Given that the attribute values' origin could come from another system or be contextual to a single login session, how would the application owner determine who matches the rule?
4. How would the application owner grant direct access to a user in an ABAC model?
5. As a delegated admin, who can add or remove users from specific groups that might be used in many ABAC rules?
6. How would a user know that adding someone to a group grants them access to Application A? With ABAC, no relationship exists between the group and the access it gives, as there would be with an RBAC model.
7. As an auditor, how would I audit how access is granted to the application and who is giving the access?
8. In a policy example with multiple unrelated checks, was the person who assigned the person's Job Title in HR the grantor of the access or was it the person who flagged the user as having completed training?
9. What security weight should be associated with the Job Title and Training Status fields on a user?
10. From how many sources could this information be edited and by whom? Potentially many people could be unintentionally performing "Entitlement Management" on a day-to-day basis.
11. How does an auditor monitor the transitions when users are granted and revoked access to sensitive applications dynamically? No log would track persons who had access on a Monday versus a Friday because access would only be evaluated at runtime.

## TOUGH QUESTIONS: SHOW ME WHO CAN VIEW Bob's Xray?

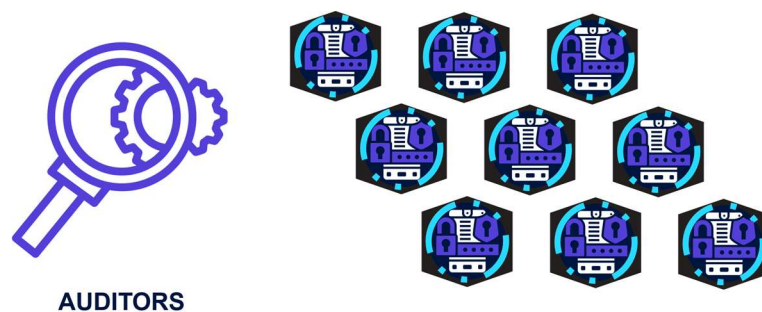


Figure 24: A Common Standard Auditor's Question

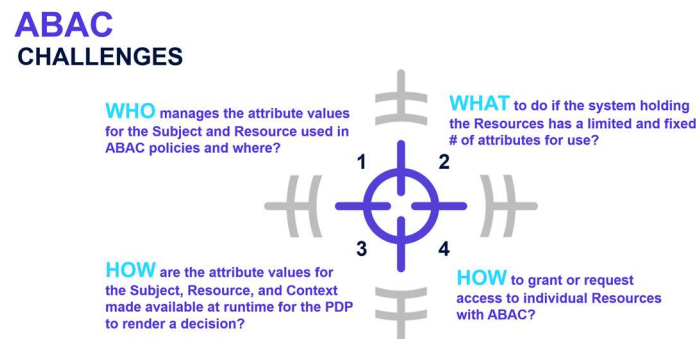
### ABAC's Primary Weaknesses

- ABAC makes it extremely difficult to perform a "before the fact audit" and determine the permissions available to a specific user. To successfully determine access, not only might a considerable number of

rules need to be executed, but they must also be done in the same order in which the system applies them. As a result, this could make it impossible to assess risk exposure for any given employee position.

- In a comparable manner to how a “Role Explosion” can occur with RBAC, an explosion can also occur with ABAC where a system with  $N$  number of attributes would have  $2N$  possible rule combinations.
- Unless rules are kept extremely simple and do not access data from various source systems, ABAC systems with complex rules from multiple attribute sources can be unacceptably slow to answer authorization queries.

## ABAC Challenges



**Figure 25: The Who, What, and How of ABAC Challenges**

The next challenge with ABAC is finding who in the organization has the skills and knowledge to write these authorization policies. To write rich, accurate, and effective policies requires both solid business knowledge and a deep understanding of security. XACML was developed 18 years ago as a standards-based language built on XML to write authorization policies. The original idea with XACML was that business users would author these policies and, in vendor marketing and some analyst circles, this idea persists to this day. The notion that business users would know best how to determine who should be able to do what from a business activity perspective is, after all, entirely logical. However, there are three reasons why the reality is otherwise: first is that most application permissions are not easily traceable to business activities. Second, in most companies, business users are so overloaded with other responsibilities that they neither have the time nor the availability to write IT policies. Third, notwithstanding the complexity of XACML itself and the time to learn it, few business users have shown interest in taking on this new skill set.

## HOW DO WE WRITE THESE POLICIES? XACML?

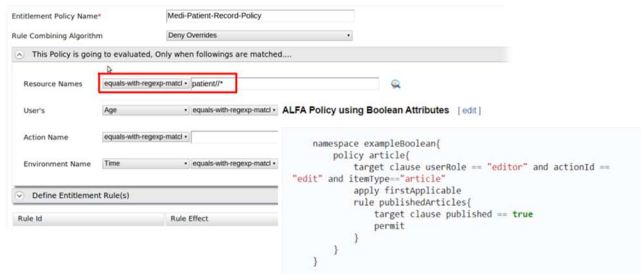


Figure 26: Examples of authoring tools and languages for XACML policies

With reference to Figure 26 above, writing XACML policies is extremely complex. Here you see that a knowledge of regular expression matching, parsing attributes, Boolean logic, queries, and order of precedence evaluation of operators is required—this is not a reasonable requirement for a business user. In the end, XACML never really took off, the magic of business user authored access control never truly materialized, and this task fell into the lap of developers bolstered by appropriate business input.

More recently, Axiomatics, an authorization vendor, developed Alfa as a friendlier XACML policy language for developers. As seen in Figure 27 below, writing XACML policies in Alfa resembles writing actual code itself. However, though these types of policy editor are the most common, and though Alfa does mask the clunky complexity of XML-based XACML policies with an overlaying code editor, like XACML, adoption has been minimal.

## HOW DO WE WRITE THESE POLICIES? XACML?

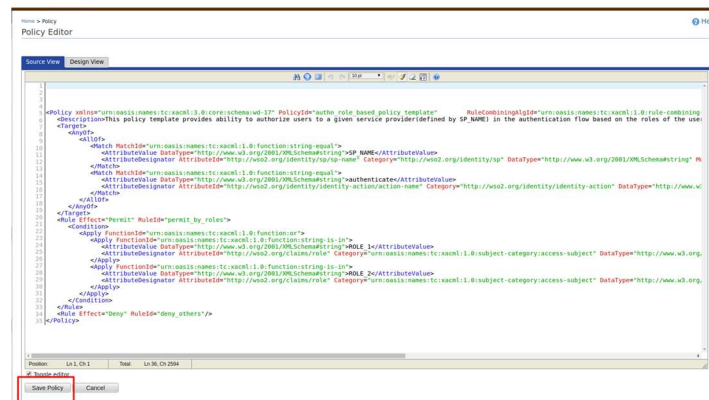
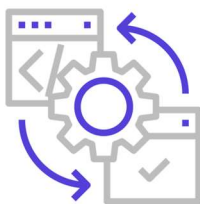


Figure 27: Example of a raw XACML policy editor

### RBAC Versus ABAC: Tradeoffs and Balance

As shown in Figure 28, the decision between ABAC and RBAC is a trade-off. On the one hand, you can have

fast, simple, and other RBAC related benefits and, on the other, you can have extensible, scalable, and other ABAC-related advantages. To date, the challenge has been to find the right balance for your own organization's needs. This is where Policy-Based Access Control (PBAC) helps.

## ABAC / RBAC BALANCE

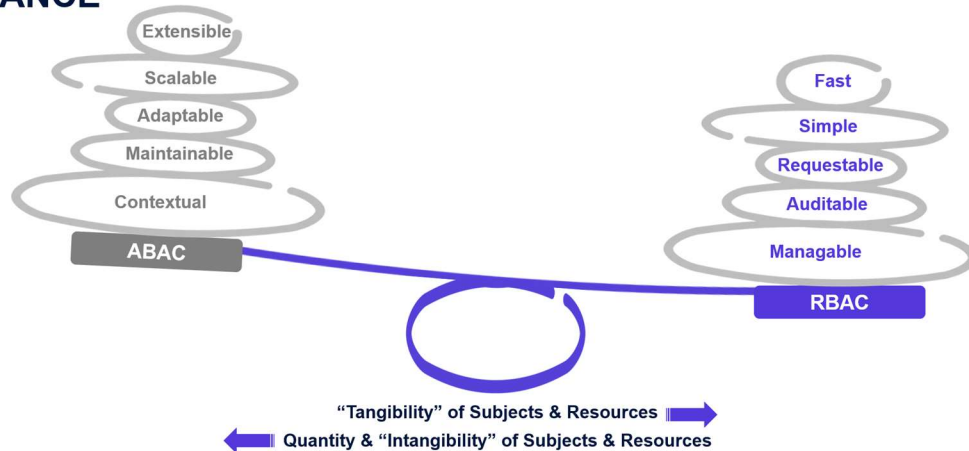


Figure 28: Balance between the benefits and limitations of attribute and role approaches

## POLICY-BASED ACCESS CONTROL (PBAC)

Policy-Based Access Control (PBAC) is not a formally defined standard but rather describes an authorization model that combines RBAC and ABAC concepts and eliminates some of their shortcomings. The closest actual standard for the PBAC model would be Next Generation Access Control (NGAC). NGAC is a relatively new standard authored by the NIST and was created to solve some of the complexities and limitations inherent in the XACML standard while maintaining its flexible and expressive nature.

*“NGAC diverges from traditional approaches to access control in defining a generic architecture that is separate from any particular policy or type of policy. NGAC is not an extension of, or adaption of, any existing access control mechanism, but instead is a redefinition of access control in terms of a fundamental and reusable set of data abstractions and functions. NGAC provides a unifying framework capable without extension of supporting not only current many access control approaches, but also novel types of policy that have been conceived but never implemented due to the lack of a suitable enforcement mechanism.” [5]*

The key concept behind PBAC and NGAC is that policies are expressed as assignment relationships that can be visualized and manipulated graphically. Access rights to perform operations against resources or objects are acquired through relationships referred to as associations. This includes the ability to define complex hierarchical relationships with inheritance, which is overly cumbersome in ABAC. One might consider PBAC as “relational ABAC.” PBAC is best used for real-time enforcement of authorization decisions where a well-

developed role model can be leveraged for policy assignment.

PBAC policies are inherently more efficient than ABAC policies because authorization decisions are not based on multiple computed and then combined local decisions. Instead, they are based on the net result of multiple policies based on relationships existing within a single database. This aspect also allows PBAC to enforce dynamic Segregation of Duties (SOD) rules, which are not entirely achievable with ABAC. A last key feature mentioned is PBAC support for “before the fact audit,” which is the ability to see who has access to a resource at any time, and not just during the real-time evaluation of a policy set.

PBAC and NGAC are similar and, for this paper and examining our model, we will use the more common term PBAC. In Figure 29 below, we have a PBAC policy that grants the Read, Edit, Print, and Delete permissions to the Doctors’ role members but with some ABAC style constraints. Our constraints or policy conditions are that the company cannot be in ‘Emergency Mode,’ the user must be accessing from the ‘internal’ network, and with strong MFA authentication (LoA/MFA  $\geq 2$ ). The main difference between this PBAC policy and ABAC is that the policy is assigned to the Doctor role. This assignment would typically be represented in a relation or graph database giving auditors a clear picture of who granted the assignment and a least a partial answer concerning which permissions the Doctor role members have themselves been granted. The assignment is visible, tangible, and can be recertified periodically. Furthermore, it could also be added to a self-service workflow process where end-users could request it.

