Oracle ® Entitlements Server 10*g* (10.1.4.3)

Introduction to Oracle Entitlements Server

September 2008

ORACLE

Introduction to Oracle Entitlements Server, Oracle ® Entitlements Server 10*g* (10.1.4.3)

Contents

4. Integration and Deployment

5. Summary

CHAPTER 1

Introduction

The following sections describe the content and organization of this document:

Note: Oracle Entitlements Server was previously known as BEA AquaLogic Enterprise Security. Some items, such as schema objects, paths, and so on may still use the term “ALES."

Audience and Scope

The document is intended for all users of the Oracle Entitlements Server, including business analysts, security architects, security developers, application developers, and administrators.

Guide to This Document

This document summarizes the features of Oracle Entitlements Server 10*g* Release 3 (10.1.4.3) and presents an overview of its architecture and capabilities. It includes the following topics:

- “The Entitlements Problem” on page 2-1 describes enterprise security in the context of entitlements and explains how Oracle Entitlements Server addresses the requirements of entitlements management.

- "What is Oracle Entitlements Server?" on page 3-1 describes Oracle Entitlements Server components, services, features, and functionality.
- "Integration and Deployment" on page 4-1 describes how Oracle Entitlements Server integrates with application environments using different deployment models.
- "Summary" on page 5-1 provides a short review of this document.

Related Information

All documents are listed on the Oracle Entitlements Server documentation home page. Among others, this documentation set includes:

- Policy Managers Guide — defines the policy model and describes how to manage, generate, import, and export policy data.
- Integration Guide — provides instructions for integrating out-of-box SSMs with applications.
- Getting Started with Oracle Entitlements Server — provides a number of tutorials that show how to use the Entitlements Administration Application to secure application resources.
- Securing OES Production Environments — provides security 'best practices' when deploying OES on production systems.
- Programming Security for Java Applications — describes how to secure Java applications.

CHAPTER 2

The Entitlements Problem

Overview

The requirements for providing security access to enterprise applications have evolved dramatically in the past decade. The advent of the Web led to the need to protect the perimeter of the enterprise with technology principally focused on "keeping the bad guys out." The introduction of Web security and single sign-on solutions pushed access and security deeper into the enterprise. Identity management systems enabled the management of millions of users. Access control and federated identity systems allowed customers and partners to reach deeper into the enterprise, increasing automation and self-service, and improving cost-efficiency.

In the last few years, changes in the enterprise application landscape have mandated a change in the approach to enterprise security. Sarbanes-Oxley mandates documented controls on who has access to information systems that affect the finances of publicly held companies. Health care and privacy laws have placed stricter requirements on access to applications and auditing of access. A rapid rise in outsourcing of application development means that security logic embedded in the application tier is no longer directly controlled by the enterprise. These changes in the regulatory

and development environments mandate a change in how access to the application tier is managed.

The next wave of application security technology is the management of user entitlements and to the separation of application security logic from the application tier. Oracle Entitlements Server is a product that provides centralized entitlements management with distributed enforcement of access to both application components and application business objects. This document focuses on how Oracle Entitlements Server can be used to solve your application entitlements problem.

What are Entitlements?

The word ‘entitlements’ can mean different things to different people. For the purposes of this paper we will define entitlements to be the set of privileges that govern what an application user can do. The ability to manage and enforce these permissions is also know as fine-grained authorization.

Figure 2-1 Typical Online Banking Application

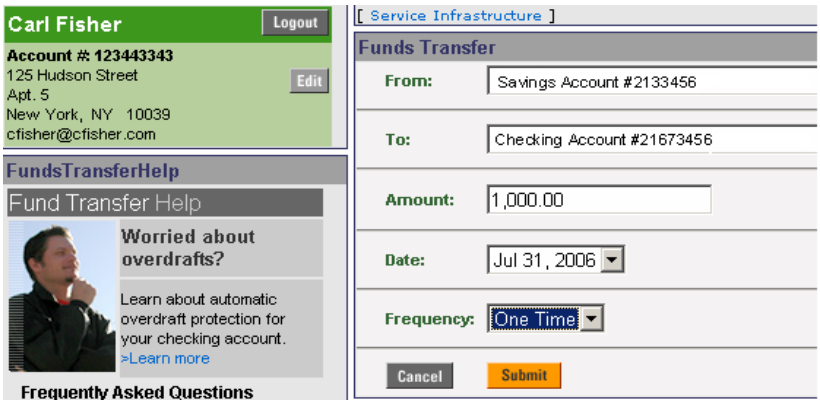


Figure 2-1 shows a typical scenario for an online, retail banking application. The application must provide certain capabilities to external users of the application as well as employees of the bank. Customers must be able to view their balances, change their profile information, and conduct certain transactions. Bank employees must be able to make balance adjustments to customers, open new accounts for customers and provide other customer services.

The bank's entitlements system must be flexible and able to answer questions like the following:

- Can and under what conditions may this customer transfer funds from this account?
- Which accounts can a user manage?
- Which actions can a customer delegate? To whom? To which limit?

The entitlements system is used to create and manage those actions and to make and record access decisions that are made at run time. The banking application may consist of a combination of software components (e.g. JSP's, EJB's, links) as well as logical objects such as customer accounts. The bank's entitlements system must be able to manage entitlements for both types of resources in the application. The bank also needs to be able to provide new capabilities to their customers that they can roll out quickly. As these new capabilities are rolled out, the user's entitlements will change also. The bank needs to do this without requiring recoding and retesting of the application.

Entitlements Challenges

There are a number of challenges associated with managing and enforcing entitlements in an application.

The first challenge is the presence of embedded decisions. These are authorization decisions that are hard-coded in the application code. Embedded decisions present several problems. The security logic may not be consistent across applications. Any change to the security logic requires changes to the application which must then be retested and redeployed. If these access decisions are in the application code then there is no central management or oversight over the security logic. There is also no way to analyze the access logic to determine who can do what. Finally, the decision may not be audited.

