

ORACLE®

FUSION MIDDLEWARE
ENTITLEMENTS SERVER

An Oracle White Paper
June 2011

Fine Grained Authorization: Technical Insights for Using Oracle Entitlements Server

ORACLE®

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle

Executive Overview	1
Introduction	1
Overview of Oracle Entitlements Server	3
Oracle Entitlements Server Architecture	4
Component Architecture	4
Auditing and Reporting	5
Scalability	6
Administration Console.....	8
Oracle Entitlements Server Authorization Policy Model	9
Overview	9
Policy Design.....	9
Building RBAC Models Using Oracle Entitlements Server	13
Building ABAC Models Using Oracle Entitlements Server	14
Supporting Claims and Federated Authorization Using OES	14
Delegated Administration.....	14
Oracle Entitlements Server Integrations	15
Java EE	15
Java SE	16
Service Oriented Architecture	17
Content Portals and Content Management Servers.....	17
Data Security.....	18
Authorization Standards	18
NIST RBAC	19
XACML.....	20
OpenAZ PEP Decision API.....	21
Java Authentication and Authorization Service	21
Integrating OES with Other Identity Management Software	21
Conclusion	22

Executive Overview

Identity management has evolved over time by externalizing individual security services such as identity stores, provisioning and authentication. Enterprises by relying on COTS (Commercially available Off-The-Shelf) solutions have benefited because they do not need specialized in-house teams and expertise to build security products. When it comes to authorization, the industry has only externalized perimeter (coarse grained) authorization for URL access, but much of the complex fine grained access logic continues to be hard coded in applications. Oracle Entitlements Server is the strategic Authorization Service for Oracle. It allows applications to externalize authorization and provides a sophisticated policy model to represent complex authorization requirements. It provides flexibility by supporting several standards based authorization models and has predefined integrations with Java SE, Java EE, .NET and SOA ecosystems.

Introduction

Identity management has come a long way since its humble beginnings. Originally all user-names and passwords were stored in flat text files; provisioning was little more than modifying these files in a text editor and authentication was just a string comparison of user passwords. Now most enterprises store users in corporate LDAPs use specialized provisioning solutions which integrate with human workflow and sophisticated SSO mechanisms. But there continue to be areas in identity management which are still directly handled by applications. Fine grained authorization is a prime example. Even though URL based perimeter authorization is externalized, the core application side authorization is often handled by custom application code. Implementing security requirements in code means the following:

- a) Security policies become brittle and all changes need to go through lengthy development and test cycles.

- b) Inability to rapidly respond to threats and security breaches.

- c) It is difficult to analyze and audit security policies and runtime authorization decisions.
- d) Application developers are forced to reinvent the wheel over and over, leading to longer development cycles, higher cost, and often rigid or lacking security implementations.
- e) High development and maintenance cost.

The underlying cause for these problems is that security is a specialized area which needs to be dealt with outside of regular applications. Application development teams are often working towards their corporate vision of delivering the best products or services. Security is often orthogonal to their revenue generating business objectives. Fusing security with application code results in combining two goals which are mutually exclusive of each other. The best way to address this problem is to *Divide and Conquer*; break the problem area into smaller pieces which can be solved individually. For applications this means security needs to be treated as an external service. Applications should not be computing decisions internally, but relying on an external service for authorization decisions.

By using an external authorization service, applications don't need to understand the intricacies of managing security. They can focus on providing the best of breed solutions to their business problems without concerns about security and compliance. A specialized authorization service should be able to support intricacies of security all the way from meeting high level government/industry compliance requirements to low level enforcement.

Overview of Oracle Entitlements Server

Oracle Entitlements Server (OES) is a fine grained authorization service which can be used to secure applications and services end-to-end across the enterprise. It provides authorization for a broad set of ecosystems including Java EE, Java SE, .NET, SOA, content management systems and databases. OES comes with several out-of-the-box (OOTB) integrations which can be *dropped into* a given deployment with minimal impact. It allows for separation of development and deployment cycles, so application developers can be agnostic of deployment issues. As OES is Oracle's strategic authorization solution for all our applications and technology it has been designed to meet the performance and scalability requirements of Oracle's largest and most complex customer deployments. Unlike authentication, authorization requests have latency constraints in order of micro seconds and a single web page access can generate over 50 individual authorization requests. OES provides a rich hierarchical policy model based on the Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC) standards. It supports multi-level delegated administration which allows for precise control over authoring and management of security policies. OES is the most mature fine grained authorization product in the market and it has been in continuous use for well over a decade.

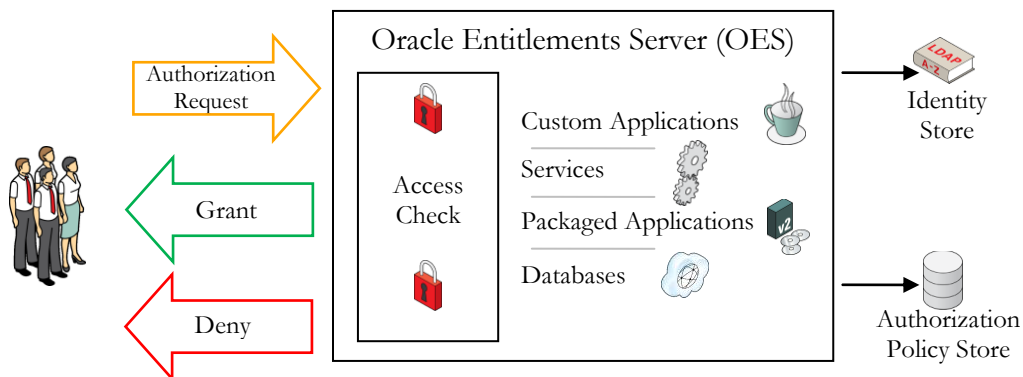


Figure 1: Overview of Oracle Entitlements Server

The figure above provides a high level overview of OES. Users as part of their normal activities, such as accessing web pages, generate access requests. OES maps these requests into a normalized form and performs checks against authorization policies. During policy evaluation OES can utilize information from external data sources such as LDAP systems, databases and Web Services. At the end, OES sends an authorization response back to the caller in the form of an Authorization Decision and Obligations (Obligations are described in the section *Policy Design*).

Oracle Entitlements Server Architecture

Component Architecture

Oracle Entitlements Server (OES) consists of the following components:

- a) **Administration Console:** The Administration Console provides a rich Web based UI for policy authoring and management. It also serves as a provisioning service and can distribute policy updates to applications. It has import, export and migration tools for policy lifecycle management.
- b) **Policy Store:** The Policy Store serves as a central persistent store authorization policies. This helps in centralized management of security. Applications can optionally bypass the Policy Distribution Service and get policies directly from the central policy store.
- c) **Security Module (SM):** This is the runtime component which includes the core authorization engine (also known as Policy Decision Point or PDP). When the SM gets an authorization request from a user or application, it evaluates this request against all relevant policies and gives a final authorization result. As part of policy evaluation, the SM can look up information from external data sources such as LDAP systems, databases, Web Services and other data sources. An SM also includes PEPs (Policy Enforcement Points), which can be used to automatically enforce OES authorization decisions in environments such as WebLogic and SharePoint among others. An SM can also be optionally configured to directly administer policy. This allows a single application to perform Policy Administration, Policy Decision and Policy Enforcement.
- d) **Policy Distribution Service:** This acts a bridge between the OES administration server and various Security Modules. Policy distribution process is initiated when an administrator decides that a set of policy changes are ready to be distributed. The Policy Distribution Service then automatically handles the provisioning lifecycle and computes the delta between what the SM already has and the latest set of policies to minimize distribution payload. Apart from ensuring transport level security the Policy Distribution Service also makes sure that only the required set of authorization policies are sent to each SM /Application. When an SM starts up it gets all pending policy updates from Policy Distribution Service. Optionally customers can replace the OES Policy Distribution Service with their own provisioning solution.
- e) **Policy lifecycle management tools:** OES provides automated tools for moving policies from development-to-test and test-to-production. Customers can also store the policy files along with their source code in a revision control systems. Migration tools can be used for policy backup, restore and disaster recovery.