## What is PBAC?

### WHAT IS POLICY-BASED ACCESS CONTROL? “ASSIGNABLE ABAC”

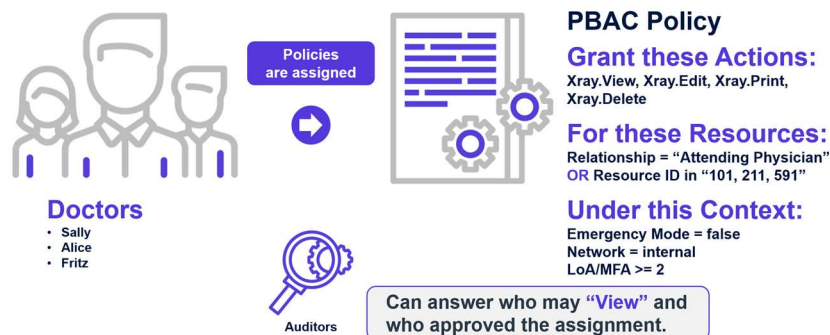


Figure 29: Information Technology – Next Generation Access Control – Functional Architecture (NGAC-FA)

In an actual real-world implementation of PBAC, the challenge of where to acquire the attribute data used in a rich policy is the same as that with ABAC. One possible solution is to split responsibility for making the final decision between the PBAC PDP and the application itself—a “Get Permissions” PBAC query. In this model, it is assumed that the application, or PEP, has access to some of the attribute data required for the policy decision. The application, or PEP, would query the PDP asking if the user were authorized to perform a particular action or operation. The PBAC PDP would then return a partial answer that would be No if there were no policy assignments to the user or their roles that authorized this action, would be No. However, if there were policies that granted the user this permission, then the PBAC response would be Yes, but with a

list of the conditions or constraints under which the action would be permitted. In this model, and assuming it has access to the data necessary to evaluate the condition constraints, it is the PEP's responsibility to enforce the rest of the decision.

Continuing with our example in Figure 30, below, Dr. Alice attempts to View Bob's X-Ray. The application/PEP asks the PDP if this should be allowed. The PDP then looks within its PBAC policy store and locates one assignment that grants Doctors this permission and one where Alice has been assigned to the Doctors role. In the PBAC assignment for Doctors granting the View permission, there are constraint conditions defined. The PDP returns a partial Yes decision to the caller, which will include conditions. The application/PEP must then lookup the needed coded or scripted attribute information and make a full decision.

## PBAC QUERY TYPE GET PERMISSIONS



Figure 30: Example of a PBAC Get Permissions Decision

Another model for evaluating PBAC decisions is what we term an "Access Check" query. This model is common where the policies are not overly complex and will rely on information easily accessible to the application or by an application gateway acting as the PEP. As shown in Figure 31 below, the application or PEP has all the necessary information required for policy evaluation and sends it to the PDP as part of the query. The application/PEP then asks the PDP if Alice can 'View' Bob's X-Ray but, because this request includes some pre-arranged information, the PDP evaluates this against its policies. Suppose the PDP finds a PBAC policy assignment applicable to Alice or one of her roles authorizing this permission. In that case, it will first evaluate the additional information to verify that conditions are met and, if they are, then return a Yes answer.

## PBAC ACCESS CHECK

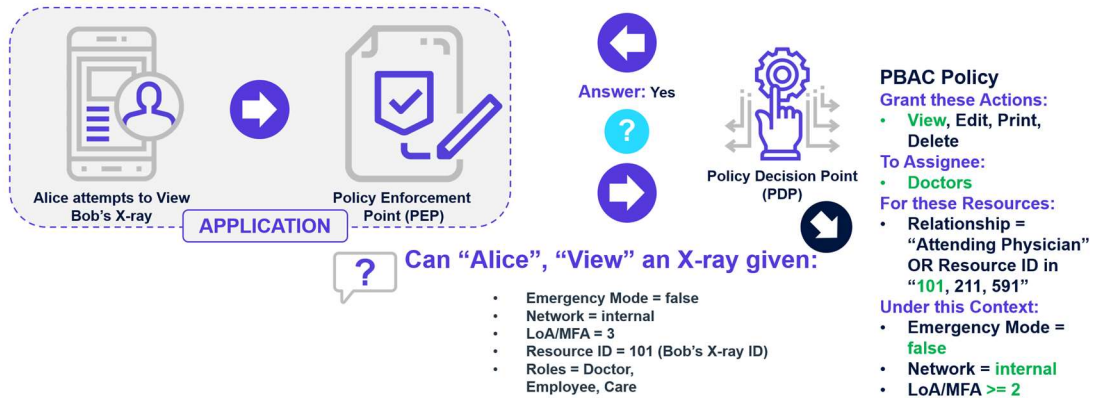


Figure 31: Example of a Full PBAC Access Check

## CHALLENGES WITH EXTERNAL AUTHORIZATION

Having evaluated multiple options for externalized real-time authorization, including RBAC, ABAC, and PBAC, we see that among their strengths and weaknesses, they fulfill their primary goal of extracting authorization decisions from within applications to a central, auditable, and visible source. As such, if an organization decided to add one of these types of PDPs to their application infrastructure, the question of "will our most important applications plug into this infrastructure and allow us to support external authorization?" remains unchanged.

## IF YOU BUILD IT, WILL THEY CALL?



Figure 32: Will our critical apps use this new external decision engine?

Unfortunately, for most of your critical applications, the answer is no. Commercial application vendors have exhibited a lack of interest in adding support for external authorization PDPs. This may be because they view

this as either an undesired external dependency that might slow down delivering features faster or ensuring compliance with various PDPs means granting external parties control over their product roadmap. Either way, a lack of support by enterprise customers for external authorization of these applications means this has not been prioritized and is why it is absent in most applications.



## HOW ARE POLICIES ENFORCED?

Most application vendors lack support for external authorization. Most organizations' problems are how to both define and manage policies centrally and then how to enforce them. Centralized authorization faces what is known as the Last Mile problem<sup>1</sup>. The Last Mile refers to the proportionally more challenging and expensive final delivery and enforcement of Compliant Access within applications and systems. To produce the most business value, compliant access policies must be “delivered” to your systems and applications and not serve merely as a reference system for recording and evaluating policies. To that end, your authorization platform must support all modes of policy enforcement or Compliant Access Delivery.

Unfortunately, because only a small number of existing applications or systems support using an external source for authorization decisions, RBAC, ABAC, and PBAC systems also encounter this problem. An example of this can be seen by examining Office 365 mailboxes.

Office 365 mailboxes maintain ACL-based permissions, controlling who may read the emails contained within a mailbox, who may send as the mailbox owner, and who may access and manage the owner’s calendar for scheduling purposes. As such, regardless of whether the authorization system relies on RBAC, ABAC, or PBAC, there is no practical way to insert an external authorization system into this equation and have the email system call out for real-time centralized decision-making for mailbox access.

Though a logical conclusion would be that the problem outlined above would be limited to legacy ACL-based permissions systems such as mailboxes and file shares, this is incorrect. While examining the much more modern and sophisticated authorization models employed within Amazon AWS, we see this relies on extraordinarily granular and flexible ABAC-like policies stored as JSON documents. Here, security admins can

---

<sup>1</sup> The Last Mile problem is a colloquial phrase originating in the telecommunications industry to describe how the final delivery of their service to the end consumer proves proportionally more challenging and expensive.



precisely define which operations or actions can be performed for which AWS services and the specific resources within these services. The policies include the ABAC model's flexibility and contain some of its inherent lack of auditability. Again, as with our mailbox example, the AWS authorization system does not “call out” for decision-making from a centralized PDP.

Given these inherent limitations in much of the IT application landscape, how do authorization vendors maintain relevancy for the permissions they define and manage centrally? In many cases, the answer is there is nothing they can do to enable these systems to perform external real-time authorization. There are, however, a few commonly employed methods to allow centrally managed authorization policies to be enforced in these types of systems.

## Self-Enforcement

The optimal scenario would be “self-enforcement” or native enforcement. Self-enforcement is where application development teams embrace the idea of external authorization for new custom applications. With this method, the mindset changes to this new way of writing application authorization logic, toolkits are created, and new application templates include the necessary components for interacting with a PDP. External authorization provides complete auditability for applications and centralized management. We know that this is typically only possible for custom applications where the organization is in control and can implement the necessary plug-ins to call a PDP for decisions.

### SELF-ENFORCEMENT EXTERNAL AUTHZ-ENABLED APPS



**Figure 33: Self-Enforcement – External Authz-Enabled Apps**

One standard supported for applications to leverage a PDP for external enforcement is User-Managed Access 2.0 (UMA – as shown in Figure 34). UMA provides a standard protocol for registering their protected resources and performing access checks by calling an external PDP. An extension to OAuth, UMA is a flexible standard that makes it familiar for application developers and an excellent fit for many scenarios where privacy and consent are vital concerns, including IoT and consumer data sharing.

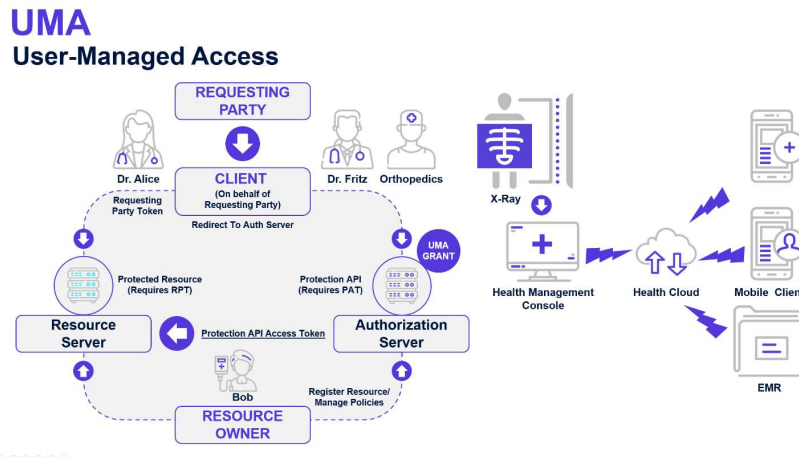


Figure 34: Example of User-Managed Access (UMA)

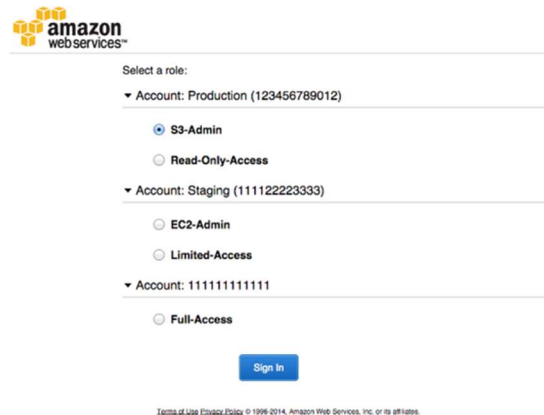
### Claims-Based Enforcement

Many modern web applications support centralized authentication. This is where the user signs into a trusted identity provider, which then delivers the user's identity and requisite information to the application as "claims." The application then trusts this information and uses it to verify the user's identity, leveraging the claim information to control access within the application itself.

A sophisticated example of this can be seen in the AWS authorization model (an example is in Figure 35, below). AWS supports receiving AWS IAM "Roles" as claims. AWS receives the identity provider's role claims and then asks the user which role they would like to assume for the current login session. These roles are, in turn, mapped within AWS to the authorization policies we discussed above. This mapping allows a centralized system tied into the federated authentication process to "control" the authorizations within AWS. Again, it is important to note that AWS does not call out to the authorization system for real-time decisions, and the control is not granular in that the central system. This model offers the benefit of centralized management of AWS roles, the ability to request AWS roles and recertify AWS role assignments and access. However, what is lost through this method is the actual meaning of the AWS roles and visibility into the permissions they grant within the AWS system.

# CLAIMS-BASED ENFORCEMENT

**Federated applications trust a central identity and authorization source to deliver Compliant Access as claims during login.**



**Figure 35: Example of Claims-Based Enforcement for AWS**

Because the claims-based enforcement model is completely disconnected and does not require any modifications to the vendor's application, it provides a good compromise when it is impossible to call out to an external PDP. The claims-based authorization model can be supported by either an RBAC, ABAC, or PBAC system.

## Gateway Enforcement

The Gateway Enforcement model prevents end-users from directly accessing applications or systems. Instead, they must connect through a gateway that authenticates, authorizes, and monitors their activities. For web-based applications and APIs this is known as Web Access Management (WAM), or the Reverse Proxy Approach. In this model, web applications and APIs are protected behind an Application or API Access Gateway. No direct connections are allowed, and the Access Gateway enforces the organization's authentication and Compliant Access policies. This model is the most common for on-premises or legacy applications and new microservices.