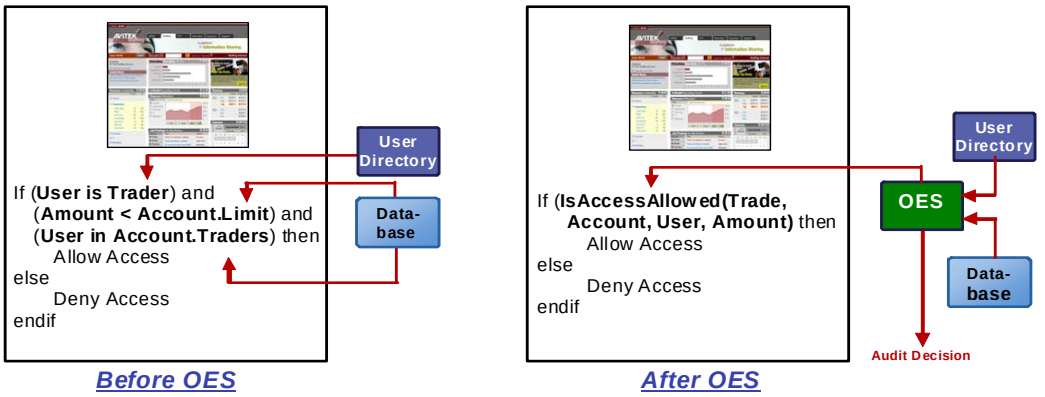
The second challenge is the presence of existing security technologies in the enterprise. Many companies have deployed centralized directories for storing user data, Web SSO systems, and user provisioning products. Any entitlements system must be able to integrate with these technologies to get user identities and user attributes required to make access decisions.

The last challenge is associated with the complexity of the entitlements model within an organization. Industries that are highly regulated may have to create complex rules to govern access to certain actions or data in an application. Furthermore, large enterprises may have complex hierarchies of roles, application resources or actions that can be performed. An Entitlements management system must allow the user to centrally define and manage application entitlements and enforce those entitlements across a variety of application development environments.

Getting Security Out of the Application

A major benefit of Oracle Entitlements Server is that it allows you to get security logic out of the application where it can be managed from a central point. Figure 2-2 shows a prototypical trading example. This application may be used by many different types of users. However, only those users who are traders are allowed to make trades in the application. Furthermore, traders are restricted to trade only for those client accounts for which they are authorized and only up to a specified limit set for each account.

Figure 2-2 Externalizing Security Decisions



The panel on the left shows the traditional way to control access in the application. In this case, the access control logic is embedded as code in the application. The application has to know how to go out to other systems in the infrastructure to get information about the user and information about the account. This makes the application brittle and difficult to change. It also means security logic is visible only by inspecting the code. Finally, the access decision may not be audited.

The panel on the right shows how the application would look using Oracle Entitlements Server to externalize the security decision. Instead of complex logic coded as part of the application, the developer uses an authorization call to get an access decision.

The authorization call (“Is Access Allowed”) is made with the following information:

- the requested action (“Trade”)

- the resource (the “Account” for which this trade will be made)
- the subject (who is requesting authorization to do this – the “User”)
- application context required to make the authorization decision (e.g. the “Amount” of the trade)

In this case, Oracle Entitlements Server is responsible for getting the required information about the user and the account. It gathers the required information, evaluates the policies that apply to accounts in the application and returns an access decision. It also audits all the information about that decision.

It is important to note that what is shown in Figure 2-2 is how a developer would make an explicit call for an authorization decision. For objects managed by WebLogic Server, this call is made for you automatically by the server. For J2EE objects like JSPs, EJBs, Web Services and for WebLogic Portal objects like books, pages, and desktops, the “IsAccessAllowed” call is made any time an application tries to access one of these resources. To control access to these “container-managed” objects, the application does not need to be changed.

Entitlements System Requirements

Any technology or product that is intended to solve the type of entitlements problem described above must be able to do certain things.

First, an entitlements solution must have a rich model for modeling access control. In particular it must be able to define and manage hierarchies of user roles, permission, and application resources. It must support an entitlements model that is rich enough to handle complex sets of conditions under which access will be granted (or denied). It must be able to enforce entitlements on both software components and business objects. Finally it must have the ability to represent and implement a variety of access control paradigms from role-based access control (RBAC) to data driven approaches based on user and resource attributes.

Next, an entitlements solution has to be easy to administer. Central management of entitlements is a key feature. However, while access should be administered centrally, those policies should be distributed to a set of policy decision points (PDPs) which are close to the application. Since business conditions are constantly changing, business users should be able to manage entitlements for their applications. The entitlements solution must provide the ability to approve and version security policy.

Finally, an entitlement solution should be easy to deploy and integrate with other systems in the corporate infrastructure. It should be flexible enough to allow the user to choose between a centralized policy decision point and a fully distributed set of policy decision points (PDPs). It

should be possible to make calls to the PDP from a variety of applications environments through some language neutral approach (e.g. like Web Services). It should be easy to integrate the entitlements solution with security infrastructure that is already present in the enterprise including corporate user stores, Web SSO solutions, and user provisioning.

CHAPTER 3

What is Oracle Entitlements Server?

Overview

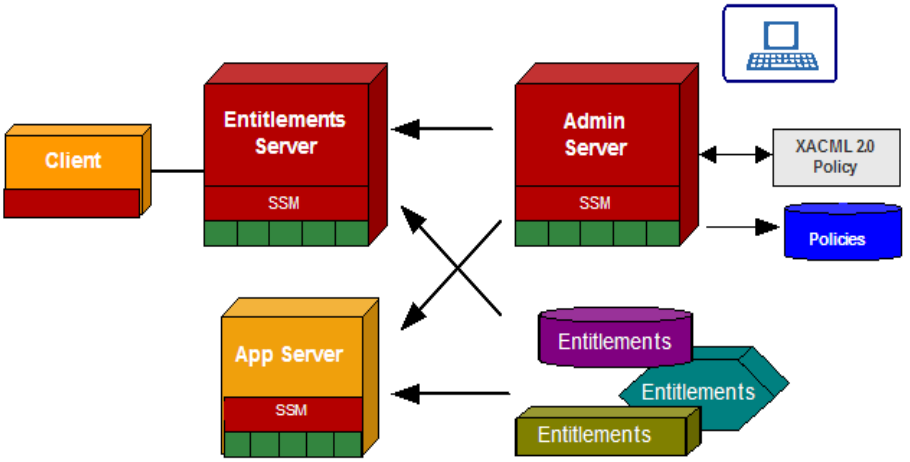
Oracle Entitlements Server is an entitlements system that supports the centralized definition of complex application entitlements and the distributed runtime enforcement of those entitlements. It allows you to externalize entitlements – remove security decisions from the application. It provides the means to define application resources and application businesses objects, represent those objects in hierarchical relationships, and write policies that describes which users, groups and roles can access those objects.