As part of computing authorization decisions, Security Modules (SM) can use information from external identity stores, databases and web services (Policy Information Points or PIPs).

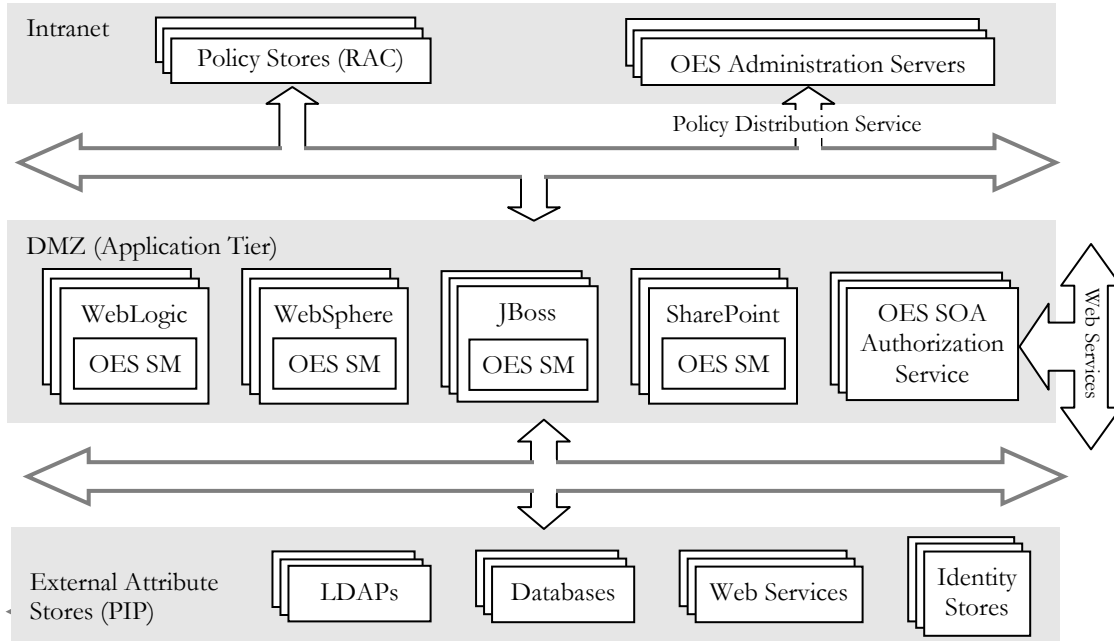


Figure 2 (Component Architecture)

Auditing and Reporting

Auditing is an important part of every security system. Analytics, compliance and reporting heavily rely on audit data. OES auditing is designed around tracking accountability. Events irrespective of how they are initiated get tracked in the audit records. This includes tracking of changes to OES authorization policies as well authorization requests initiated by applications. All authorization policy changes are tracked in the backend Management API layer so events are audited irrespective of how they were initiated. Authorization decision requests are similarly logged in the back end API so access is uniformly tracked across Java SE, Java EE, SOA and .NET environments.

For an application under heavy load, turning on full auditing will fill up disk space and use large amounts of CPU and Disk IO bandwidth. OES auditing support a flexible hierarchical audit configuration, which allows selecting the right amount of information for persistence to minimize other overheads.

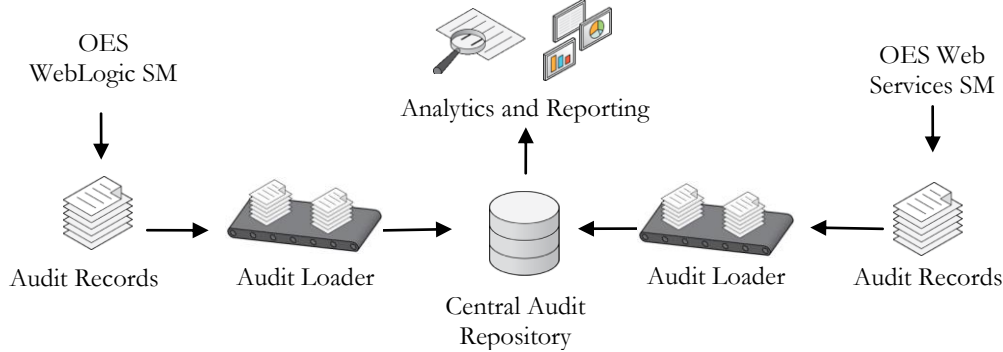


Figure 3 (Audit and Reporting)

The figure above shows the OES auditing architecture. It is based on the centralized Oracle Platform Security Services (OPSS) auditing framework. Audit records are spooled on the local file system and are periodically pushed into a central store. OES audit records are formatted such that they can be processed by analytics and reporting tools. For example, BI Publisher can be used generate reports and perform analysis on OES audit data.

Scalability

Caching

The OES caching architecture strives to maximize performance while maintaining flexibility. The figure below shows the OES caching architecture. The OES has multilevel caching. The authorization APIs have a built-in cache which can determine if the requested decision was recently computed. This minimizes expensive roundtrips for centralized OES SM deployments. Decision caching is collocated within a SM and is closely integrated with the Policy Distribution Service. When policies change, the decision cache is automatically invalidated. PIP (Attribute) caching in the next tier stores information fetched from an external data sources such as databases, LDAP systems and Web Services. In addition to these caches, users can add custom caching or fetch data from enterprise grids. This multi-tiered caching architecture allows applications to avoid network roundtrips and expensive policy computations.

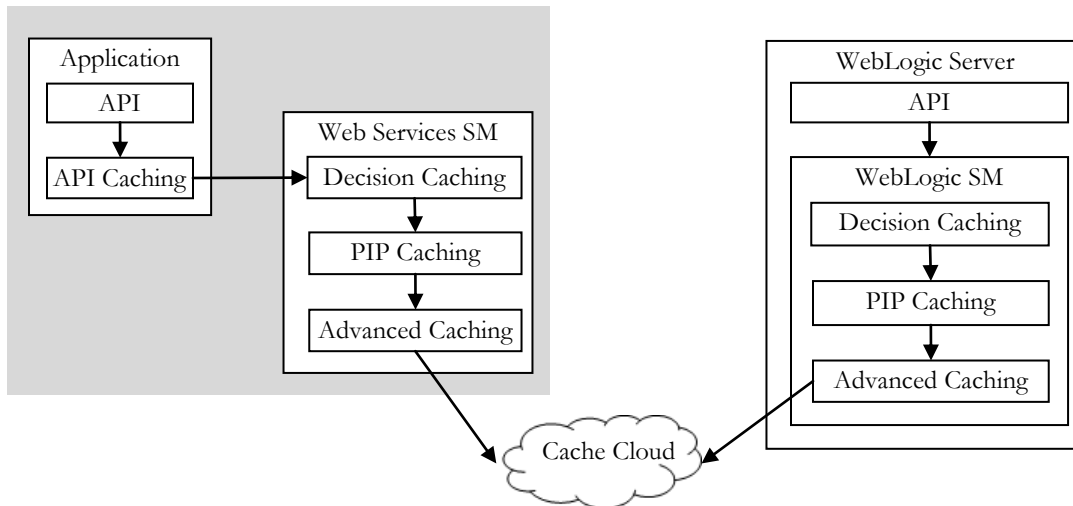


Figure 4: Caching Architecture

SMP and Multi Core Architectures

The OES authorization engine is innately multithreaded. It is designed to scale up based on applications' needs, available cores and CPUs. There is minimal contention and locking within the authorization engine. This allows an application to scale across multiple threads and CPUs. Maximum thread and connection pool sizes can be specified to limit resource usage.