A Gateway Enforcement model is often the best approach to enforcing external authorization for legacy web applications where there is no other option to modify the applications themselves. It is also a recommended approach for microservices. Maintaining authorization enforcement logic in the gateway rather than within the microservices reduces the complexity of the microservice code and supports a single purpose design goal.

## GATEWAY ENFORCEMENT

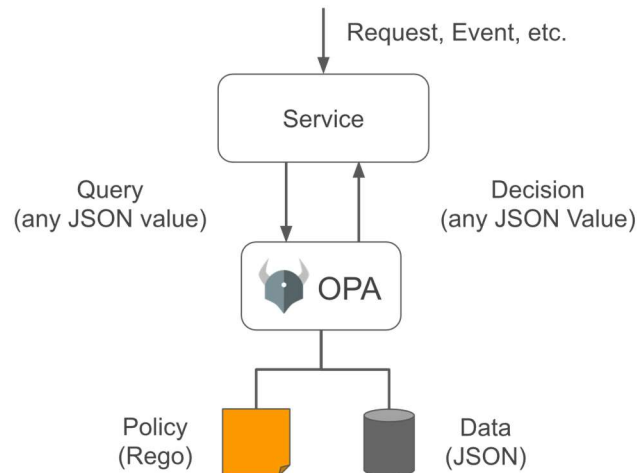


**Figure 36: Example of Gateway Enforcement**

To illustrate this concept, imagine our fictitious organization is writing microservices and wants to implement policies and authorization to control who can access the xrays API endpoint. In this example, though there are looser restrictions on who may retrieve records via an HTTP GET, they are much tighter on who may issue an HTTP DELETE to avoid unauthorized deletion of records. The best practice for developing microservices is that they are designed for a single purpose, and general code, such as authorization, is not duplicated. The desired architecture would be to place all microservices behind an application or API gateway that acts like a reverse proxy controlling access to the APIs and pages it exposes. The API gateway would then support authorizing all requests by calling out to a PDP. In our previous example, Dr. Alice attempts to view Bob's X-Ray via an HTTP GET request to the `/xrays` API endpoint. Suppose the policy was not overly granular and did not require additional information. In that case, this information plus any contextual information, such as IP address, client operating system, and browser, etc., could be sent along with the request to allow a complete decision. A commonly used method for enforcement of access policies, Gateway Enforcement, is a trusted model.

### Open Policy Agent (OPA)

Relatively new to the distributed microservices world, Open Policy Agent (OPA) is a graduated standard backed by the Cloud Native Computing Foundation (CNCF). As shown in Figure 37, OPA is designed to serve as a generic policy-based decisioning engine. Commonly used for cloud infrastructure policies to control critical infrastructure, such as Kubernetes, OPA can also act as a distributed PDP for microservices authorization. OPA can run as a lightweight Docker container that easily plugs into a microservices architecture. Its lightweight nature means it can run as many unique instances as required. The ability to run side by side and scale-out along with your microservices offers enormous flexibility, including isolating and distributing loads and providing faster decision times than calling back to a central PDP.

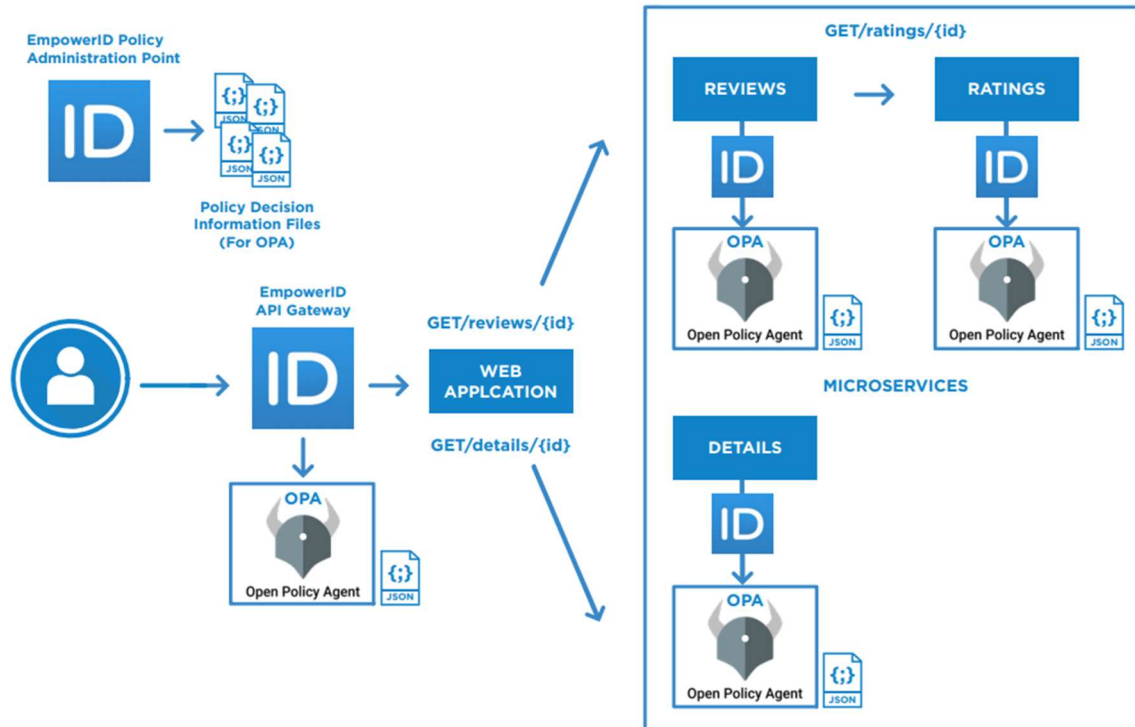


**Figure 37: Example of OPA Enforcement Model**

Though OPA acts as a distributed PDP, it does not act as a Policy Enforcement Point (PEP). This latter role is typically performed by some type of application gateway or ingress controller function that calls out to OPA for decisions and enforces that result. OPA requires three components to make decisions: policies, query, and data files.

- Policies – are written in REGO, a custom policy language.
- Query – this input is comprised of the “question” plus associated data passed in by the caller or PEP. This query typically includes information about the caller's identity, the URL they are attempting to access, and some contextual information.
- Data files – OPA uses these to refer to, or obtain facts about, the world, such as definitions of roles and their members, the resource definitions, and other supporting information be required to evaluate the policies.

The OPA engine takes these components and then returns either a Yes or a No decision to the PEP caller (refer to Figure 38).



**Figure 38: Example of Cloud Native OPA Authorization**

OPA shines in scalability and performance but is still restricted by the same limitation as ABAC and PBAC. This limitation is that the authored policies' sophistication is directly limited by the availability of the data required to evaluate their rules. In the OPA model, the data used in the policy evaluation exists as one or more JSON files that are local to the engine itself. Because the files must exist locally, OPA faces the same issues as ABAC concerning the auditability of these distributed REGO policy files and access to data required for decisions.

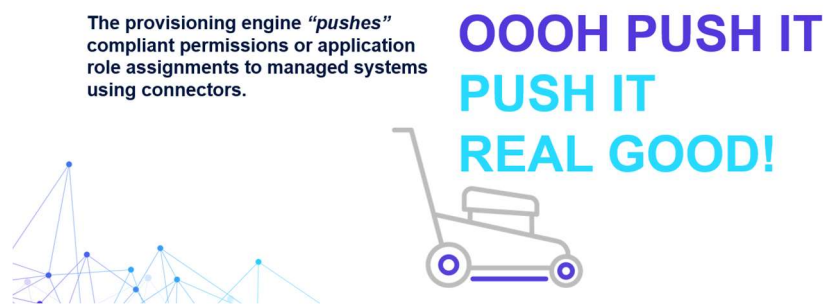
EmpowerID has embraced OPA and extended the EmpowerID PBAC engine to add additional centralized manageability and auditability to the distributed OPA model. PBAC policies define who may access which microservices and under what conditions can be authored, audited, certified, assigned, and requested within the EmpowerID user interfaces and by REST API. As shown in Figure 38 above, these PBAC policy assignments are then available via the REST API. They can be pulled down periodically by the distributed OPA instances to produce a locally available data file. In conjunction with the PBAC policies and role memberships that reside as local JSON data files, simplified REGO policies—that are the same across all OPA instances—then base their access decisions on the request information. This hybrid OPA model adds the best of centralized control and visibility without losing the benefits of OPA's disconnected and distributed model.

### Provisioning Engine Enforcement

The last option available for granting and enforcing Compliant Access relies on the EmpowerID provisioning engine to provision permissions or application roles to managed systems using connectors. An excellent example of this is the SAP Advanced Business Application Programming (ABAP) systems, which rely on

complex roles and profiles to grant user access. Roles and profiles in SAP authorize users to execute transactions known as Transaction Codes (TCodes) within the constraints defined by the assignment of fields and their values. In this case, a gateway or claims approach is not possible. Instead, though policies defining Compliant Access and business risk are modeled in EmpowerID, any resultant access must be manually granted and revoked by the EmpowerID provisioning engine. The Provisioning Enforcement model is the most common model and is available for many systems with out of the box connectors. This model works for any system supporting the System for Cross-domain Identity Management (SCIM) standard and can be extended to other systems through the creation of a custom connector.

## PROVISIONING ENGINE ENFORCEMENT

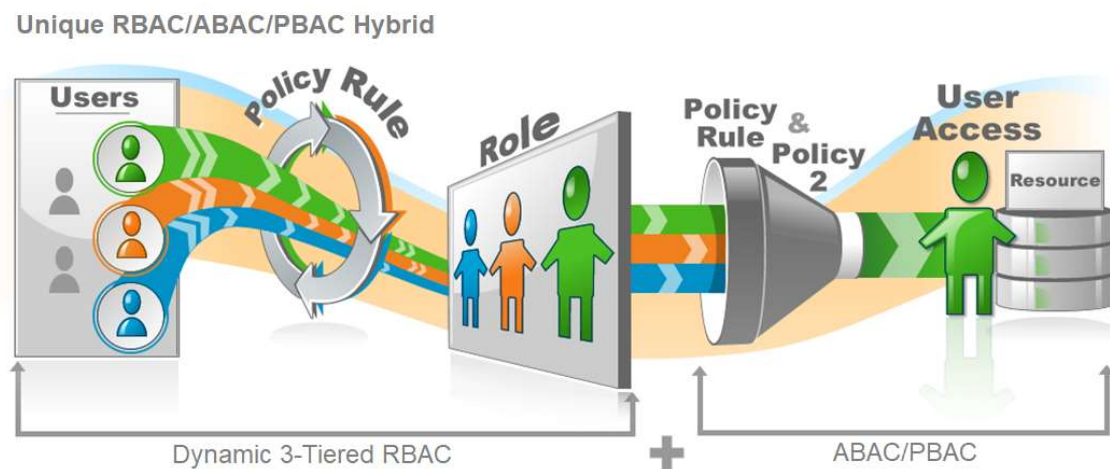


**Figure 39: Example of Provisioning Engine Enforcement**

A final option for controlling application permissions based on centrally managed policies is to “push” or “enforce” them to the system using a scheduled provisioning engine. However, because contextual on the fly decisions are not possible here, this involves centrally managing access policies using RBAC and then pushing down the resultant “calculated” access into the external systems. Using our earlier Microsoft 365 mailbox example, this would involve an RBAC-based system calculating all the role assignments or policies granting mailbox permissions for each mailbox and then triggering a provisioning system to grant or revoke the permissions via PowerShell with Microsoft 365. Though those policies use inheritance and query-based rules, this model relies on a unique type of RBAC within a Big Data-like engine that can continuously calculate who has which permissions to what external system resources.

## EMPOWERID'S HYBRID ROLE AND ATTRIBUTE COMPLIANT ACCESS APPROACH

Defining and maintaining compliant access for a large organization can be a daunting task. Some types of applications and use cases are better suited to a more structured role-based approach, whereas others require real-time contextual decisions. RBAC, ABAC, and PBAC are three ways of managing authorization policies. Moreover, while both have overlapping qualities, individually, each one cannot cover all the necessary aspects of access control. However, for optimal, dynamic support of an IT organization's needs, EmpowerID supports RBAC relational modeling. RBAC relational modeling provides the backbone or structure for defining an organization and its overall policies while leveraging the flexibility and real-time contextual nature of ABAC and PBAC to offer the best comprehensive solution.