You can write policies that control access to both application software components as well as arbitrary business objects in the application. Oracle Entitlements Server also includes a security integration framework that provides an easy way to integrate with existing authentication, Web SSO, identity management, and user provisioning systems.

Architecture

As shown in Figure 3-1, the Oracle Entitlements Server architecture is made up of two major components – the Administration Server and the Security Service Modules (SSMs).

Figure 3-1 Architecture



The Administration Server is the Policy Administration Point (PAP). It manages the storage of policy data in the database and the transactional distribution of policies to the SSMs. The user interacts with the Administration Server through two browser-based administration console where roles, policies, and application resources are defined and managed. The information can also be managed programmatically through the Java management APIs or through Web Service calls.

Runtime enforcement in the application container is accomplished through a set of Security Service Modules. The SSMs are the Policy Decision Points (PDPs) and can be deployed in one of two ways:

- As a centralized entitlements server that can be invoked via Web Services, RMI, or through the XACML 2.0 request/response protocol.
- As a distributed set of PDPs which are embedded in the application container. In the distributed deployment model, policy is evaluated and enforced locally in the application container.

The SSMs can make use of application context that may be required to make a policy decision. Additionally, the SSMs can integrate with a number of Policy Information Points (PIPs). These PIPs can be user or application directories or databases that contain information that is required to make an access decision. Such information includes user, group, and resource attributes (e.g.

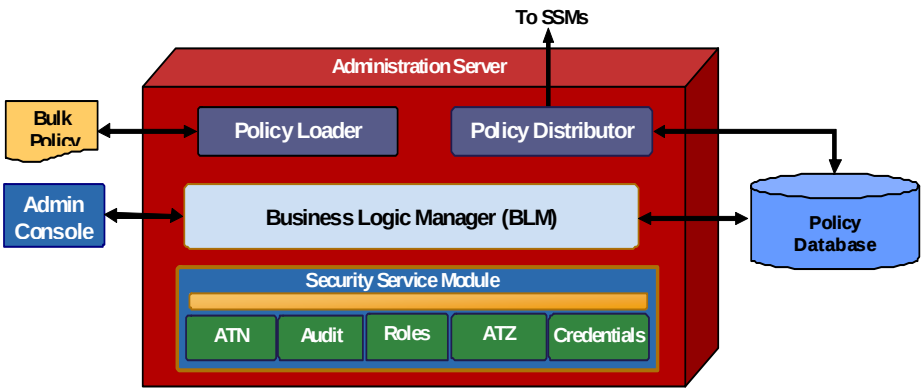
user profile information, account balances and limits, etc.). These attributes can then be used in the policies which control access.

Administration Server (PAP)

Figure 3-2 shows the components of the Administration Server. It provides browser-based consoles where an administrator defines users, groups, roles, the application resources, and the policies that control access to those resources. The Administration Server can be used to delegate administrative functions to other users. It also provides a policy simulation function that allows you to identify the policies that apply to given resources, users, groups, and roles.

The Administration Server is made up of several components. The Business Logic Manager (BLM) interacts with the policy database to control persistence of policy and other data. The BLM supports transactional management operations for creating and modifying users, attributes, roles, and policies. The Policy Distributor manages the process of distributing policy to the Security Service Modules for runtime enforcement. Policy distribution to the SSMs is incremental and transactional. The Policy Loader manages import and export of policy data. Policy can be exported in XACML 2.0 format.

Figure 3-2 Administration Server



Java and Web Services administration APIs are provided so that you can incorporate security services functionality into your own applications to automate administration. The administrative APIs can also be used to provide integration with user provisioning systems. For example with a provisioning connector, the attributes of a new employee can be pushed to Oracle Entitlements

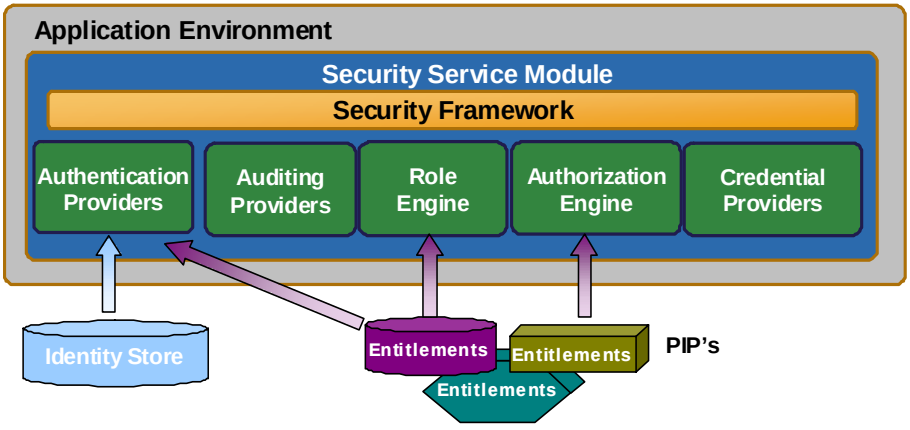
Server. Those attributes will then provide that employee with immediate access to the required set of applications.

Security Service Module (PEP, PDP)

A schematic of an SSM is shown in Figure 3-3. The SSMs are the Policy Decisions Points. SSMs can be fully distributed in the container where the application is running or deployed as a central entitlements server. In the distributed deployment model, they plug directly into the application container to intercept requests and enforce access policies. Applications can invoke the security services directly through either the Java or Web Services APIs. All policy decisions are made by the SSM. SSMs get updated policy information through an incremental and transactional policy distribution mechanism that keeps policy sets up to date and consistent between SSMs.