Clustering, Load Balancing, Failover and High Availability

The OES configuration and deployment architectures allow for easy replication. OES has simplified installation and configuration mechanisms which allow deployments to easily scale up and scale down. OES supports full end-to-end redundancy and has no single point of failure. All components can be deployed in fully redundant mode, and every OES component can be configured with multiple endpoints and supports transparent failover between them. The OES Administration Server has built-in support for multiple policy stores, identity stores and Single Sign-On (SSO) providers. Multiple OES Administration Servers can be deployed in primary/hot-standby mode. OES Runtime API can be configured with redundant SMs. If there is a connectivity problem with the primary SM, the API transparently fails over to a backup without impacting the application. Likewise the OES SM supports redundant endpoints, it can be configured with multiple Policy Store instances (using Real Application Clusters – RAC technology), administration servers and attribute repositories (PIPs). When an SM detects an error it transparently fails over to a standby instance. All OES components support installation and configuration from command line in non-interactive mode. This allows entire deployments to be automatically recreated, for disaster recovery.

Administration Console

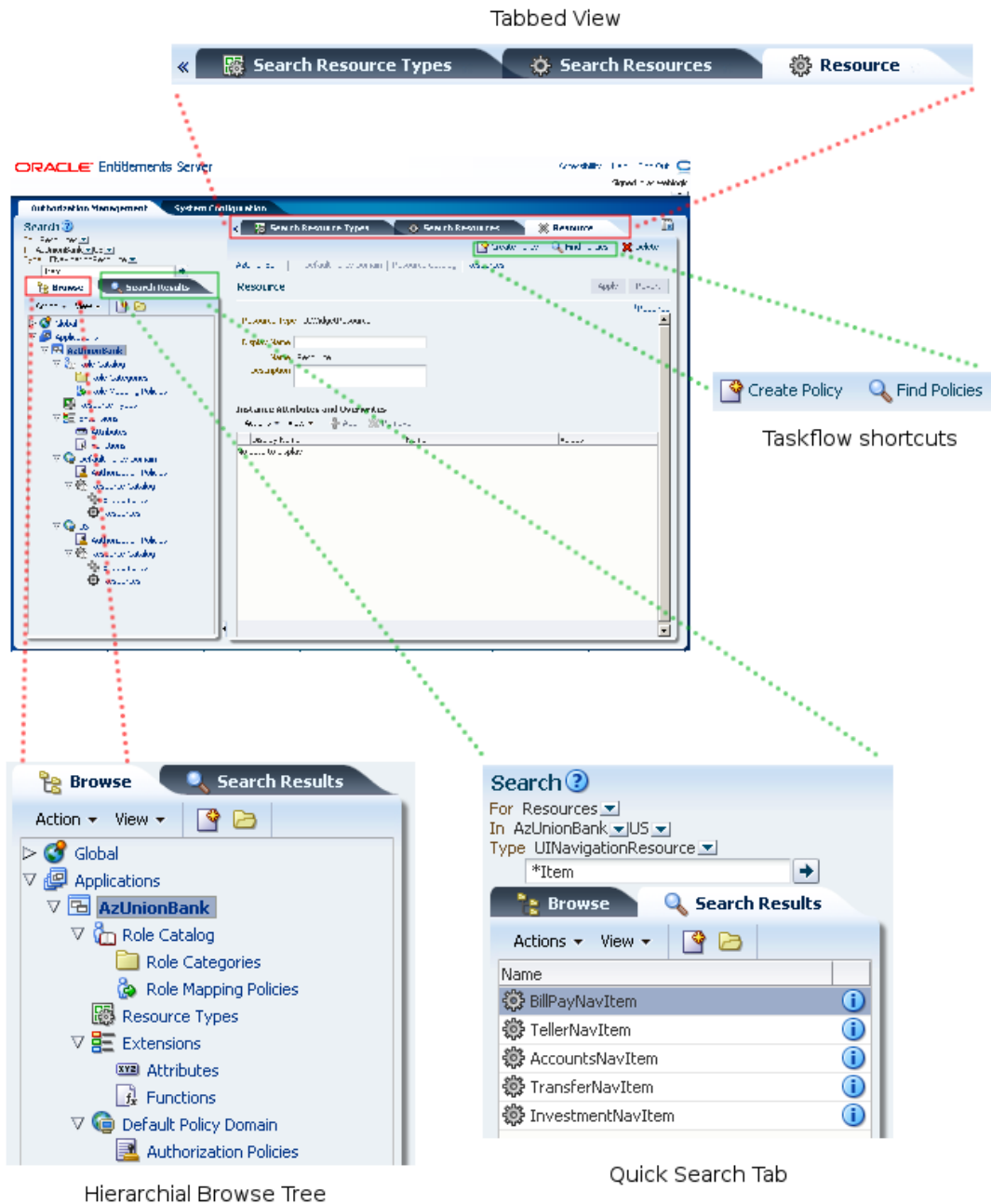


Figure 5 (OES Administration Console)

The OES Administration Console (UI) is structured around hierarchical and search concepts. For navigation, users can either step through a hierarchical browse tree or they can quickly perform a search and directly locate objects. The structured organization is helpful for novice users because it allows

them to step through the hierarchies and locate policy objects. In contrast experienced users prefer search because they know what they are looking for and need direct access to various policy elements.

The OES Administration Console (UI) provides shortcuts to simplify common task flows. For example, Resource screens have shortcuts to Policy Search and Policy Creation screens. Administration Console tries to minimize the number clicks and context switches needed to perform common task flows with a tabbed UI layout that allows users to perform multiple tasks in parallel. The Administration Console is built using Web 2.0 technologies and provides facilities such as drag and drop. The look and feel of the UI is customizable.

Oracle Entitlements Server Authorization Policy Model

Overview

OES strives to provide a policy model which closely represents common business, application and service flows. Its goal is to provide users with a *Policy Modeling Tool Kit* which allows them to easily construct models for representing corporate compliance and regulatory requirements. A simple one to one mapping of individual business roles and privileges often leads to *policy bloat* and more importantly doesn't capture the underlying process and object relationships. OES uses hierarchy as basis for policy modeling. This facilitates grouping of objects based on their relationships. Furthermore it allows attribute and privilege inheritance. The OES hierarchical resource, role, policy, and policy domain hierarchies structures allow for exponential reduction in size and complexity of policies.

OES supports ABAC (XACML), RBAC (NIST RBAC), ERBAC (Enterprise RBAC) and JAAS policy models. Based on business needs either one of the policy models can be used or several of them can be used in combination.