**Figure 40: EmpowerID's Innovative Hybrid RBAC/ABAC/PBAC Model**

EmpowerID's sophisticated role and relationship modeling allow security architects to model the organization and its structure and policies, including segregation of duties policies to prevent undesired combinations of access. As illustrated in Figure 40 above, flexible attribute-based ABAC or PBAC policies support the centralized real-time decision point for applications that can call the EmpowerID API for authorization decisions. The ABAC/PBAC engine enhances or modifies the powerful RBAC engine's decisions, allowing their use only when greater flexibility or contextual information such as risk, location, and MFA type is required. By including the pre-calculated access results that the engine derives from complex RBAC policies that account for inheritance and even attribute-based queries, ABAC/PBAC policies are made much more potent. The end-goal of leveraging each approach's best is to deliver what EmpowerID calls "Compliant Access Delivery."

### What is Compliant Access Delivery

"Compliant Access Delivery" is, first, the capability to map out in advance the position appropriate access for employees, partners, and customers and the risk policies that will measure and ensure continued compliance and to then monitor and enforce this compliant state through automation. The natural



follow-on question to this is “What makes access compliant?” Compliant access is **appropriate** to the person to whom it is assigned in accordance with the organizational standards and **business policies** to minimize risk. Compliant Access Delivery synthesizes multiple Identity and Access Management (IAM) technologies with a business modeling approach to automate and maintain each user's appropriate access to IT systems while continuously minimizing risk.

---

*“Compliant Access is a well-defined target state against which the current state can be measured.”*

---

The keys to Compliant Access are the words “appropriate” and compliant with “business policies.” With Compliant Access, the devil is in the detail and is where traditional solutions miss the mark. With its distinct language, processes, and policies, Compliant Access cannot be delivered by an IAM system that does not bridge the divide between the technical *systems* world and the business. With its operating procedures, industry norms, and regulatory restrictions, the business defines what job appropriate is, what is a risk, and what is non-compliant. It bears repeating that, in this model, “Compliant Access is a well-defined target state against which the current state can be measured.” To achieve this, EmpowerID leverages both your organization's business model and its language to enforce and to provision Compliant Access across Cloud and on-premises systems.

### **How Does Compliant Access Leverage Your Business Model?**

Defining business-appropriate access is a challenge for IT organizations using their existing IAM systems. These systems were designed to manage only the technical aspects of access control and universally lack a conceptual “bridge” to tie the technical entitlements to the business's operating model and the activities or “Functions” performed by its participants.

### **Business Processes and Functions**

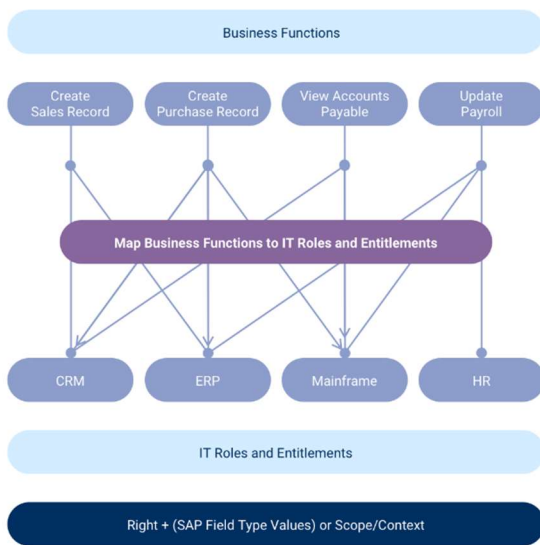
All businesses can be broken down into a series of business processes performed during the production and delivery of their goods or services. These business processes comprise a series of tasks that internal and external participants perform to complete that process. Each task can be further broken down into a series of functions that need to be executed to complete that task. An example of a function would be creating a purchase order that is part of a more extensive purchasing business process. A purchase order exists within the business realm, is part of the everyday business language, is easily recognizable, and the actual function of creating one is straightforward. However, not everyone in the business needs or should be allowed to create a purchase order. As such, when periodically reviewing employee access, managers and auditors would easily spot those who can generate purchase orders and those who cannot. This simplicity contrast starkly with the often-unintelligible names and descriptions of technical entitlements that grant access to systems and applications. Functions are the missing link that binds together traditional technical access management with the world of business-friendly terms and recognizable job-appropriate access. Functions afford us a clean and distinct separation between business and IT.

## Risk Policies

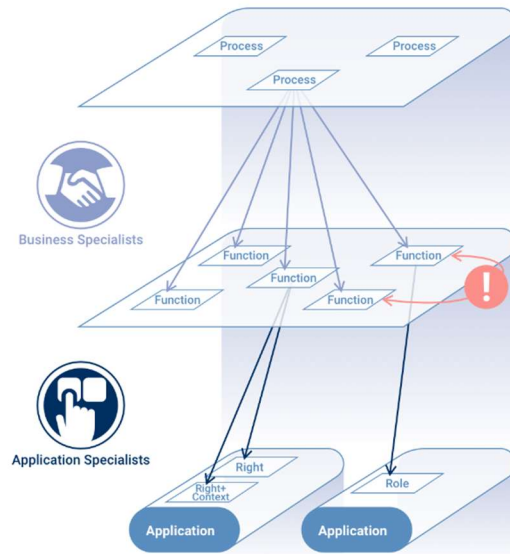
As outlined, Compliant Access must be “appropriate”, and it must adhere to an organization’s “business policies.” These business policies are specific to the organization itself and are tailored to match how the organization operates, its industry, the processes executed to deliver goods or services, and the regulatory requirements it must adhere. It is important to note that business processes are non-technical and are independent of an organization’s IT systems. For example, an organization’s customer acquisition process and related risks do not vary based on what Customer Relationship Manager (CRM) they use, whether Salesforce.com, HubSpot, etc. Regardless, the appropriate end to end functions that a user will perform in their position can be known and mapped out in advance of ever connecting to any IT systems. Moreover, by defining which of these functions are higher risk or identifying combinations of functions that, if held by the same person, could lead to fraud or misdeeds, organizations can offset such threats. Defining risk policies at the function level is crucial because these are the actual business activities performed and which an organization must monitor and control. For business users, confusion often arises when risk policies are based on technical entitlements that lack visibility into what the user can actually do rather than directly related to business concepts they know and are familiar.

## Mapping the Business World to the Technical

If functions are the proper level to define business-appropriate access and risk policies, then a method of bridging, or mapping, them to your IT systems is a must. This is where your business specialists are crucial. These specialists map out your organization’s business processes, activities involved, and which roles are participants in these processes (refer to Figure 41). Subsequently, together with your IT application specialists, they work to create the needed bundles of functions as task-based application roles in the specific systems used to perform these tasks (as shown in Figure 42). EmpowerID supports the less granular model of mapping application roles to the functions they deliver or, for some systems, the ability to go a level deeper and map the fine-grained application permissions to functions. In the fine-grained model, EmpowerID can discover which functions are granted by application roles even if not explicitly identified in a mapping. In either case, the result is a system that can determine which IT entitlements grant which functions, which users currently can perform which functions, and in what manner they were assigned that access. Functions bring business-friendly clarity to your heterogeneous IT landscape.



**Figure 41: Business Function Mapping to IT Roles and Rights in external Systems**



**Figure 42: Conceptual Model of Function Mapping to Technical Access**

The benefits of business function mapping include:

- Making roles and access intelligible by adding a business explanation layer.
- Increasing end-user satisfaction by empowering users to make informed decisions and find needed access.
- Combatting risky privilege creep by eliminating “rubber stamp” detective controls.

### Role-Based Access Control

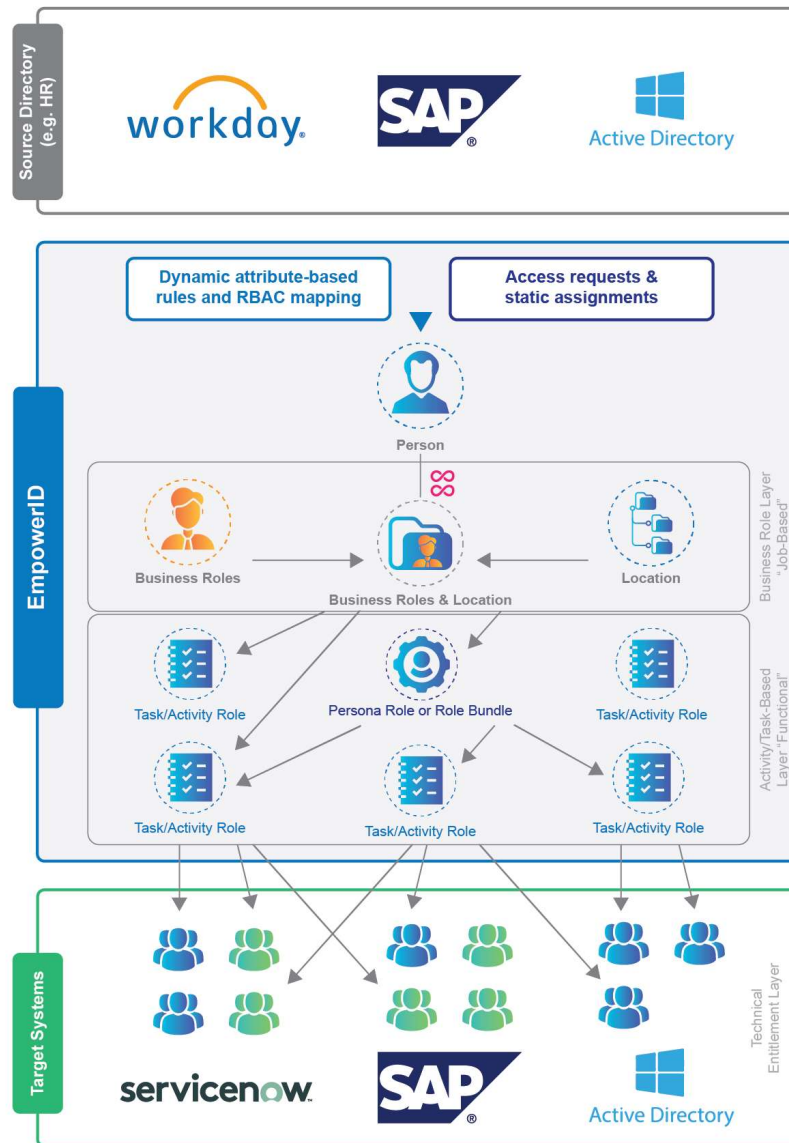
Defining position-appropriate access for a large organization can be challenging, and maintaining it even more so. However, without this guideline, IT organizations are forced to resort to costly and inefficient manual processes that make achieving Compliant Access far harder. In contrast, Role-Based Access Control (RBAC) makes defining and efficiently maintaining position-appropriate access easy. Roles are bundles of access that can be assigned to users and linked to your organization’s policies. By defining the appropriate access for each type of employee or supplier that must be provisioned across an organization’s on-premises and Cloud systems, roles also optimize Compliant Access delivery.

Make no mistake, the flexibility and power of an IAM solution’s RBAC model can make or break many projects. If poorly designed, then no amount of consulting or engineering will lead to a manageable RBAC model, and the RBAC system itself will become both source and scapegoat of project failure. Often cited as the most significant single contributor to customer projects' success, EmpowerID’s 3-tier RBAC engine is unique and is the most sophisticated and the most powerful.

Comprised of a Business role tier, a Functional role tier, and a Technical role tier, a best practice implementation of this 3-tier model is shown in Figure 43 below. Here, attributes from authoritative

upstream sources like HR are used to automate the assignment and revocation at the Business Role tier using the EmpowerID “RBAC Mapping” engine. Business roles are used to bundle up easily recognizable activity, task, or persona-based functional “Management Roles.” The functional layer allows business roles to align with the organization and its processes more closely. It does this without being tied directly to the systems' technical roles where the access is used, thereby adding significant flexibility to the model. There are three significant advantages to this:

- There is no impact on the general business role strategy when the technical systems are changed.
- Functional roles are more easily understood by business users, greatly simplifying the access request and recertification processes.
- Technical roles are the bridge to granting actual access in the target systems, most commonly through the assignment to application roles or groups.