The entitlements server is also an SSM but one which is deployed on a central server rather than distributed to individual machines hosting the applications. In this case, the centralized entitlements server gets an authorization request from the application either as a Web Services call or using the XACML 2.0 request/response protocol. Like the distributed SSMs, the centralized server can integrate with various Policy Information Points to retrieve and cache user or application information used to make policy decisions.

Figure 3-3 Security Service Module (SSM)



The SSM contains a security framework that provides a set of standard security services including authentication, authorization, role mapping, auditing and credential mapping. Through the integration with various data repositories (PIPs), the SSMs can retrieve data required to make an access decision. Those data may include static user information which is retrieved when the user

is authenticated or dynamic entitlements data that is retrieved at the time a policy is being evaluated. The SSM maintains a fully configurable cache to maintain data required for policy evaluation to minimize attribute retrieval calls to the PIP(s).

Managing and Enforcing Entitlements

Managing and enforcing entitlements generally involves three groups of people. Security architects define security policies that will determine a user's entitlements in the application. Typically the security architect works with the developer to determine which application resources need to be protected. They work together to define role policies and access policies to be enforced at runtime.

Business users manage the data that controls how the policies are evaluated at runtime. They add and delete users for the applications and put those users into groups or assign them to roles. They manage sets of actions (permissions) that can be logically grouped for a particular business function. They assign those sets of actions to users or to roles defined for the application.

Developers building new applications (or retrofitting existing applications) integrate Oracle Entitlements Server with those applications. This can be done in a variety of ways but the objective is to get the security logic out of the application where it can be managed in a single place. This is accomplished by making an authorization call to Oracle Entitlements Server which will then evaluate access control policy and return an access decision. Since a set of security services are also provided, the developer can use Oracle Entitlements Server as the integration platform for other systems like Authentication, Web SSO, and Identity Management.

Managing Policy

Policy is written and managed by administrators and security architects. The policy model is very rich and powerful and can be used to implement a variety of access control paradigms, including rule-based, role-based, and label-based access control.

A label-based model lets a policy authorize access when attributes in a resource match attributes in a user or group. For example, "Allow access to customer records only when a customer's region code matches a call center representative's assigned region code." This type of model is very useful when there are descriptive attributes for both resources and users that can be used to develop policies.

With a rules-based model, policies are written as conditional statements using information gathered at runtime. For example, "Branch Managers are permitted access to monthly revenue reports if their organizational level is greater than 5 and senior management has approved the

report". A rules-based model typically leverages a hierarchical set of resources (e.g., monthly revenue reports) and the conditional expression can be of arbitrary complexity.

With a roles based model, policies are assigned to application roles. Users and Groups are then assigned to Roles, thereby gaining access to various resources. The benefit of this approach is that the concept of a Role decouples an individual or group from the actual policies that determine their access rights. This reduces maintenance costs considerably since an individual can change access to resources by simply moving to a new role. It also aligns better with real world use cases since many individuals work in multiple roles, often simultaneously.

The Policy Model

The general form of a policy is as follows:

Effect (Role | Action, Resource, Subject) Constraints

This policy format can be read as "Grant, deny, or delegate an action for a given resource to a subject under some set of constraints" where:

- *Effect* can be grant, deny, or delegate.
- *Role* is a role to be assigned to users or groups.
- Action is a privilege on the resource. Actions can be standard actions for software components (e.g. URL: Get, Post) or custom actions for business objects (e.g. Bank Account: Transfer)
- *Resource* is a part of the application to be protected. Resources can be components managed by the container (e.g. URLs, EJBs, etc.) or arbitrary business objects in the application. Resources are stored in a hierarchy and may have attributes. For example, bank accounts have owners, creation dates, and transfer limits. Policies that are attached to resources in the hierarchy are inherited by any child resources.
- *Subject* is who the action and resource is granted (or denied) to. Subjects can be users, groups or roles.
- *Constraints* are other conditions that must be true for the policy to evaluate to true. Constraints can be complex combinations of Boolean expressions that test the value of some user, resource, or system attribute. Custom Java evaluation functions can be used in constraints to evaluate complex business logic.

Role Policy

Role policy is used to dynamically determine role membership. Role policies are always scoped to a resource or set of resources. Consider the following example role policy:

Grant (//role/BankManagers, //app/AccountReports, everyone) if (JobTitle=BankManager)

In this example, the BankManager role will be granted to everyone for the AccountReports resource in the application, but only if the user's job title is Bank Manager. Here a user attribute is used in a constraint to further control the conditions under which the role will be granted.

It is also possible to use role policy to affect separation of duties. The second example shows a simple separate of duties policy. This policy will deny the Analyst role to anyone who has the Trader role in the Brokerage application.

Deny (//role/Analyst, //app/Brokerage, //role/Trader)

The last example shows how a policy can be used to delegate a role to someone else. In this case John is going on vacation. His Approver role in the AccountsPayable application will be delegated to the people in his group (the group called JGroup) during his vacation (August 1 – August 10).

Delegate(//role/Approver, //app/AcctsPayable, //grp/JReports, John)
if (date > 08/01/06) and (date < 08/10/06)

Access Policy

Access policy is used to grant or deny actions to resources in the application to specific users, groups, or roles. Access policy is used to control who is allowed to perform specific actions on an application resource. Consider the following access policy.

Grant (view, //app/Reports, //group/UnitManagers) if reportBusUnit=userBusUnit

In this example, the policy grants the view action on the application resource called Reports to everyone in the UnitManagers group, but only if that user belongs to the same business unit as the report. Here the constraint uses an attribute of the user (BusinessUnit) as well as an attribute of the resource (BusinessUnit).

Access policies can operate on logical resources as well as software resources. A custom resource type can have custom actions associated with it. The next example shows how a custom action called transfer for a bank account might be controlled. The action is granted for a particular account if two conditions are true. The user must be in the list of account owners and the requested transfer amount must be less than the limit set for the account. In this case, the list of account owners and the transfer limit are both attributes of the account (the resource).