Policy Design

Business process and objects often tend to be hierarchical. For example, a loan approval process consists of multiple sub processes and some of these sub process might constitute of additional finer grained processes. In a similar way a web application can consist of multiple web pages, each page has different sections and each section has different pieces of information. The figure below shows a hierarchical decomposition of a web page. The left side shows the structure of a regular web page and the right side shows a Resource hierarchy for the same structure. There is one-to-one correspondence between Web page elements and Resources. If a user is not allowed to access the main page (shown in the figure below), they are automatically denied access to all underlying sub-elements. A request is therefore denied even if a user tries to bypass regular checks and directly access underlying content.

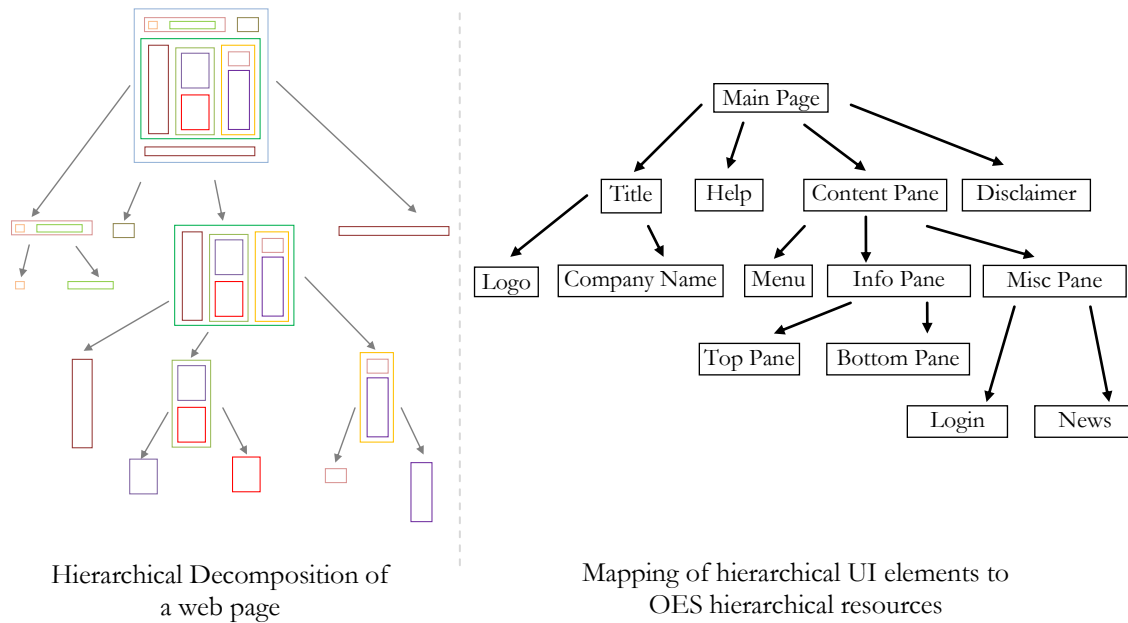


Figure 6: Mapping Webpage to OES resource tree

Role Based Access Control (RBAC) is a simple, well-understood technology. It has been in active use for over three decades. Currently RBAC is one of the most widely used security models across the industry and many security professionals understand security modeling using RBAC. Roles are the foundation of RBAC. Enterprise Roles (or Groups) form the basis for coarse grained authorization. Enterprise roles are statically assigned at the time of authentication and they last for the duration of the login session. This type of role assignment leads to excess permissions. In contrast Application Roles (fine grained roles) are dynamic in nature and come into existence at the beginning of an authorization request and are deleted once the authorization decision is computed. Application Roles are assigned based the context. For example, consider a grant for *Manager* role across the entire company vs. a grant for *Manager* role only when the authorization request is in the context of their direct reports.

OES allows dynamic assignment of Applications Roles based on policy. Consider this sample use case: universities allow much flexibility to professors on how they want to manage their courses (classes). They are allowed to change location, exam schedule, enrollment, grades etc. These privileges are restricted only to the main instructor of the course. This restriction can be summarized as an authorization policy: “A professor can only administer university courses that they teach”.

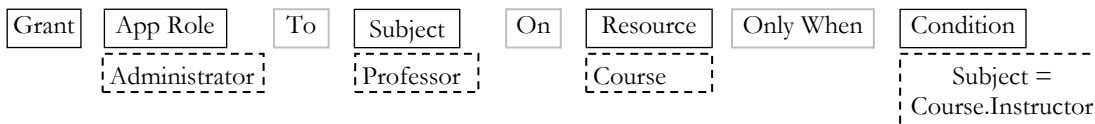


Figure 7a: Structure of Role Mapping Policy

Role Mapping Policies can also specify Deny (or Negative Grant). The policy below states “One hour after a bank closes none of its employees can have the role Bank Teller (i.e. cashier)”

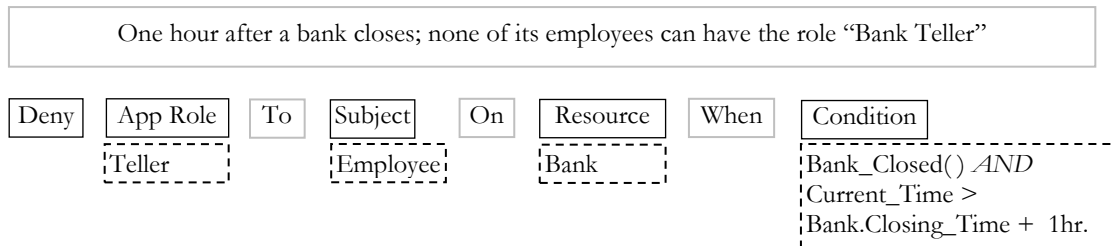


Figure 7b: Structure of Role Mapping Policy with deny

Business roles are often structured hierarchically. Employees in higher positions are automatically granted privileges of people in their reporting hierarchy. To model these real world relationships OES supports hierarchical roles. As shown below an R&D director is implicitly assigned the Developer, Build Engineer, Development Manager, Test Engineer and Test Manager roles. This type of structure will allow a Vice President to have full control over their organization.

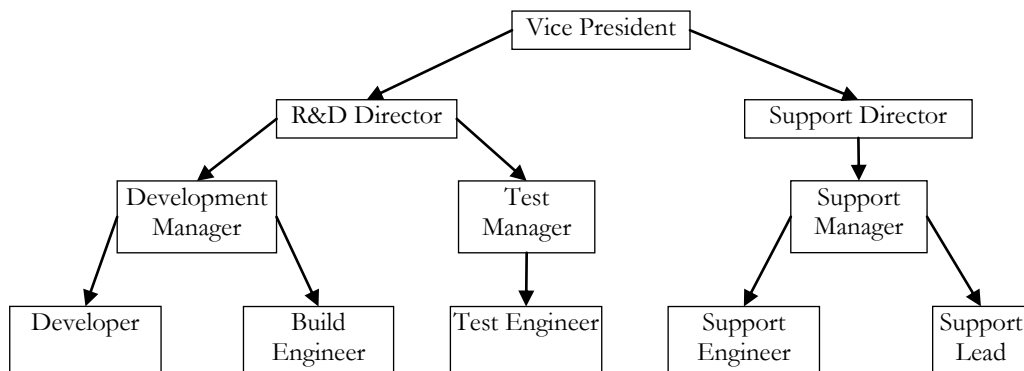


Figure 8: Hierarchical nature of Business Roles

The Condition statement in an OES policy gives additional control over when to grant/deny roles. It can have expressions based on User, Enterprise Role (Group), Resource, Environmental and custom attributes. For example, the figure below shows how to write the policy Condition “junior stock traders can only trade for less than a million dollars per day”. OES allows direct mapping of policy attributes to information from LDAP systems, databases, Web Services etc. Custom plug-ins can be invoked within conditions using OES Functions. Users can package their code as a loadable module and map their function definitions to OES Functions.