**Figure 43: Illustration of the layers of an optimized best practice EmpowerID role implementation**

The EmpowerID hybrid RBAC/ABAC/PBAC model was designed to solve the following common RBAC challenges:

- Roles and their entitlements are cryptic and unintelligible to business users.
- Too many roles lead to an eventual “role explosion.”
- Roles are not tied to an authoritative source like SAP HR.
- Lack of automation for role assignments/revocation.
- Minimal enforcement of access expirations and renewals.
- Lack of a centralized policy model and authorization service across all systems.

- Not flexible or adaptable enough for modern technologies, such as IoT.

## Business Roles

As discussed previously, RBAC's fundamental weakness is the risk of "role explosion." This is where organizations end up with vast numbers of roles to accommodate people performing the same job function within an organization but in different geographical or business areas.

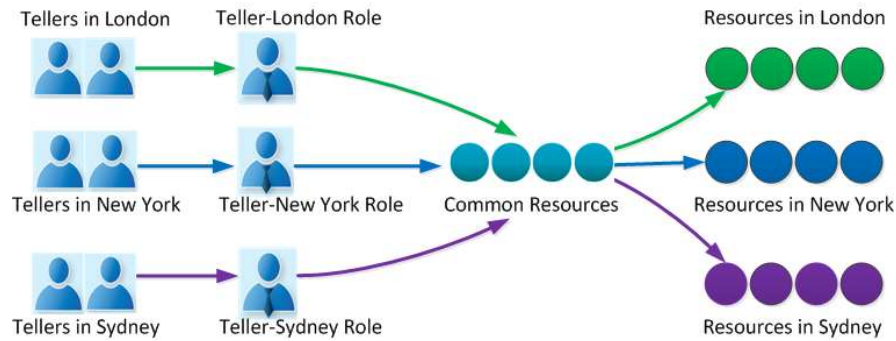
EmpowerID solves this role explosion challenge through its unique two-trees or "polyarchical" RBAC approach. The top, Business Role tier describes a user's position in the organization. This is then used in combination with a hierarchical Organizational Location representing where in the organization or in which context the user performs their Business Role. As shown in Figure 44, below, this position is visualized as two trees with people assigned to one or more Business Roles combined with an Organizational Location. A person's Business Roles bundle up direct technical entitlements and, more commonly, Task or Activity-Based roles.

The screenshot displays the EmpowerID interface for managing Business Roles and Locations. It features two search trees at the top: 'Business Role' and 'Location'. Below these is a 'Members' section with tabs for 'RBAC' and 'Policies'. A search bar and 'Search' button are present. A table lists members with columns for First Name, Last Name, Login, and Inherited from Business Role and Location.

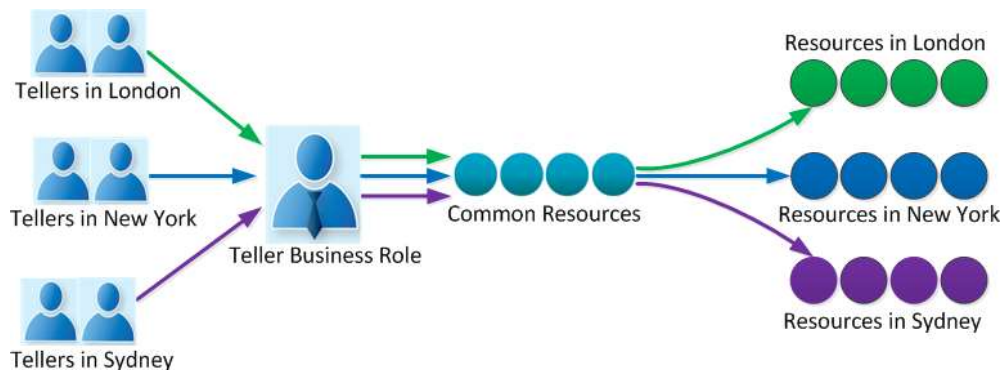
	First Name	Last Name	Login	Inherited from Business Role and Location
	Otto	Hahn	100000000015	Team Leader in Warehouse Management
	Otto	Stern	100000000016	Team Leader in IT

**Figure 44: EmpowerID's polyarchical RBAC approach showing people and their Business Roles and Locations**

As we showed in Figure 19, and added below for clarity, this is a current common method where multiple roles are required and directly impacts the quantity or roles required and 'role explosion'.



In contrast, as shown in Figure 45, below, EmpowerID's Business Role and Organizational Location RBAC model easily solves this challenge. Because EmpowerID allows you to differentiate access assigned to a Business Role based on the entitlements' Organizational Location, we now only need one Teller Business Role. In this manner, tellers in London, New York, and Sydney can all have the same Business Role without accessing each other's entitlements. This helps minimize the number of roles required, reduces administrative efforts, and always ensures Compliant Access.



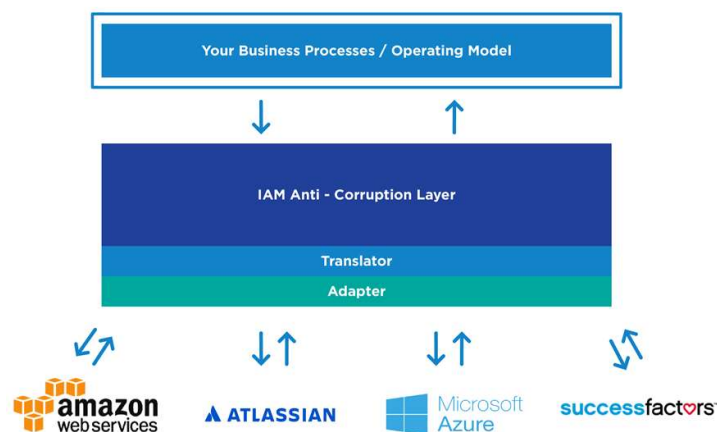
**Figure 45: Role Explosion Solution – one Teller Business Role scoped on assignment by Organizational Location**

There are significant benefits where business roles and locations are concerned, including:

- A familiar and commonly accepted grouping mechanism that non-technical users of the system can recognize and easily navigate. The structure can be mapped to the organizational structure of the business.
- Anchor points for mapping external roles and locations from connected systems. This way, master person identities can easily be provisioned into a business structure.
- Leveraging powerful and complex inheritance relationships allows you to anchor common access and policy assignments at varying inheritance levels. This inheritance eliminates the need to create unnecessary duplicate assignments.
- A simple structure for linking multiple and varied assignments to a common anchor point which allows the administrator to accumulate widely varying types of assignments and policies to an easily recognizable business structure.

## Management Roles

Business Roles typically represent job positions within an organization and are used to bundle and report appropriate Compliant Access. However, modern organizations are composed of cross-functional teams working on initiatives or projects, and not all access is either job-based or necessarily assigned directly to each Business Role. Instead, in EmpowerID, these are commonly bundled into manageable Task-Based RBAC or T-RBAC “activity-based” functional roles known as “Management Roles.” These Management Roles can be designed to grant the bundles of technical roles, entitlements, and permissions in external systems required to complete everyday job duties or tasks, such as “New Customer Onboarding.” It is quite possible that, in an organization, multiple Business Roles might perform this task, and, therefore, granting the task as a bundle makes access far more manageable and auditable. Moreover, it is this middle layer that bridges the gap between the organization’s job-based business roles and their cryptic, external system technical entitlements and permissions and enables the user to perform their business activities. As shown in Figure 46, below, these IAM activity-based roles then act as an ‘Anti-Corruption Layer’ (to borrow a Microservices term) by ensuring that the business activities performed by various job roles remain unaffected by any changes to the IT landscape, in turn protecting the business processes and operating model.



**Figure 46: IAM Acting as an Anti-Corruption Layer Insulating the Business Model from Technical Changes and Limitations**

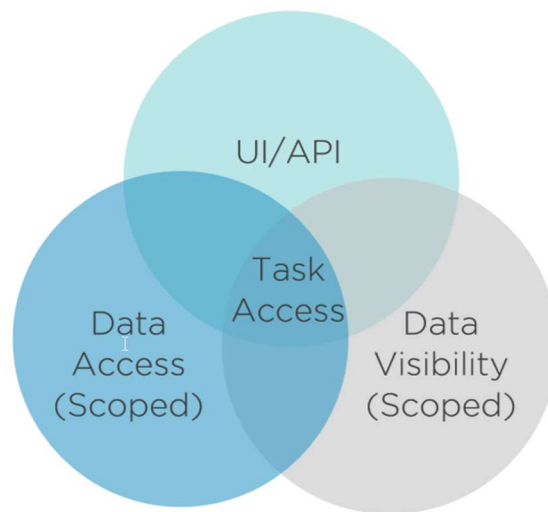
To further expand upon this concept, if the organization were to define the activities performed by its sales staff and directly map the required technical roles and permissions to these business roles, any change in the system used to perform these activities would require a redesign of multiple Business Roles. Now though, with Management Roles acting as our anti-corruption layer, the Business Roles would remain the same even if sales tasks were now spread across numerous SaaS applications. In this instance, the only changes required would be mapping the clearly defined activity-based Management Roles to the new system’s corresponding technical roles and permissions. One further added advantage is that the technical system owners could handle this mapping without requiring the business team's business role redesign and involvement.

In addition to providing activity-based roles, Management Roles are also used for teams, cross-department collaborations, or project teams. This tier is looser and more flexible than the Business Role tier as it is less



tied to a person’s job description.

T-RBAC is used internally by EmpowerID to organize who may use which user interfaces, APIs, and workflows, who may see which objects and data, and who may perform which actions against objects. T-RBAC separates access for UI, data visibility, and data access to avoid bundling and over-permissioning. As shown in Figure 47, these are broken down into three primary types to segregate the access they grant: “UI/API” roles, Data Visibility (“VIS-”) roles, and Data Access roles (“ACT-“). Segregating these activity-based or task-based roles in this manner allows them to be easily reused and “composable” into any number of combinations without requiring the creation and maintenance of new roles.



**Figure 47: Venn diagram of the 3 types of T-RBAC Management Roles and how they combine to enable task-based access**

To further elaborate, UI/API-Management Roles grant access to user interface elements, such as pages and controls, and access to run workflows. VIS-Management Roles grant the visibility access to view specific types of objects or resources in a particular scope, such as at the organizational level or those in or below an organizational location. VIS-Management Roles also control access to the API endpoints that allow users or applications to make the calls to retrieve the data for that object type. And lastly, ACT-Management Roles grant the authorization to perform specific actions or “operations” within EmpowerID user interfaces and workflows against scoped data, such as individual mailboxes or all objects of a particular type by organizational location.

### Technical Roles

The bottom tier in the EmpowerID RBAC model is comprised of Technical Roles known as Access Levels, which are proper roles granting permissions from an RBAC standard perspective. Access Levels are the system or application-specific roles used to connect the policies in EmpowerID to the actual permissions those policies grant to resources contained within external systems or applications. An advanced example of an Access Level would be the Mailbox Publishing Editor Access Level, which would grant permissions to a mailbox delivered as ACLs within Office 365. Access Levels can grant these “Rights” within external systems and “push” them out via the provisioning engine. The most common Access Level is “Member,” which is

used to give a person or an EmpowerID role membership in external systems groups or application roles.

Access Levels also define Compliant Access within EmpowerID as bundles of low-level permissions known as operations. As mentioned above, the user actions in EmpowerID's web interfaces or APIs undergo real-time access checks to determine if they may perform the intended operation against the resource in the given context. These actions can range from requesting membership of an SAP Role in the IT Shop to assigning user accounts to Azure RBAC roles in a Microsoft Azure tenant. These same low-level checks govern the EmpowerID RBAC model's management, with RBAC management activities represented as operations.