Grant (transfer, //app/account, everyone) if user IN accountOwners AND transferRequest <= transferLimit

The last example shows an additional feature of the policy engine. Normally, the policy engine evaluates a set of policies for a particular access decision and returns an authorization decision (grant or deny). Using the Report function, the policy engine will return additional data with the authorization decision. In this case the value of the *transferlimit* attribute is returned to the application. The application can then use that data for additional processing.

Grant (transfer, //app/account, everyone) if user IN accountOwners AND (Report(transferLimit))

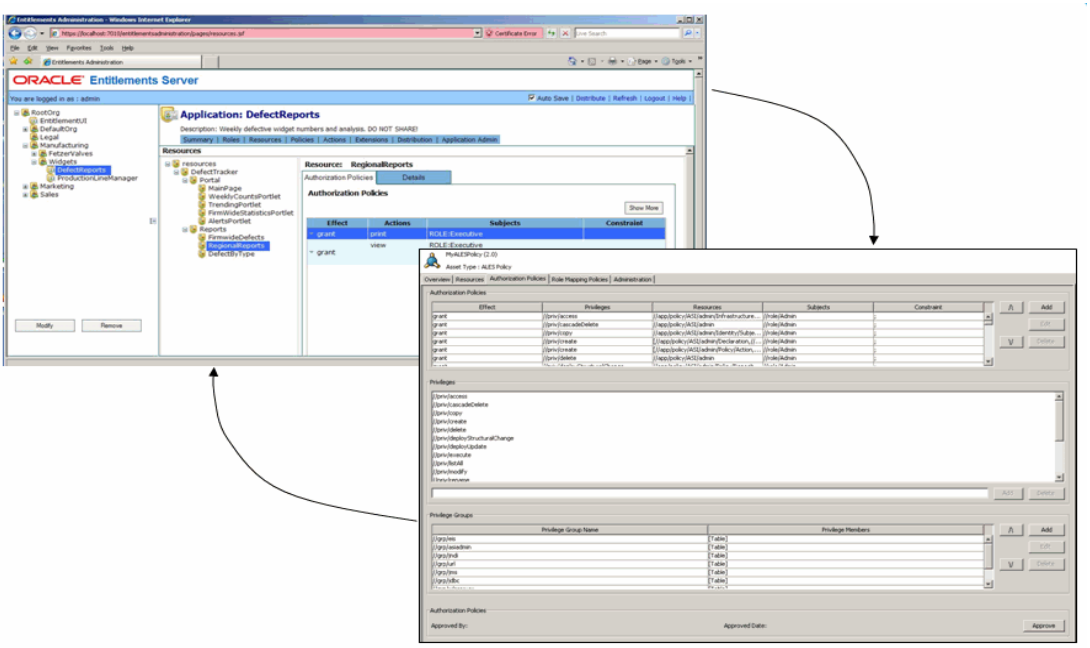
Policy Versioning and Workflow

Oracle Enterprise Repository manages the metadata for any type of software asset, from business processes and Web services to applications and components. It maps the relationships and interdependencies that connect these assets to improve impact analysis, promote and optimize reuse, and measure their impact on the bottom line.

Figure 3-4 shows a typical example of Oracle Enterprise Repository and Oracle Entitlements Server working together. When used with Oracle Entitlements Server, Oracle Enterprise Repository can provide policy versioning and workflow for approval and notification of policy updates. A security administrator defines roles, policies, attributes and other data required to enforce a set of entitlements for an application (or set of applications). At any point, the administrator can store that information as an asset in Oracle Enterprise Repository. If the name chosen is the same as the name of an existing policy set in Oracle Enterprise Repository, a new version number is created. Workflow defined in Oracle Enterprise Repository will be enforced whenever a new policy set is created or a new version of an existing policy set is created. That workflow may involve notifications and/or approvals before the new asset (or version) is registered. The administrator can also pull an existing policy set from the repository into the Entitlements Administration Application to be inspected or modified.

A typical use case involves the migration between staging environments. Let's take the example of going from the development environment to the QA environment. The security administrator exports policy information to Oracle Enterprise Repository. The policy asset in Oracle Enterprise Repository requires approval by a project manager before it is registered. Once it is approved, Oracle Enterprise Repository can issue a set of notifications. The security administrator must then import the policy into Oracle Entitlements Server in the QA environment. That import also requires approval which can then trigger a second set of notifications.

Figure 3-4 Versioning Policy with Oracle Enterprise Repository

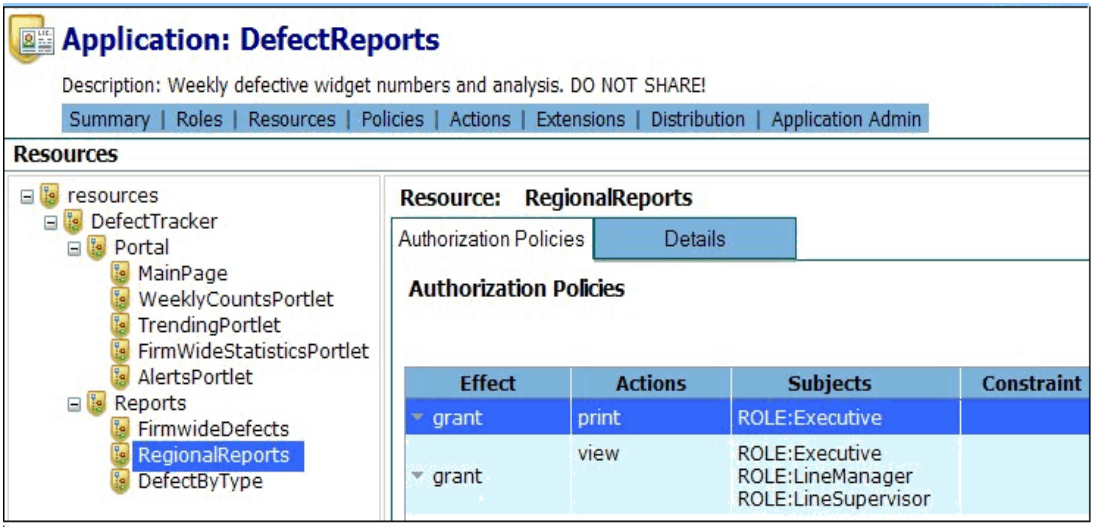


Managing Entitlements Data

Within the Entitlements Administration Application, administrators define the components of policy definitions and marshall these components into the policies that govern user entitlements. Administrators may manage entitlements data by organization and application.