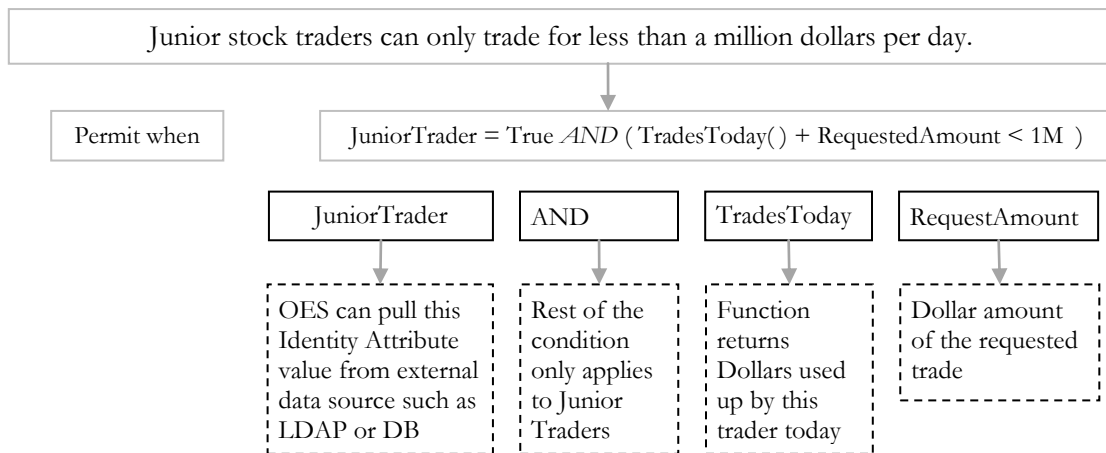


Figure 9: Sample Policy Condition

Real world processes often span more than just one resource. Using *Entitlements* OES allows users to closely model business and UI task flows. An entitlement is a combination of Resources and the Actions needed to perform a certain business task. For example, opening a bank account consists of several individual steps which include filling out multiple forms, creating entries for each account holder and then finally creating an account; when a banker is allowed to open a new bank account they will need access to several web forms, ability to check customer's financial background. An Entitlement can be used to represent an end-to-end task flow such as opening a bank account. All privileges (Resources and Actions) required for a business task flow can be grouped into a single Entitlement. Thus an Entitlement allows security administrators to work with high level business task flow instead of worrying about micro privileges. Another advantage with Entitlements is that it allows usage of regular expressions for identifying resources. Automatically generated Resource names often follow certain text patterns, so it is easier to identify collections of such resources using regular expression pattern such as "*.xls" to refer to all spread sheet documents.

Dynamic assignment of roles only addresses the first half of an authorization problem. Ultimately these roles need to be mapped to real world privileges. OES authorization policies dynamically map Users, Enterprise Roles and Application roles to Privileges. OES Authorization policy gives a Permit/Deny for a Subject to perform a certain Action on a given Resource only when the Condition Expression is true. For example, a Call Center might have a policy that allows employees to lookup customer credit card information only when they are physically present in office. This policy can be phrased as "Allow employees to view customer credit card numbers only when they are connecting from an intranet IP address".

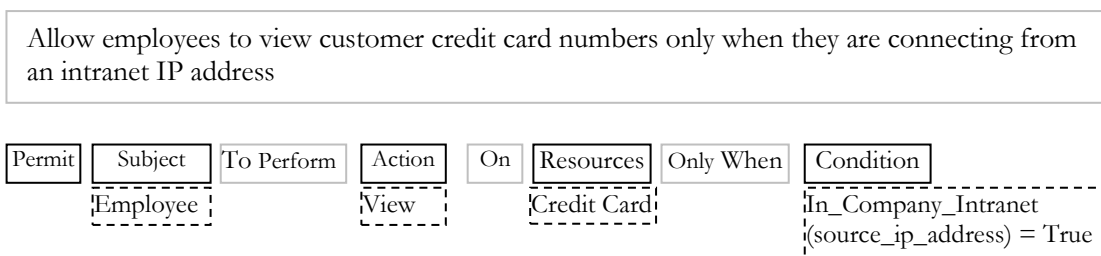


Figure 10: Hierarchical nature of Business Roles

OES also supports the XACML notion of Obligations. For certain use cases, a simple permit/deny from authorization engine is not sufficient. This happens because a certain part of the condition cannot be evaluated by the authorization engine. For example, in data security use cases a simple yes/no answer is not sufficient. Rather a search filter (such as an SQL Where clause) needs to be applied from within the database. So while evaluating an authorization request, OES authorization policy can return an SQL filter as an Obligation

Building RBAC Models Using Oracle Entitlements Server

RBAC models use roles to model enterprise security. For RBAC, Permission (or privilege) is a trivial extension of role. Roles can be divided into three categories:

- a) **Enterprise Wide Static Roles:** Users are granted these roles irrespective of what they are trying to do. For example, an Employee role can be a static assignment because it is common across enterprise. These roles are best stored in an LDAP or some other enterprise wide identity store. These roles are normally assigned at the time of authentication and are generally valid till session expiration.
- b) **Application Specific Static Roles:** These are static role assignments which are specific to a subset of enterprise applications. A user is granted different roles based on the applications they are trying to access. These roles typically don't change during a user's session. OES has facilities to directly manage assignment of these roles.
- c) **Application Specific Dynamic Roles:** These are dynamically or conditionally assigned application roles. They are assigned on an as needed basis depending on the action initiated by the user. For example, role Fund Manager should be granted to a person only on certain funds. They come into existence when an authorization request is made and they are destroyed once a decision is computed. As discussed previously OES provides sophisticated facilities to accurately control role assignments based on the context.

As seen above Enterprise Wide Static roles are easy to compute, but they are rigid. Dynamic roles need to be computed for every authorization request but they are very flexible and dynamic. However due to locality of reference, these roles can often be fetched from the cache and do not require full computation. The static/dynamic nature and scope of roles are deciding factors in choosing the correct role type. Depending on changing requirements, an Enterprise Wide role can be converted to an

Application specific dynamic role or vice versa. This type of interoperability is possible because OES is based on the NIST RBAC and ABAC standards.

Building ABAC Models Using Oracle Entitlements Server

Attributes are the basis for ABAC (Attribute Based Access Control) model. OES supports a variety of attributes (for example user, enterprise role, resource, application role and requested action); in addition custom attributes can be defined for special needs. ABAC has heavy reliance of conditions, essentially all authorization decisions are made by evaluating expressions based on these Attributes. As an authorization model ABAC offers more flexibility than RBAC because it heavily relies on dynamic computation. However this comes at the cost of being able to perform audits based on static policy data. Also the added complexity means that ABAC models are more computation intensive than their RBAC counterparts.

OES lays out a policy structure to help map business security requirements into raw ABAC models. Subjects (Users, Enterprise Role and Application Roles), Resources, Actions and attributes are the most commonly used elements in ABAC polices. The OES UI creates an easy to use abstraction around these objects to simplify the construction of policies. OES also supports Resource, Attribute and Policy inheritance which allow policies to be placed strategically so that they will be able to control large number of Users and Resources.