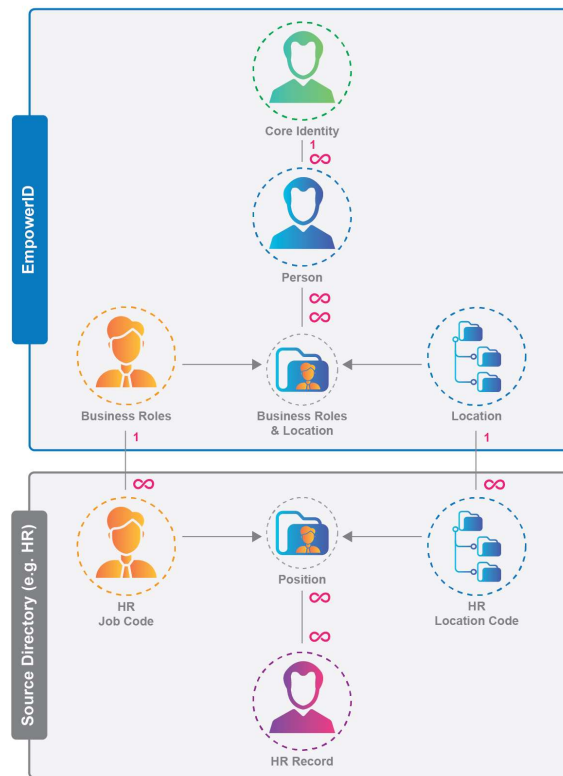
## **Role Mining and Optimization**

If defining and maintaining a role structure is critical to Compliant Access Delivery, how do organizations tackle the overwhelming challenge of getting started and generating their initial roles? Without the right tools and techniques, role-based security projects can become delayed as organizations struggle to define this initial set of roles.

EmpowerID's approach leverages data from your existing systems to create an initial structure. Sophisticated machine learning algorithms then uncover the existing "implicit" roles already in use within an organization by analyzing current access assignments for users. These existing roles become the starting point for defining standardized access roles and adopting RBAC. The first step in discovering the initial structure is to inventory the data from all authoritative source systems.

## **Sources of Business Roles and Organizational Locations**

Establishing Business Roles and Organizational Locations is usually the starting point for many EmpowerID projects. This data's best sources are usually an organization's HR or Human Capital Management system (HCM) and Active Directory (AD). HR systems such as Workday, SuccessFactors, or SAP HCM maintain a rough organizational structure and all employees' positions. Even when only available as user attribute data, these systems provide an invaluable source for the initial Business Roles and Organizational Locations to start the preliminary analysis. EmpowerID's out of the box connector model supports inventorying this data into what we term "External Roles" and "External Locations." At any point in time, EmpowerID has a copy of an organization's external roles and locations and which users are assigned to each. AD is often a rich source for the organizational locations as some portions of the Active Directory Organizational Unit (AD OU) tree represent a hierarchical view of the company (refer to Figure 48).



**Figure 48: Logical illustration of the RBAC mapping concept**

Once this external data resides within the EmpowerID system, it can generate the initial Business Role and Organization Location trees. As shown in Figure 49, below, EmpowerID’s role and location mapping technology, “RBAC Mapping,” allows the internal trees to be designed independently of the actual external structures. The advantages here are that many structures are often more rigid and not optimally designed for managing access. With RBAC Mapping, a one-to-one match between external roles and locations and internal is neither required nor desired.

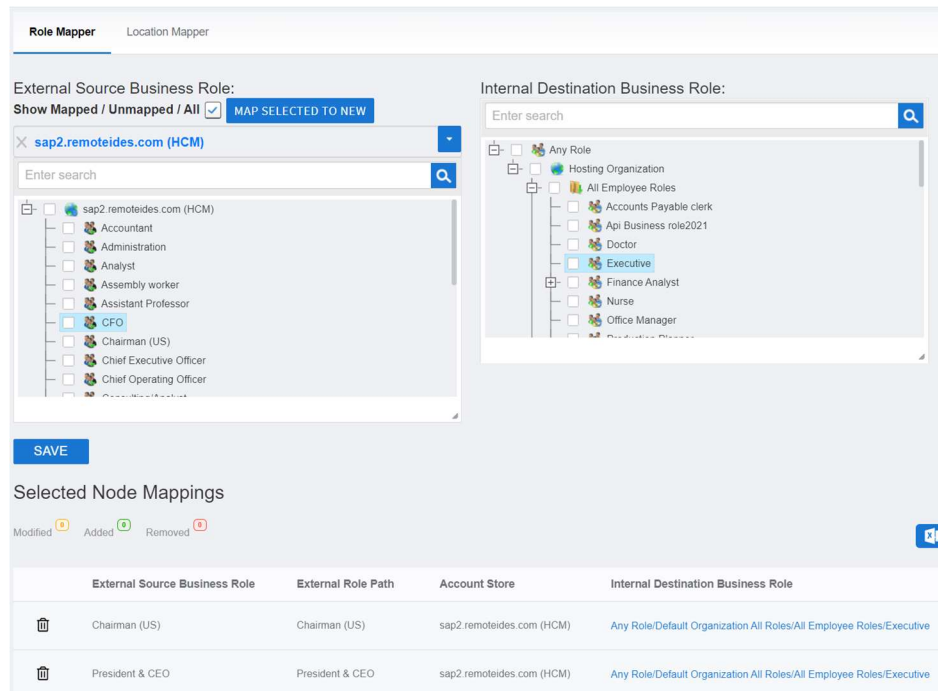


Figure 49: RBAC Mapper showing external role mappings

## Dynamic Attribute-Based Maintenance of User to Role Assignments

Once an organization has generated its roles, maintaining accurate membership can be alleviated by employing a Dynamic-Roles model—a hybrid of RBAC and ABAC technologies. Dynamic roles are assigned to users by attribute-based policies. An example of one of these policies is the RBAC Mapping technology mentioned above (refer to Figure 49). RBAC Mapping allows “sensitive attributes” in external authoritative source systems to continuously automate the Business Role and Location assignment, ensuring that users are assigned to the correct, compliant roles in EmpowerID. These changes are typically the key driver for the Joiner, Mover, and Leaver (JML) process and ensure Day Zero access for employees in their new positions and Day Zero termination when they leave. Automation based on authoritative systems changes is a key driver in continuous Compliant Access Delivery to eliminate laborious and expensive manual administration.

## Dynamic Hierarchies

EmpowerID’s Dynamic Hierarchies engine is an automated powerful policy engine for creating attribute-driven roles. The idea behind Dynamic Hierarchies is simple: organizations require self-maintaining roles based on attribute combinations such as location, company, division, department, and title. Any attribute within the EmpowerID Identity and Entitlement Warehouse can be used in a policy. Dynamic Hierarchy policies define the attribute rules to generate or create the roles based on the distinct combinations of attributes found in an organization’s data and between the matching members. The Dynamic Hierarchies engine automatically handles these dynamic roles’ lifecycle, with new roles being created as new combinations of attributes appear and older roles being retired when they no longer have members. Dynamic Hierarchies improve the ability of users to collaborate effectively and can save organizations significant administration time.

## Role Modeling Inbox

For organizations working with consultants and other role modeling tools, EmpowerID supports leveraging the roles and locations designed in these systems. The Role Modeling Inbox integrates external role and access management with EmpowerID by providing a set of inboxes into which roles and access changes can be published. Configurable rules within EmpowerID determine if these upstream decisions are automatically put into effect or must go through workflow approval processes before becoming active.

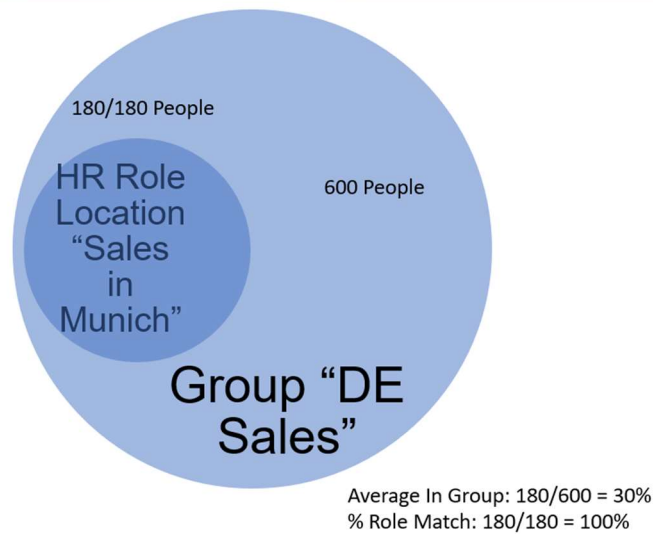
## Top-Down Role Mining

After establishing the organization's initial Business Role and Organizational Location trees, Top-Down "Analytical" Role Mining can optimize the existing entitlement landscape to convert direct user-assigned entitlements such as application role memberships in external systems to "role-automated" access. Compliant access requires that the entitlements granted are appropriate for the position. At this point in our process, we have our initial Business Role and Location trees with people assigned one or more of these roles based on attribute policies and RBAC Mapping to position information in authoritative external sources. What is missing is for these new roles to be responsible for granting and controlling their members' access. The users in these roles already have access directly assigned, but these are exceptions and will not yet be automated when they change positions. The users are assigned to Business Roles and Locations, and the system has an inventory of the entitlements they hold from all inventoried systems. To optimize these and go forward with Compliant Access Delivery requires these direct user-assigned entitlements to be reassigned to the appropriate Business Roles and Organizational Locations. This transforms static assignments lacking business context or justification into compliant role-managed access.

Analyzing the Business Role and Location hierarchies to optimize and reassign direct user access to the optimal Business Roles and Locations within the two trees would be an impossible task for even a team of people using manual spreadsheet analysis techniques. However, it is well within the capability of a computer leveraging complex algorithms.

In the top-down analytical role mining process, an organization's existing Business Role and Organizational Location hierarchy and the inventoried access assignments of their members are used as the basis of the analysis. EmpowerID uses analytical techniques to take the inventoried direct access assignment data of who can access what and then tries to optimally fit these assignments on the Business Role and Location tree. An algorithm starts with each entitlement at the bottom of the two trees (most specific roles and locations). It then crawls up each tree until it finds the highest point that entitlement could be optimally assigned, and all the people in that role and location, and their children, would have it. At this level, the entitlement could be granted by that Role and Location, producing no net change but making the entitlement now role-managed. We can also employ fuzzy logic matching to relax the tolerances to optimize placement by looking for a 90% rather than a 100% match. Once the desired match is found, the customer can then choose to publish as a managed assignment controlled and automated by the RBAC system. In the example shown in Figure 50, below, we see a suggested match determined by the top-down engine. It has uncovered that every person in the "Sales in Munich" HR Business Role is also a member of the "DE Sales" group in Active Directory—it is a 100% match. If we accept this suggestion, we would then publish this group membership to this role and convert 180 exceptions or direct assignments to one role-managed assignment.

## Example: Good to Publish

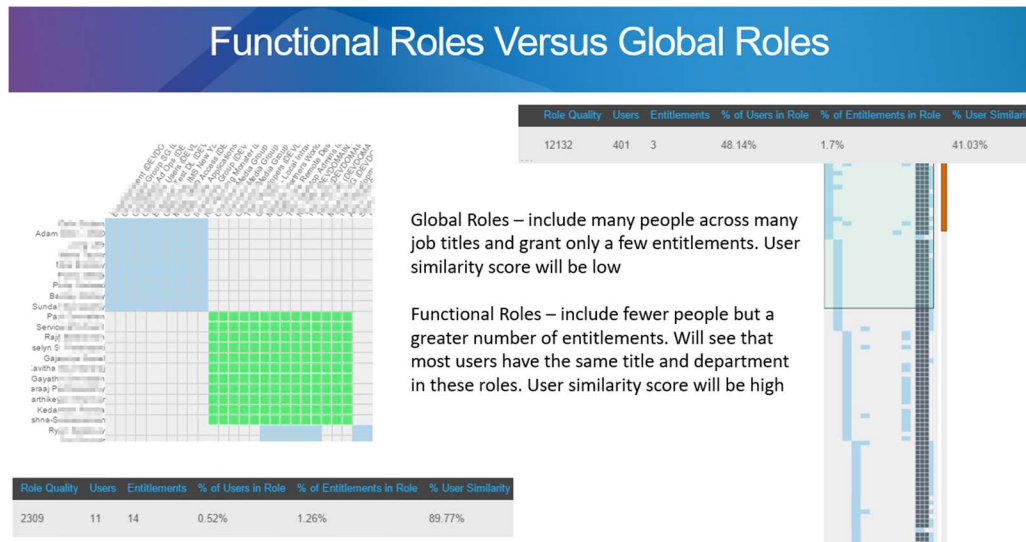


**Figure 50: Top-Down Analytical Role Mining Example – suggested match to convert the DE Sales group to a role-based assignment for the Sales in Munich Business Role and Location**

### Bottom-Up Role Mining

The top-down model is effective for optimizing access based on what a person does within an organization. Once top-down role mining is complete, much of each user's access will be delivered and controlled via Business Roles. The remaining unoptimized access assigned to users consists of the less structured team or matrix-based access and exceptions. This access can also be optimized using a technique known as bottom-up analytical role mining. Bottom-up role mining is a multi-step process that involves creating, running, and analyzing "Role Mining Campaigns."