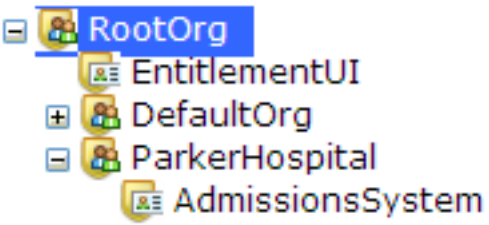
It is important to note that the Entitlements Administration Application allows for the shared management of entitlements among security specialists, technical administrators, and business managers. For example, a security specialist might devise a policy model that provides role-based access to an application and then turn over to a business manager the job of assigning users to the application's role. This would shield business administrators from the complexity of writing policy and allow them to concentrate on role assignments pursuant to business needs of the organization.

Figure 3-5 Managing Policies for an Application



In previous releases, users of the Entitlements Administration Applicationn maintained all policies in one flat structure. In this release, ‘organization’ and ‘application’ nodes have been added so that policies can be organized and managed under separate applications. The organization node contains all users and groups as well as one or more applications, where the policies securing one application can be managed separately from another. This not only constitutes a useful user interface improvement, it also allows you to set up different application administrators with full rights to manage the policies securing a particular application, but with restricted rights (or no rights at all) on the policies securing other applications.

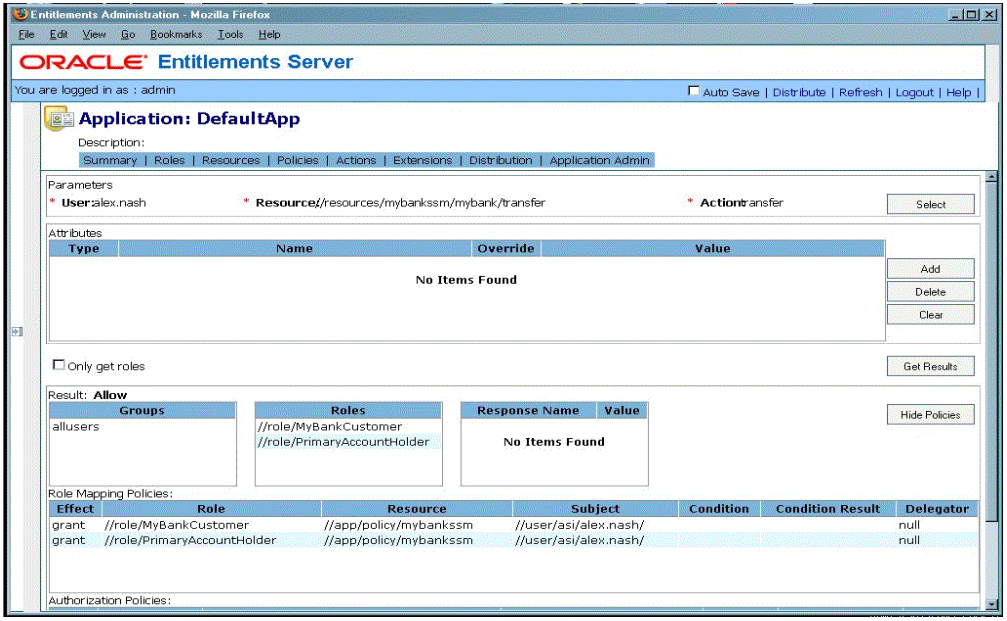
Figure 3-6 Organization/Application Tree



To assist in the development and ongoing analysis of policy, the Entitlements Administration Application provides policy reports function and a policy simulation module:

- **Policy Reports** can generate a variety of reports to analyze entitlements. These include what roles a user has, what users are in a role and for what resources the roles is granted.
- **Policy Simulation** allows an administrator to troubleshoot, test, and understand how policies are being enforced on a specific application. For policies that contain constraints, "what if" scenarios can be run by substituting different attribute values for the policy to evaluate.

Figure 3-7 Running Policy Simulation in the Entitlements Administration Application



CHAPTER 4

Integration and Deployment

The chapter describes methods of integrating Oracle Entitlements Server with application environments under different deployment models.

Integration with Applications

Oracle Entitlements Server supports a number of integration points to interact with other applications and environments in your enterprise. The runtime API is the interface to the SSMs and can be invoked via Java or Web Services calls. Java clients have the option of calling the SSM which is embedded directly inside their runtime or making remote calls to the RMI or Web Services SSM via a proxy. The choice can be made outside of their application via configuration and Java clients can switch protocols without any recoding."

The management APIs can be used to create users, groups, and roles, define access or role-mapping policies, or distribute policies to the SSMs. The management APIs can be invoked either through Java or Web Services calls. Finally, Oracle Entitlements Server supports a wide variety of directories and relational databases for either authenticating users or retrieving user attribute information.

Out-of-Box Security Service Modules

Out-of-box SSMs plug directly into the application container and automatically intercept access requests and enforce access decisions with no changes to the application. These out-of-box SSMs are available for WebLogic Platform products and Web Servers. They also provide integration with the Oracle Enterprise Repository product for policy versioning and workflow.