Supporting Claims and Federated Authorization Using Oracle Entitlements Server

.NET and Microsoft ecosystems heavily rely on Claims based models. From the OES perspective, a Claim is a signed attribute/value pair which can be verified on demand. Claims (including SAML Assertions and X-509 certificates) can be mapped to environment attributes and directly passed in as part of an authorization request. Also, during policy execution OES can get the required Claims information from the appropriate claims issuer using the OES Custom Attribute Retrievers and Custom Functions. Federation is easy with Claims because of its decentralized model. Excess federation leads to loss of centralized control, so federation should be used only when necessary.

Delegated Administration

Policy administration is a crucial part of end-to-end security. Proper checks and balances around administration are essential to make sure that only privileged users are allowed to make changes. OES allows multi-tier delegated administration at the system, application and custom sub application levels. Users and Enterprise Roles can be mapped into these administration levels. For example, an Enterprise Auditor role only has view privilege across all application policies, a QA engineer can be given view-only privilege for a subset of applications, and a developer can be given privilege to only create resources and manage policies for a certain application subcomponents.

In the figure below:

- a) Alice has view privileges only on *Application 1* policies

- b) Bob has full privileges to manage policies on *Application 1* and *Component X of Application 2*
- c) Charlie has view privilege across all applications

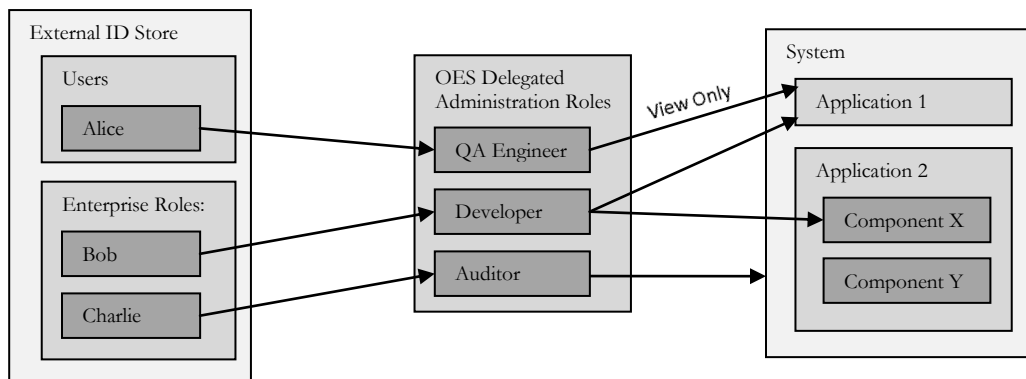


Figure 11: Delegated Administration

Oracle Entitlements Server Integrations

Oracle Entitlements Server is an enterprise authorization service that provides an end-to-end solution covering Java SE, Java EE, SOA and .NET ecosystems. OES not only ships with several out of the box integrations, but is designed from ground up to easily integrate into many environments.

Java EE

Java EE is a mature Java ecosystem which provides services for building and deploying enterprise applications. Java EE applications expose services such as EJBs, Web Services, Web applications and RMI. Containers only provide a limited form of perimeter authorization; they cannot for example, enforce restricting access to a Web Service method based on identity attributes defined in a users SAML assertion. Also containers do not have facilities for sharing authorization policies across domains, which becomes a serious restriction for large enterprises.

OES supports standard Java EE containers including but not limited to WebLogic, WebSphere and JBoss. The OES naming mechanism supports clustered architectures. All nodes in a cluster automatically get the same configuration and policies. This allows easy management of authorization policies and configuration for large clusters. OES supports securing Java EE components such as JSPs, Servlets, URLs, Web, Services, EJBs, JDBC, JMS, JNDI and RMI.

For Java EE deployments OES provides JSP tag versions of its authorization API. These can be used by Web applications for declarative security.

For WebSphere and JBoss, OES provides container wide authorization service which can be used by any application deployed in the container. Applications can not only make authorization requests, but with the right privilege they can also view and modify authorization policy itself.

On WebLogic in addition to API and Management API support, OES also serves as a primary security provider for the container itself so applications don't have to be aware of OES. As they use the Container's services, the underlying security framework will automatically call into OES and enforce the authorization decision. So in essence for WebLogic, OES also serves as the PEP. An architecture diagram is shown below.

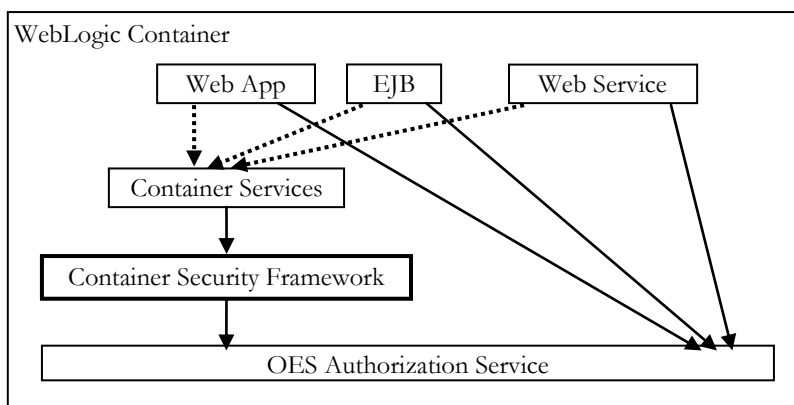


Figure 13: OES as Authorization provider for container security

Java SE

In the last couple of years, light weight Java technologies such as Spring, Hibernate and Apache Tomcat have become popular. Because of their smaller footprint and flexibility these are more suited for certain types of projects. Each of these technologies focus on a specialized problem area, for example, Spring address Inversion of Control and Aspect oriented programming. Security is often orthogonal to their primary service. OES brings enterprise class authorization to these frameworks by providing rich ABAC and RBAC policy support.

The OES Java SM is a pre-integrated form factor for Java SE environments. It comes with full set of OES capabilities for including Management and Runtime authorization APIs. Security and especially authorization is a classic example of a cross cutting concern which is not only common to all applications, but also common to all modules and objects within each application. Using Spring annotations OES can protect all the way from packages to individual methods. This way, invocations of Spring methods automatically result in a call out to OES for authorization. OES can also be natively integrated as a Spring AccessDecisionManager provider and can secure UI elements with the “authz:authorize” tag.

Hibernate relies on JAAS for declarative security. Applications using Hibernate can rely on OES as a JAAS authorization provider. OES can also be used as a Hibernate Interceptor to control read, create, update and delete operations for objects in database.

Service Oriented Architecture

Many enterprises rely on Service Oriented Architecture (SOA) as their service backplane. SOA is used to interconnect services across Java, .NET and legacy mainframe applications. OES can not only be used to secure individual SOA elements and services but it also provides authorization as a service. OES has explicit support for SOA deployment models to aid in building scalable high availability environments.

XML gateways such as Oracle Enterprise Gateway, Vordel, DataPower and Layer 7 can use OES to manage authorization for Web Services. OES policies can authorize invocation of individual APIs based on the user, invoked method and information in the message body, such as the requested Customer-Id and SAML assertions or other attributes from the soap header and body. OES can also be used to make decisions on whether to redact or encrypt certain information in the response for the Web Service request - without changing any of the web services code. To reduce latency, OES can be directly embedded into XML Gateways such as Oracle Enterprise Gateway and Vordel. Using Oracle Web Services Manager (OWSM), OES brings rich ABAC and RBAC support to the embedded Web Service Security agents in Oracle's SOA and WebLogic infrastructure.