Role Mining Campaigns analyze entitlement and user data using powerful machine learning algorithms to produce optimal "candidate roles" containing combinations of people and entitlements. These combinations are then analyzed and accepted or are further manipulated to create subsets of combinations. Once these candidate roles are accepted, they can be published as standalone Management Roles, mapped to Business Roles and Locations, or used to create new Business Roles and Locations.



**Figure 51: Role Visualizer showing examples of candidate functional and global roles**

Some benefits of analytical Role Mining include:

- Reuses existing roles and HR position information.
- Supports role optimization and clean-up efforts to remove unneeded access.
- It makes roles more intelligible as business roles match more closely HR structure.
- Dramatically reduces the effort required for manager access recertification.
- Role mining can convert 80% of exceptions to automated access.

### Compliant Identity Lifecycle (Provisioning Engine Enforcement)

The most common method of granting and enforcing Compliant Access relies on the EmpowerID provisioning engine to provision (“Push”) permissions or application role membership to managed systems using connectors. Initially, policies defining Compliant Access and business risk are modeled in EmpowerID, but the resulting access must then be manually granted and revoked by the EmpowerID provisioning engine. The provisioning enforcement model is the most common and available for many systems with out-of-the-box connectors. This model works for any system supporting the SCIM standard and can be extended to almost any other by creating a custom connector.

Provisioning Engine Enforcement happens continuously as access assignments are requested, revoked, and user attributes change in upstream authoritative systems. The most common source for large-scale readjustments to a user’s access typically comes from the HR or HCM system. Many organizations use an HCM system to maintain user data for employees and to initiate all status changes. This includes the pre-hire interview process, start-date, transfers, and terminations—all of which are managed and initiated within the HCM. EmpowerID integrates with an organization’s HCM to detect these life cycle changes and then automates Compliant Access management throughout the JML processes. EmpowerID supports all the major HCM systems, including any that support the SCIM standard.

The definition of Compliant Access in the Identity Lifecycle is the state against which the current state must

continuously be measured and adjusted versus a series of events that must be triggered. Defining Compliant Access is done in EmpowerID as part of the process to design both the position-based roles and the risk policies to detect and prevent risk. These roles and policies are the benchmarks against which all users' currently assigned access is measured. To know at any moment who has access to what, EmpowerID continuously inventories all relevant organizational systems to retrieve users, roles, and technical entitlements. The actual inventoried state is then evaluated against the desired "compliant" state and risk policies to detect compliance gaps.

Gaps in compliance most often occur based on JML lifecycle changes in the HCM system. Joiners are quickly identified as gaps due to missing access entitlements based on their roles. Whether changing jobs or locations, movers are usually seen as having incorrect access because they lack some access required for their new position yet still retain access to their old (either way, they are now non-compliant). Leavers are users that have been marked as no longer with the organization and, as such, all their access is non-compliant. EmpowerID's state-based Compliant Access Delivery engine continuously recalculates these variances of actual versus desired and automates the provisioning engine enforcement of new access and revocation of uncompliant access.

No two organizations' JML processes are identical, so a cookie-cutter approach is not an option. Attempting to bend an organization's operations to the configuration options available in most IAM platforms can be painful, and often results in the loss of any automation benefits. EmpowerID has a unique DNA among all IAM vendors as it was developed entirely on a Business Process Automation or "low code orchestration" platform.

In the EmpowerID model, entire processes are described and automated in a visual workflow, not merely as human approval processes. EmpowerID's "everything is a workflow" model's unique flexibility allows organizations to maintain their business requirements for the identity life cycle without compromising costly and unsupportable custom development. EmpowerID's JML processes follow the 80/20 rule, where 80% of the options an organization needs are configuration settings. The 20% that remain are those unique requirements that are handled in the flexible, visually designed workflows. This model's flexibility allows for much greater automation, better-enhanced reporting, and more stringent enforcement of Compliant Access policies.

### **Compliant Access Automation Benefits**

The benefits achieved from automation of Compliant Access throughout the Identity Lifecycle can be felt within the HR and IT spheres and throughout the organization. The clarity provided by a well-defined and documented Compliant Access Delivery system is the key to both understanding and bridging the gap between IT and the business. Doing so empowers them to partner, increases organizational security, and drastically cuts costs through automation. Organizations automating Compliant Access Delivery are proven to achieve measurable increases in cost savings, decreased outsourcing expenses, quicker delivery times on service, reduced overall organizational risk, and less cumbersome compliance and recertification.



Other key benefits:

- Productivity and first impressions – day one birthright access for new employees and contractors eliminating delays due to the back and forth between HR & IT.
- Security – timely and complete deprovisioning in minutes.
- Audit/compliance – a measurable benchmark for compliant access reducing the cost of proving compliance for auditors.
- Decreased manual requests for access – automating the bulk of user access based on roles reduces the workload for end users, their managers, and IT staff.
- Improve data freshness and fidelity – ensures employee attributes stay up to date everywhere with bi-directional sync that gets critical IT data, like email and phone, back into your HR systems.
- Decrease security risks when employees leave – HR triggers automatically closes security loopholes upon employee departure with real-time deprovisioning and access suspension.

### **Compliant Access Self-Service (Exceptions Management)**

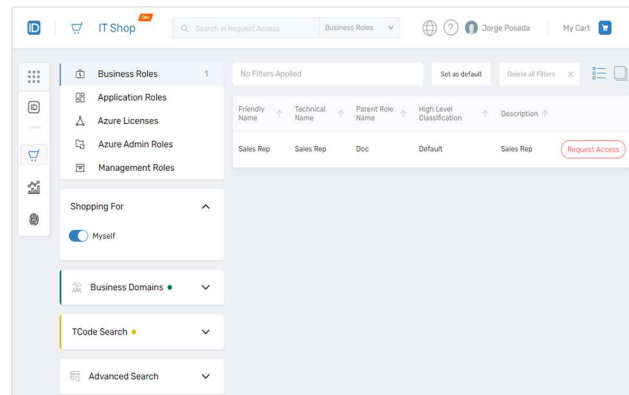
Where end-users and Compliant Access Delivery are concerned, the overall goal is to reduce the need for them to request additional access (aka “exceptions”). Any access not granted by a person’s roles is considered an exception and must go through a controlled yet easy-to-use process before being granted. By implementing a well-defined role model driven by changes in authoritative systems and enhanced by Role Mining and optimization, EmpowerID expects exceptional access to remain under 20% of a user’s total access. Exceptions represent an additional risk, create manual work for processing and approval, and extra audit work during compliance recertifications. EmpowerID’s best practice approach to exceptions management ensures that exceptions are always based on proper justification, are traceable and auditable, manageable, and, whenever possible, temporary.

### **Powerful Search Filters and Access Intelligibility**

One of the key challenges with any end-user access request facility is the difficulty for business users to identify and find the access they require to achieve their desired outcome. End users are requesting technical IT access to perform activities in the organization’s business processes for which they are responsible. The challenge in identifying and finding this access is a disconnect between the unintelligible technical system entitlements and names and the business activity access they wish to acquire. As discussed previously, EmpowerID bridges this divide with Business Functions that describe the business activities that the technical entitlement enables for a user. Function mapping dynamically categorizes and classifies even the most complex technical roles to visually display the activities they allow. Roles and entitlements can also be classified with metadata that defines to which parts of an organization’s Business Processes they belong. In addition to these intelligent search capabilities, users may also use other critical search criteria such as application, owner, system type, etc.

As shown in Figure 52, below, EmpowerID provides a best-in-class access request microservice known as the IT Shop that allows them to search for, locate, and request access to licenses, services, etc. As an additional access request interface option, EmpowerID extends ServiceNow’s capabilities with an Orchestration Pack

that enables end-users to initiate compliant access requests within the ServiceNow interface and have EmpowerID fulfill them.



**Figure 52: EmpowerID's IT Shop**

## Eligibility

A critical aspect of simplifying the end-user experience for access requests is to ensure that only compliant access can be requested. If all end users are presented with the same catalog of requestable items, they must filter through large amounts of data to find and request the required and relevant access. This not only makes for a poor user experience but also quickly becomes both overwhelming and confusing. However, exposing unnecessary data also creates a severe security vulnerability as external users or potentially malicious actors may browse the entire catalog of the organization's most sensitive roles and resources. Also crucial for regulatory compliance is to blacklist or explicitly deny the ability of certain groups of users ever to see or request specific roles and resources to enforce country-specific restrictions such as the International Traffic in Arms Regulations (ITAR).

EmpowerID offers the powerful "Eligibility" policy engine to control which users may see and request which roles and resources in the IT Shop. Eligibility policies may apply to users by attribute query, role, group, or other criteria, making it easy to target who receives which policies and have this assignment automated and maintained throughout their lifecycle. To further ease the administrative burden, eligibility policies can be applied to all requestable items of a type, either on a one-to-one basis or by location. They can also be applied to members of roles to control what they may see and request in the IT Shop.

## Inclusion/Exclusion Rules

Eligibility policies can be defined as either inclusion or exclusion rules. Inclusion rules define the items a user is authorized to see and request in the IT Shop. These rules can be customized to ensure that they can only see and request those items that make sense for them to order. For example, with a multinational company, you would not want a Field Sales employee in Austria to see the same requestable items as a Developer in Brazil. The catalog of requestable roles and resources should be different to provide a pleasant user experience and ensure that it is impossible to request access to unnecessary resources. In contrast, exclusion rules would be created as a protective measure to enforce regulatory restrictions and ensure that specific classes of users are not accidentally given the ability to request items that they should not. A sample

eligibility policy is shown below in Figure 53.

**Figure 53: Eligibility policy being applied to a Person**

Eligibility policies also include the capability of affecting the approval flow for an item requested by a user. As shown in Figure 53, above, when assigning eligibility policies, the policy author may assign an Eligibility Type for the assignment. EmpowerID has three eligibility types:

- **Eligible** – Users can request items in the IT Shop, and unless the requesting person has the RBAC delegations needed to grant the access being requested, the request will go for approval.
- **Pre-Approved** – Users assigned the policies are pre-approved for the items to which the policy is applicable. When the IT Shop user later requests access, it will not require an approval step before being fulfilled.
- **Suggested** – The IT Shop item will show a “Suggested” additional item they may request because of their existing roles or in the context of a role they are currently requesting. The item will still follow standard approval routing rules.

## Approvals and Approval Routing

Built on a workflow paradigm, EmpowerID includes a powerful approval routing engine and user-friendly interfaces for task tracking and decisions. As mentioned previously, eligibility and approval flow policies are used when calculating whether a request requires approval. If so, how many approval steps and to whom should the tasks be assigned at each step. The approval process itself is dynamic and considers the requestor's roles, the requested items' sensitivity, and both an organization's risk and its Segregation of Duties (SOD) policies. Based on these factors, an item may require many approvals followed by an additional SoD approval by the risk owner or entirely skip the approval process.

Approvers are notified via configurable and localized email notifications with reminder emails configured based on flexible policies. All decisions at each step in the process are logged and traceable up to and

include the final access.

## Access Request Policies and Temporary Access

Compliant Access Delivery's overall goal is to reduce the need for end-users to request additional access, aka “exceptions.” Access not granted by a person’s roles is considered an exception and must go through a controlled yet easy-to-use process before being granted. In addition to being compliant, a vital component of a Zero Trust strategy is that access should be assigned based on least privilege. Least privilege dictates that the access should be the minimum required to perform the desired activities and that the privileged access should be given for the shortest amount of time needed. If held by your admins permanently, even compliant privileged access presents an unneeded attack surface for hackers. The question can be asked, “do your admins truly require their admin privileges while they lay asleep in their beds at night, and hackers are busy probing for open doors.”