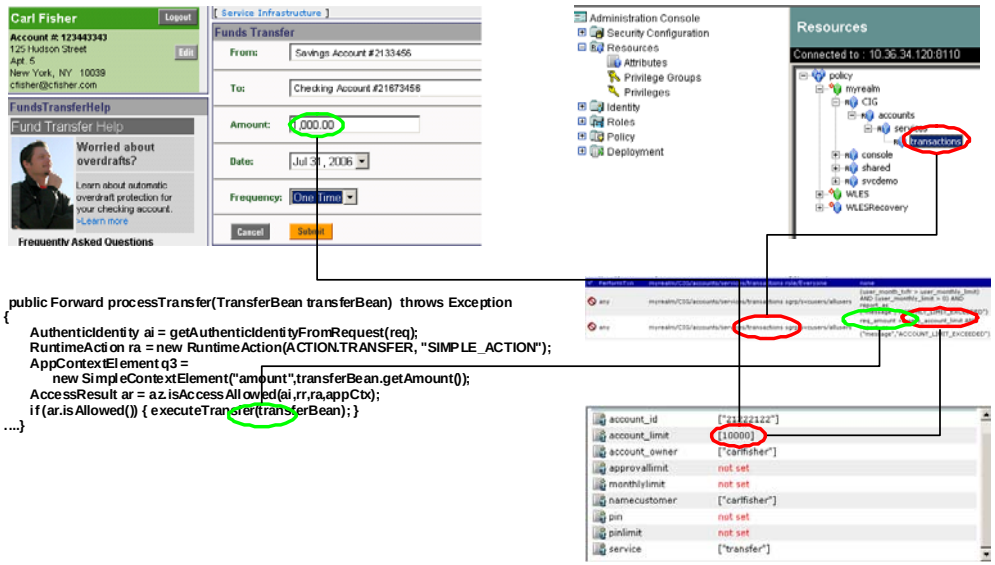
The following out-of-box SSMs are provided in Oracle Entitlements Server 10g Release 1.0.4.3:

- WLS SSM (for WebLogic 9.x/10.x)
- WLS 8.1 SSM (for WebLogic 8.1)
- Web Server SSM (for Microsoft IIS and Apache Web Server)
- Web Service SSM
- WebSphere SSM
- Oracle SSM
- RMI SSM
- SharePoint (MOSS) SSM

Using the Java API

For custom application objects, the developer creates explicit calls to Oracle Entitlements Server in the application code in place of application-specific access control logic. Figure 4-1 illustrates how this is typically done.

Figure 4-1 Application Integration using the Java API



Let's look at this example in some detail. At the top left we have the application – the user is trying to transfer $1000 from one account to another. The application needs to check whether the user is authorized to make that transfer.

In the top right corner we have depicted the administration console and the policy associated with the transaction. In this case the policy is written to Deny access to the transaction resources if the request amount is greater than the limit imposed by the bank on that account. Note that the policy decision is completely data driven – the transfer will be granted or denied based on comparing the value of a user attribute to the account limit. On the bottom left, we see the Java that makes a call to the Client API. The call is straight forward – the user is trying to access a resource (transaction) within a specific context (amount). Finally on the bottom right side – the users account limit will be compared to the requested transfer amount. In this example the user's attribute values are stored in the entitlements system. However, that data could be stored in any external system (e.g. RDBMS, LDAP, or Web Service).

Note also that the Java application could be making calls to the Java SSM embedded within its runtime or remote calls to the Web Service or RMI SSMs using the same API.

For clarity we have shown how an entitlements system might be integrated programmatically with the application. It is also possible to encapsulate the client integration (depending on programming model) to minimize the amount of integration code and to standardize the integration with the application.

Using JSP Tags

For presentation elements, a tag library is provided for JSP developers. The tag library allows you to protect access to page level elements like buttons, menu items, tabs, links, or any other data on the page. In the example shown in Figure 4-2, the JSP developer wants to show the private information only to those authorized to see it. The IsAccessAllowed tag in the JSP code initiates a call to the policy engine. The grant policy shown below applies to the resource called */examplewebapp/private* as identified by the tag. All users in the group called Private will be granted access and will see the information in the private table. All others will be denied and the private data table will not be displayed.

Figure 4-2 Using the Tag Library

Annotations in WebLogic Workshop

Oracle Entitlements Server also provides users of WebLogic Workshop a way to annotate java objects to indicate that they are to be protected. The annotation indicates the name of the resource, the action on that resource, and any security related attributes of the resource. The Workshop project can be inspected and its resources, the action, and any resource attributes can be imported into OES. Then, from the Entitlements Administration Application, the security administrator can create the desired role or access policies for that resource. This allows the developer and security architect to work together. In this model, the developer indicates what needs to be protected and the security architect defines how that object will be protected.

Security Providers

While developers and third parties may use the Security Service Provider Interfaces (SSPIs) to develop custom security providers, the product comes with a range of out-of-box providers that provide authentication, authorization, identity assertion, credential mapping, and auditing with leading vendors.

OES provides the following out-of-box attribute retrievers that are used by ASI Authorization and ASI Role Mapping providers to retrieve attributes for use in policies:

— RDBMSAttribute Retriever
— LDAPAttribute Retriever
— SDOAttribute Retriever
— ALESIdentity Attribute Retriever
— Custom Attribute Retriever

If the out-of-box attribute retrievers do not meet your needs, you may develop your own and manage it from the console.

Deployment in the Enterprise

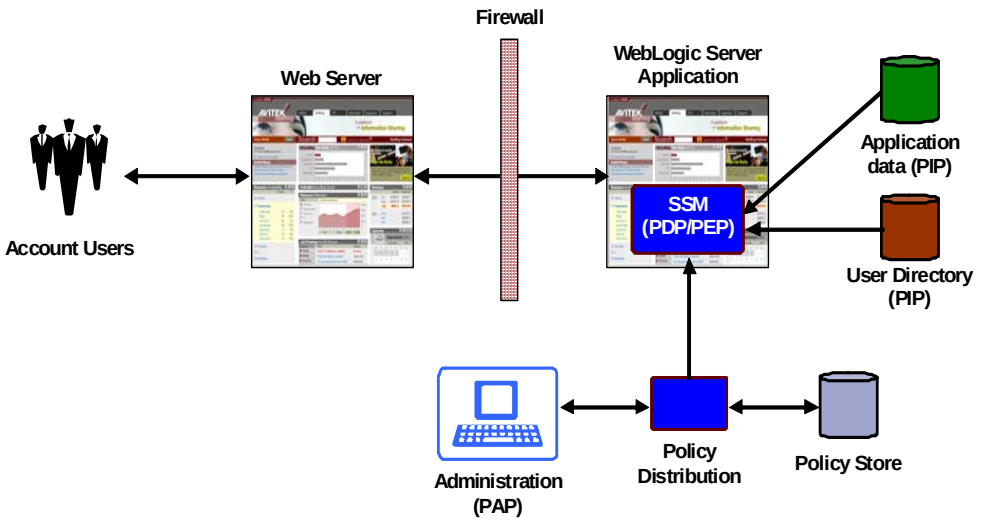
Two main deployment models are supported. The two models differ in the placement of the Policy Decision Point (PDP). The SSM serves as the PDP. In the distributed model the PDP is embedded in the application container and all security services are fully distributed. In the centralized model, client applications make calls to a central PDP via RMI or SOAP calls. The central PDP makes the access decision (by evaluating policy) and returns an authorization result to the client. These two models are not mutually exclusive. Any combination of the two is possible.