Enterprise grids such as Coherence and Memcached primarily focus on providing robust and efficient ways of caching and distribution of data; security is often not their primary concern. This becomes a problem in sensitive environments because there is no authorization for individual object access. OES can be used as a security tier to intercept cache requests and enforce authorization. OES can protect access to Coherence Caching Services, Named Caches and Object queries by extending its native implementations.

OES supports the .NET ecosystems using the XACML standard based request/response mechanism. This allows any .NET application to directly make authorization requests to OES. OES provides a core authorization service which can be customized to meet application specific requirements.

Content Portals and Content Management Servers

Content Management Servers such as SharePoint and Oracle Universal Content Management (UCM) provide excellent facilities for storing, retrieving and sharing documents. They often come with standard facilities to secure documents. OES can extend these simple security models with sophisticated RBAC and ABAC based models. For example, a policy such as "Only employees with clearance level 4 can view confidential documents" can be easily implemented using OES policy constraints. OES can use UCM's Event Filters to intercept document operations (e.g. read, update, delete, search etc.). UCM document meta data can be directly mapped into OES attributes which can be used in authorization and role mapping policies.

SharePoint serves as both a portal and document repository. OES provides OOTB policy enforcement Points (PEPs) for securing SharePoint Sites, URLs, Pages, Portlets, Web Parts, page contents and documents. An OES HTTP module secures Web pages and the OES Web Control secures Web Parts. In addition OES provides an authorization tag library (*AuthorizationTagLib*) which allows conditional execution of code and custom UI rendering.

Data Security

In enterprises, most data originates from a database, flows through various service tiers and is finally rendered by the UI. Securing data where it originates is a guaranteed way of making sure that the information is not leaked. OES supports creating VPD based filters in the data tier as well as securing access at DAO (Data Access Object) tier.

The DAO tier within an application is responsible for handling object persistence. As part of mapping Object queries to SQL, OES policies can be enforced in the form of SQL filters. For example, when a manager runs a query to display salaries of all employees, OES can generate an *SQL Where* clause such as `where employee.manager_id = $current_user`. Because the application context is available in DAO tier, rich authorization policies can be written using all the available context information. Policies can also be written to determine what operations (for example hire, promote and view benefits) a user can perform on a given set of employees.

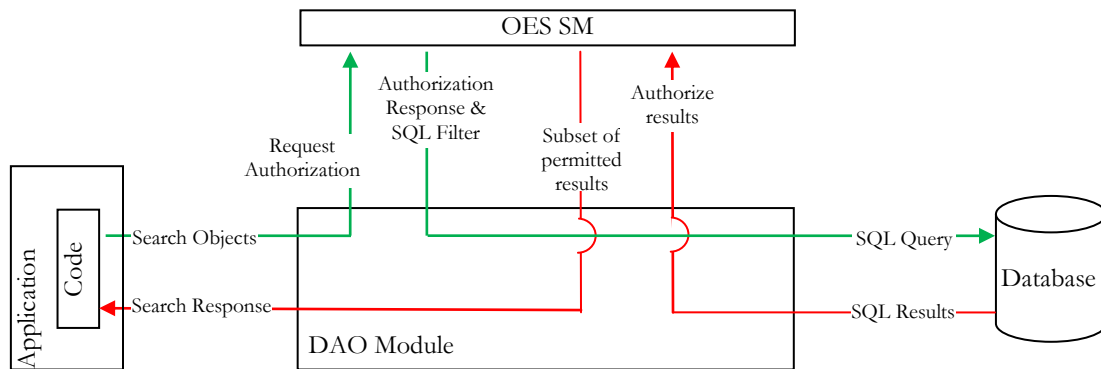


Figure 14: Fine Grained Authorization for Data Security

Sometimes information stored in a database is extremely sensitive and extensive checks need to be done irrespective of the application. For example, credit card numbers and passwords should only be shared on a need to know basis. In these situations it may be desirable to enforce restrictions from within the Database itself. OES can be used in conjunction with Oracle VPD to do Row and Column level filtering based on OES authorization policies. Because this filtering is done within the database, security policies will be enforced irrespective of the application. This solution is also useful with legacy applications which cannot externalize authorization.

Authorization Standards

There are several authorization standards and models. Each standard specializes in solving a certain type of problem. OES strives to support a wide variety of industry standards allowing customers to pick a model that suites them the best. As we shall see, every standard has advantages and shortcomings. OES allows users to pick a standard of their choice based on what matters the most. OES supports NIST RBAC, XACML, Open AZ (PEP Decision API) and JAAS standards.

NIST RBAC

RBAC is one of the most widely used security models and has been in use for over 3 decades. It enjoys broad industry support across operating systems, databases, applications, networking and Web software. In order to ensure interoperability between different products, NIST working with industry created standard ANSI/INCITS 359-2004. The NIST publication “[The NIST Model for Role Based Access Control: Towards a Unified Standard](#)” gives an overview of this standard. OES supports NIST RBAC Level 4. This is the highest RBAC level and it includes Flat, Hierarchical, Constraint RBAC and Symmetric models. The following table shows how OES policy constructs map to the NIST RBAC specification.

Level	Name	NIST Requirement	OES Capabilities
1	Flat	Users can acquire permissions through roles	OES Authorization policies allow users to acquire permissions based on granted roles
		Many-to-many user-to-role assignment	OES Role Mapping policy allows a set of users to be assigned any number of roles
		Many-to-many permission-to-role assignment	OES authorization policy allows a set of permissions to be assigned to any number of roles
		User-to-role assignment review	OES Administration Console and Management API allow review of User-to-role assignments
		Users can use permissions of multiple roles simultaneously	OES Authorization policies allow using multiple roles for obtaining a Permission. For example, both role A & B are needed in order to get privilege X
2	Hierarchical	Support for Arbitrary Hierarchies	OES has innate support for Role Hierarchies and allows Role Inheritance OES supports General Hierarchy (both regular inverted trees) and doesn't have any restrictions on Senior and Junior most roles
3	Constrained	Support for Separation of Duties (SOD) over Arbitrary Hierarchies	SOD can be enforced by using OES dynamic role mapping policies, by denying roles which are in conflict. SOD can also be checked in Authorization policies to examine granted roles.
4	Symmetric	Support Permission-to-Role Review over Constrained Arbitrary Hierarchies	OES Administration Console and Management API allow review of User-to-role assignments

XACML

XACML (eXtensible Access Control Markup Language) is a standard based on ABAC (Attribute Based Access Control). The latest version of the specification can be found at the [Oasis XACML Technical Committee Home Page](#). Currently XACML 3.0 is in draft.

The figure below is based on XACML 3.0 specification's section 3.1 Data-flow model. It shows how the OES message flows align with the XACML specification. In OES both Administration Console and SM can act as a Policy Administration Point (PAP), it is up to the user to decide on the provisioning model they want to use. The rest of the XACML components are part of the OES SM. Initially policies are created and provisioned using PAP (Step 1). When a Client tries to access a protected resource (Step 2), the Policy Enforcement Point (PEP) sends an authorization request using the OES API (Step 3). If necessary external resource name is reformatted into an OES Resource (Step 4). The PDP then gets the decision request (Step 5). If needed it will query for different attributes (Step 6 and 7). In step 8, Data is fetched from relevant Policy Information Points (PIPs). In steps 9 and 10, responses are sent back to the Policy Decision Point (PDP). The PDP computes the authorization decision and the response is sent to the PEP (Steps 11 and 12). If necessary, the PEP will fulfill any obligations (Step 13) that are returned.