EmpowerID pairs “pre-approved” Eligibility policies with what is known as Access Request Policies to enforce least privilege from a time restriction perspective. Every protected entitlement registered in the EmpowerID system is governed by its Access Request Policy. As shown above, in Figure 53, the Access Request Policy controls which protected entitlements permit only temporary access, the maximum duration for the access, and if this access is renewable by the end-user.

The screenshot displays the configuration interface for an Access Request Policy, titled "General". It is divided into two main sections: "General" and "Time Restrictions".

**General Section:**

- NAME:** High Security Groups
- DISPLAY NAME:** High Security Groups
- DESCRIPTION:** High Security Groups
- Allow Access Requests
- APPROVAL FLOW POLICY:** High Security Groups Approval

**Time Restrictions Section:**

- Time Restrict Access
  - DEFAULT ACCESS DURATION (MIN):** 480
  - MAX DURATION (MIN):** 2880
- Allow Duration Selection
- Allow Request Extension
- Extension Requires Approval

**Figure 54: Access Request Policy time-restriction settings**

Revisiting our sleeping privileged admins example, we can see that Access Request Policies allow admins to be pre-approved for the access they require to perform their administrative tasks without the access being granted at all times. Admins can activate the access they require when needed, with it expiring after the policy-specified interval. This process ensures that least privilege is followed and reduces the attack surface

to a minimum and, therefore, overall risk.

## Compliant Access Recertification

Regulated and security-conscious organizations must perform a periodic review of access assignments. These reviews ensure that the assignments are still in compliance with the organization's risk policies and appropriate for the assigned person's job duties. As discussed previously, this is a challenge for organizations without an optimized role-based system automating user access, as managers are forced to review large numbers of static exceptions per employee. Furthermore, because these technical entitlements are, by nature, specific to each of the systems for which they grant access and are likely named using some form of cryptic, non-business-related, and hard to understand terminology, this increases the difficulty and workflow. The manager is then forced to decide between a lengthy investigation to determine each cryptically named entitlement's security implications or “rubber-stamp” them without understanding either risk or result.

EmpowerID's access recertification process enables organizations to periodically review and validate user access to Cloud and on-premises applications. EmpowerID continuously inventories key systems to ensure an accurate picture of user access to critical resources. Non-inventoried systems can also periodically import their data into the Identity Warehouse for analysis. A built-in access review engine disseminates this information so that auditors save time reviewing user access privileges.

Most importantly, EmpowerID can solve the key challenges most organizations face with Access Recertification. First, by drastically reducing the volume of individual exceptions that managers and role owners must review through role mining, optimization, and automation. Second, providing a Business Function view of users' access assignments provides clarity and meaning in a language they understand and empowers business users to make quick, informed decisions.

The screenshot shows the EmpowerID interface for a manager reviewing a direct report's access. The user being certified is Aarthi Raghavendra, a Software Architect. The interface shows a 'Not Started' status for the recertification process. Below this, there is a table of function access assignments.

Risk Level	Global Function	Type	Assignee	Source Assignee	How Assigned	Assigned to
Critical	SaaS Admin Management Role	Person	Person: Aarthi Raghavendra	Management Role: SaaSAdmin (All Access SaaS)	Direct	Management Role: SaaSAdmin (All Access SaaS)
High	Manage Azure AD Devices	Person	Person: Aarthi Raghavendra	Account: aarthi_raghavendra_empowerid.comEXT@idfab.oomicrosoft.com	Right Match	microsoft.directory/devices/all/Properties/all/Tasks
Critical	Delete Device	Person	Person: Aarthi Raghavendra	Account	Right Match	microsoft.directory/devices/all/Properties/all/Tasks

**Figure 55: Manager view of a direct report recertification viewing their Business Function access**

Recertification campaigns are executed quickly and efficiently with features to automate, track, and notify

everyone involved at each step, from the automated generation of audits to flexible email notification policies.

## Compliant Risk Management

Compliant Risk Management seeks to ensure that the organization is delivering compliant access and that changes from native system admin consoles are not accidentally creating non-compliant access. Current access assignments must be continually measured against a definition of non-compliant access to achieve this goal. However, to define compliant access, you must first define non-compliant access. Risk is a part of the business domain, and these must be defined and owned by business users as they relate to your organization’s specific industry and business processes. Defining your organization’s risk policies based on toxic combinations of technical entitlements, such as application groups or roles, is not a viable option. These technical objects have little meaning to business users, and the activities they enable and the risks they pose are easily obscured and change as underlying access shifts.

Risk management must be based on business functions in the business users' language and describe the actual activities that can pose a risk to the organization. Business users define which functions or activities are considered higher risk and merit additional oversight. They define which combinations of functions could lead to fraud or misdeeds if held by the same person. Defining risk policies at the function level enables you to map these to the actual business activities an organization must monitor and control. Mapping technical roles, entitlements, and fine-grained permissions to functions allow the risk Management system to know which users may perform which functions and the trail with which they acquired these capabilities.

Global Risks    Global Functions    Mitigating Controls    Rules    Mitigations    Pre-Mitigations    Violation History

Rule History

Locations	Enabled	Name	Risk Level	Risk	Type	# of Rules Using	# Violations
Business Structure	✓	F029	Low	Adjust the AR subsidiary balance using AR payments and then conceal with journal entries	Segregation Of Duties	1330	7316
Hosting Organization	✓	F006	Medium	Pay a vendor invoice and hide it via asset depreciation	Segregation Of Duties	918	6440
Business Domains	✓	F028	Low	Adjust the subsidiary balance using the vendor invoice entry and then cover it up using journal entries.	Segregation Of Duties	1292	6190
Business Processes	✓	F012	Low	Maintain asset and capitalize or add costs to master record	Segregation Of Duties	459	4100
Controlling	✓	P003	Medium	Create fictitious vendor invoice and initiate payment for it	Segregation Of Duties	918	3459
Financial Accounting	✓	P001	Medium	Create fictitious vendor and initiate payment to the vendor	Segregation Of Duties	1258	3154
Accounts Payable	✓	S004	Medium	Change customer master and enter inappropriate invoice	Segregation Of Duties	756	2941
Accounts Receivable	✓	P052	Medium	Create fictitious vendor invoice and initiate manual checks for it	Segregation Of Duties	476	2938
Plan financials							
Record / settle transaction							
Invest to Divest							
Plant Maintenance							
Sales & Distribution							

Figure 56: Risk Policies

Risk Management controls are typically described as being either “Preventative” or “Detective.” Preventative controls are the real-time checks when access is being requested or assigned to determine if the assignments violate any risk policies. When detected, the assignments go through another level of

approval by the risk owners to accept the risk and add some mitigating controls or reject the risk and prevent the access assignment. Preventative controls are easier to implement as the risk engine is only asked a smaller focused question based on the items being assigned and the recipient's existing access.

Detective controls are much more data and processing-intensive for the Risk Management system. On any given day, thousands of access and attribute changes occur across hundreds of an organization's on-premises or Cloud systems outside of the Risk Management system's control. These changes often produce ripple effects that lead to massive changes driven by inherited policies and lifecycle events that require user access readjustment. New risk violations must be "detected" by the engine To maintain compliance. This is only possible by continuously reanalyzing all access, attribute, and entitlement data pooled from these external systems. EmpowerID takes a big data approach by boiling down these access assignments' net results to manage this complex challenge. In doing so, the EmpowerID engine can detect violations even through multiple disconnected inheritance hierarchies and dynamic policies. Furthermore, this engine also captures a complete picture of exactly how the user is triggering the violation and the roles or entitlements from which they receive the segregated business functions.

Regardless of the method, whether by preventative or detective risk controls, violations of risk policies must be routed to risk owners, where they then decide to allow or refuse access.

This violator	for this Risk	Mitigation Status	Mitigator	Rule	is assigned to this assignee	and also to this assignee	Still Acti
Moctar Mohameden	F012	Not Mitigated		ABIF - J1AI	Group:Transaction_ABIF	Group:Transaction_J1AI	
Moctar Mohameden	F007	Not Mitigated		ABIF - MIGO	Group:Transaction_ABIF	Group:Transaction_MIGO	
Moctar Mohameden				Accountant - Accountant	Management Role:Accountant	Management Role:Accountant	
Moctar Mohameden				CJ44	Group:Transaction_CJ20N	Group:Transaction_CJ20N	
Moctar Mohameden				CJ88	Group:Transaction_CJ20N	Group:Transaction_CJ20N	
Moctar Mohameden					Group:Transaction_DB02		
Moctar Mohameden					Group:Transaction_F.07		
Moctar Mohameden	S006	Not Mitigated		F.13 - VF01	Group:Transaction_F.13	Group:Transaction_VF01	

**Violation Details:**

**Risk Type:**  
Segregation Of Duties

**Local Risk:**  
Clear balance and change billing doc for same customer

**Description:**  
Person: Moctar Mohameden (Moctar). Matches Toxic Combination: F.13. Has a Toxic Combination with: VF01

**Figure 57: Risk Violations**

Risk violations are logged and tracked, with risk owners alerted of violations pending their decision. The risk owners can analyze how the risky access was obtained and decide to allow the risk and add optional mitigating controls or opt for the violation to be corrected and the risky access removed.

The reality for organizations today is that enterprise risks are scattered across many Cloud and on-premises systems and are often acquired by a dangerous combination of cross-system access. To gain visibility and control over these risks, EmpowerID provides one of the largest IGA connector libraries available with the ability to connect and consume even the most idiosyncratic permissions models and inheritance used within your applications.

## CONCLUSION

In conclusion, it is easy to see how the fragmentation of the modern IT landscape caused by the adoption of a Cloud-first model has increased the importance of having a well-managed authorization strategy and the complexity of achieving this goal. A well-designed authorization system is essential for managing the day-to-day activities to control who has access to what and enable organizations to safely undergo a digital transformation moving more of their business online and into the Cloud. In this white paper, we've looked at how applications make decisions and externalizing decision-making. We also analyzed and discussed the various attribute-based and role-based approaches for application authorization, including their applicability, strengths, and weaknesses. A discussion of the RBAC versus Attribute-Based Access Control (ABAC) models highlighted the benefits and gaps in each and a new approach dubbed Policy-Based Access Control (PBAC) was introduced.

In summary, we learned that no single method covers all use cases and application technologies. The overall goal of Compliant Access Delivery requires using the best of each approach implemented following the principles of Zero Trust and Least Privilege. When executed correctly, this automates access assignments and reduces an organization's attack surface, making it harder for attackers to find privileged credentials and offers them fewer capabilities to perform malicious activities when they compromise a privileged account.

## REFERENCES

- [1] National Institute of Standards and Technology, "Role Based Access Control," NIST, 21 November 2016. [Online]. Available: <https://csrc.nist.gov/projects/role-based-access-control>. [Accessed 9th February 2021].
- [2] V. Hu, D. Ferraiolo, D. Kuhn, R. Kacker and Y. Lei, "Implementing and Managing Policy Rules in Attribute Based Access Control," 19 November 2015. [Online]. Available: [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=919151](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=919151). [Accessed 05 February 2021].
- [3] Ekran, "Role-based Access Control vs Attribute-based Access Control: How to Choose," Ekran, 04 February 2019. [Online]. Available: <https://www.ekransystem.com/en/blog/rbac-vs-abac>. [Accessed 5 February 2021].
- [4] M. Machulak and J. Richer, "User-Managed Access (UMA) 2.0," Kantara Initiative, 1 January 2017. [Online]. Available: <https://docs.kantarainitiative.org/uma/ed/uma-core-2.0-10.html#flow-summary-sec>. [Accessed 5 February 2021].
- [5] ANSI, "ANSI/INCITS 499: Information Technology – Next Generation Access Control – Functional Architecture (NGAC-FA)," ANSI, 19 March 2013. [Online]. Available: <https://standards.globalspec.com/std/1599621/ansi-incits-499>. [Accessed 5 February 2021].