Distributed Deployment Model

A typical distributed deployment is shown in Figure 4-3. In the distributed model, SSMs plug into the application container. Out-of-box SSMs are provided for WebLogic products, popular Web Servers, and for Java. In the distributed model, the SSM acts as both the Policy Decision Point and the Policy Enforcement Point.

The WebLogic SSM protects J2EE objects in WebLogic Server such as JSPs, EJBs, Web Services and POJO's. It also protects portal objects in WebLogic Portal such as books, pages, portlets, and desktops. The Java SSM exposes runtime services through a Java API and can be used to secure applications in other Java environments.

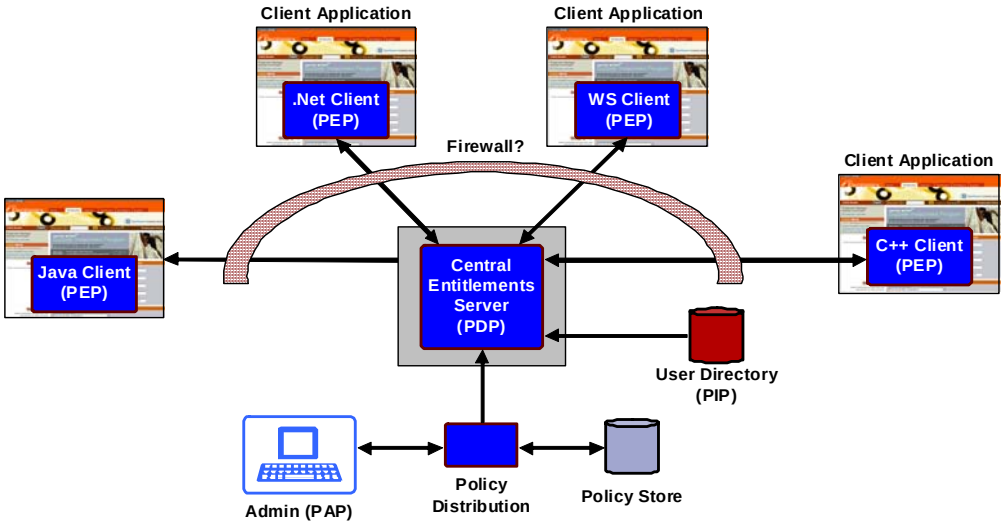
Figure 4-3 Distributed Deployment



Centralized Deployment Model

Figure 4-4 shows a typical centralized deployment. Here the SSM is hosted on a dedicated server and receives authorization requests from multiple clients. On the clients, SSM proxies provide local security services, including authorization caching, and communicate with the central SSM using RMI or SOAP as the transport protocol.

Figure 4-4 Centralized Deployment



Scalability

Oracle Entitlements Server was designed to be highly scalable. This is accomplished in a number of ways. First, the policy decision points (the SSMs) can be fully distributed. Policies are pushed to the SSMs from the Administration Server. From that point forward, all decisions are made and cached locally at the SSM (no network calls to a central server). Second, the policy model supports and encourages the use of a small number of policies that can make use of a rich set of user entitlement information. This minimizes the set of policies to be managed. Using a small set of data driven policies also aids in compliance. Finally, Oracle Entitlements Server has a scalable model for entitlement retrieval. Static entitlements are retrieved once (at authentication time). Dynamic attributes are retrieved at authorization time and cached during the user's session. The cache is fully configurable by the user. You can also write custom attribute retrievers that can implement a caching scheme specific for your application environment.

CHAPTER 5

Summary

Application security has evolved from perimeter security to basic access control and single sign-on at the Web tier. Today the need to provide fine-grained control and audit in enterprise applications is pushing security deeper in the application stack.

Currently, enterprises face three challenges

- embedded entitlements decisions in the applications,
- integration with existing security solutions, and
- the complexity of the entitlements model.

Oracle Entitlements Server is an enterprise security product that provides two important benefits. First, it is a fine-grained entitlements product that provides a means to centrally define and manage policy to control access to applications. Second, it is a security service platform to provide security services to applications across multiple application environments.

Oracle Entitlements Server provides flexibility in its deployment. It can be used in a container based deployment with plug-in Security Service Modules for WebLogic products, Web Servers, and Java applications. It can be used in a services based deployment with client applications making SOAP calls to a centrally located Web Service SSM for security services.

Oracle Entitlements Server is an entitlements systems that provide the means to define application resources and application businesses objects, represent those objects in hierarchical relationships, and write policy that describes which users, groups and roles can access those objects.

Oracle Entitlements Server allows you to externalize entitlements – remove security decisions from the application. You can write policies that control access to both application software components as well as arbitrary business objects in the application. The Oracle Entitlements Server supports a hierarchical model of resources and policies with inheritance. Oracle Entitlements Server supports two types of policies -- for defining roles and for controlling access to resources in the application.

Oracle Entitlements Server is also a security services platform that provides five basic services for authentication, authorization, role mapping, auditing, and credential mapping. This service infrastructure is based on the WebLogic security framework and supports the same Security Service Provider Interfaces (SSPIs). User can add there own providers to replace or work in conjunction with the providers that ship with Oracle Entitlements Server.