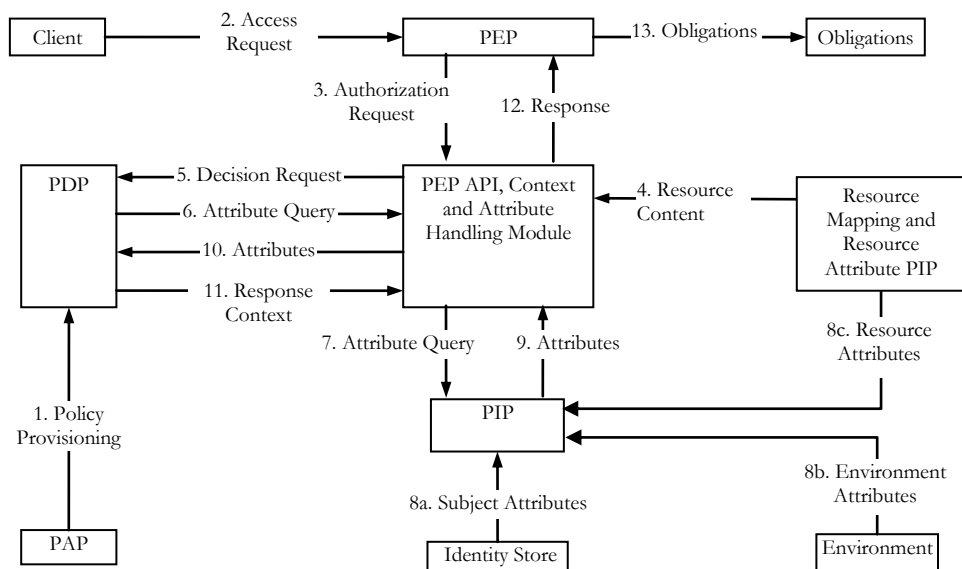


Figure 15: XACML flow diagram

The XACML request/response specification is based on a SOAP based protocol which helps in integration of different authorization services. It standardizes the message exchange format between the PEP and the PDP. This allows for independent development of the PDP and end user applications and services. OES fully supports XACML 2.0 request/response and successfully completed all stages of the Oasis XAML InterOp (2008).

OpenAZ PEP Decision API

The XACML specification does not define how the authorization service should be invoked by PEPs from applications or middleware in a high level programming language. Therefore the [OpenAZ project](#) has defined two sets of APIs with native language bindings for this purpose. They are: AzApi, which is based on the abstract XACML authorization request and response model, and PEP API, which provides a higher level abstraction based on Java objects. This API removes the need for application and middleware developers to convert to the low-level representations required by the authorization service. For example, given appropriate mappers and suitably designed policies, an application could directly provide a Java business object (e.g. medical record) as an input to a policy decision and it will be mapped into the native representations required by the authorization system.

OpenAZ PEP Decision API is the default runtime authorization API for OES. So instead of relying on a vendor's proprietary API, applications can build on top of standardized PEP Decision API; this protects long term investments in new applications and services.

Java Authentication and Authorization Service

The Java Authentication and Authorization Service (JAAS) standard is part of the [Core Java Specification](#). It defines how Java applications can use Authentication and Authorization services. JAAS authorization uses resource and principal centric views to model policies. A JAAS policy consists of principal, permission type, permission name and action. The main advantage with this model is that it comes pre-integrated with Java SE and Java EE ecosystems. An application which complies with JAAS standard does not need any changes when moving between different JDKs and Java EE servers.

OES can be used as a Java2 JAAS security provider (PDP) for both Java SE applications and Java EE containers. This means OES authorization policies can protect all the way from high level services such as EJBs to low level operations such as reading of files on the local hard disk. OES brings rich RBAC and ABAC policy model support to JAAS. For example, OES supports policies such as “A user or a given set of code is only allowed to read a file when the user's clearance level is higher than the clearance level required for the file”. To evaluate this policy, OES needs to fetch the identity attribute “User Clearance Level” from an LDAP, file attribute “File Clearance Level” from an external data source (such as database) and compare these two values. Applications need not be aware of the underlying security mechanism because the Java Virtual Machine (JVM) transparently enforces these decisions.

Integrating Oracle Entitlements Server with Other Identity Management Software

OES relies on open standards to allow easy interoperability with Oracle and 3rd party identity management products. Oracle's Identity Management stack covers the full spectrum of end-to-end identity management use cases. This allows customers to deploy OOTB pre-integrated solutions covering a large variety of use cases.

- a) Identity Stores and Virtual Directories: OES uses LDAP v2/v3 standard. OES is certified with Oracle Internet Directory (OID), Oracle Virtual Directory (OVD), Oracle Unified Directory (OUD), Oracle Directory Server Enterprise Edition (ODSEE), Sun Java System Directory Server (SJSDS), Tivoli DS, Active Directory (AD), Active Directory Application Mode (ADAM), eDirectory and OpenLDAP
- b) Identity and Role Provisioning: OES can fetch identity attributes and roles directly from LDAP directories and the databases. OES Management API can be integrated with other provisioning solutions for automatic creation of entitlements.
- c) Authentication and SSO: OES being an authorization product has several facilities for easy integration with other authentication, federation and SSO products. An authenticated users identity can be passed in the following ways:
 - a. JAAS Subject: This is Java standard for sharing authenticated subjects. After users are authenticated, their JAAS Subject can be passed into OES for authorization.
 - b. Container generated Subject: When running in a Java EE container, OES supports the corresponding container based subject (E.g. WebLogic, WebSphere subjects etc.). This allows for easy interoperability with containers security framework
 - c. Username: Applications can optionally pass in direct user name and enterprise roles.
- d) Federation, STS and Claims: An application can insert relevant claims into context attributes. Also OES can fetch the required claims from an STS during evaluation of authorization policies.

Conclusion

Embedding authorization decisions in application code leads to brittle and static policies which cannot keep pace with changing security requirements. Not having a centralized policy management and uniform enforcement infrastructure leads to security silos where each application uses a different security mechanism and the resulting authorization policies cannot be reused across organizations. Over time these types of deployments become complex, unmanageable and expensive to maintain. Lack of insight combined with poor auditing facilities leads to compliance and security nightmares. Using standards based COTS authorization solutions, enterprises can regain control over how security is managed across different organizations. Security requirements can be easily mapped to authorization policies. Centralized policy management combined with automated provisioning ensures that security policies are uniformly enforced in all applications and services across the enterprise.

Oracle Entitlements Server as an authorization service has been designed to meet stringent regulatory, business, and security requirements with minimal runtime overhead while allowing developers and administrators the ability to administer policies in a business-user friendly manner. As Oracle's strategic authorization engine, it is embedded within several Oracle products. It uses standards as a foundation to deliver highly available, scalable, externalized authorization management solution with a rich policy model for applications, middleware and databases. Oracle Entitlements Server helps organizations centrally manage entitlements, provides a central view of access rights across applications in the enterprise and generates audit records which can be used by reporting and analytics tools. It provides

developers with shared services for fine-grained authorization to ensure quicker compliance and better business agility as policies can be quickly adapted based on market, security, regulatory and business requirement changes. Oracle Entitlements Server not only supports cutting edge industry standards and features, but it also brings over a decade of experience in building a robust authorization solution which can be relied on for securing critical applications and services.



Fine Grained Authorization: Technical Insights
for Using [Oracle Entitlements Server](#)

June 2011

Author: Subbu Devulapalli

Contributing Authors: Hal Lockhart and Rich
Levinson

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com



Oracle is committed to developing practices and products that help protect the environment

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 1010

Hardware and Software, Engineered to Work Together