VPD Agent

Overview of VPD

This section briefly describes the Oracle VPD and how CEPM VPD Agent helps to implement it.

What is Oracle VPD?

Oracle VPD is the aggregation of server-enforced fine-grained access control and a secure application context in the Oracle database server. VPD limits access to the data present in the database, at the row level, and ties the security policy to a table, view, or synonym. This linking is done by dynamically appending predicates (WHERE clauses) to SQL statements that query the data that you want to protect.

How Does Oracle VPD Work?

When an application user accesses a table (or view or synonym) that is protected by a VPD policy, the Oracle server invokes the policy function. The policy function returns a predicate (some text) based on session attributes or database contents. The database server dynamically rewrites the submitted query by appending the returned predicate as the condition in the WHERE clause. The modified SQL query is then executed.

You can restrict a user to view specific records in a particular table with query criteria that is specified in VPD, and whenever the user references that table, the user is limited to view the records as per the VPD restrictions. This kind of security applies not only for a basic select query, but also for update, insert, and delete statements. Thus you can restrict the user only to view, update, or delete the records that the user owns and allow to insert records based on the permissions given to the user.

For example, consider the Order table with following data:

ORDER_ID	ORDER_DATE	CUSTOMER_ID	ORDER_STATUS	ORDER_TOTAL
2458	17-AUG-99 03.04.12.234359 AM	101	0	78279.6
2397	20-NOV-99 04.11.54.696211 AM	102	1	42283.2
2454	03-OCT-99 05.19.34.678340 AM	103	1	6653.4
2354	15-JUL-00 05.48.23.234567 AM	104	0	46257.0
2358	09-JAN-00 06.33.12.654278 AM	105	2	7826.0

User User1 can be restricted to view records based on order_id or order_date or any other column in the Order table.

Here are some examples of WHERE clauses for implementing this kind of restriction:

- WHERE order_id >2358,
- WHERE order_id >2354 and order_date < '15-JUL-00 10.52.35.564789 AM',
- WHERE customer_id not in (101,104,105)

If User1 is allowed to access records where order_id>2358, and given permissions for insert, update, and delete on the Order table, then it is possible only to update, delete, and insert records where order_id>2358.

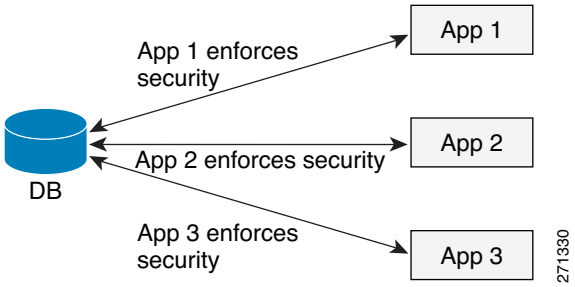
If User1 tries to join the order table with any other table, say customer table, then only User1 records (that is records having order_id>2358) can be joined with the records of customer table. This way User1 cannot see some records of customer table also.

If User1 tries to execute a stored procedure that uses the order table, then the operations performed by the procedure on the order table are restricted to only User1 records (records having order_id>2358).

What Are the Benefits of Using Oracle VPD?

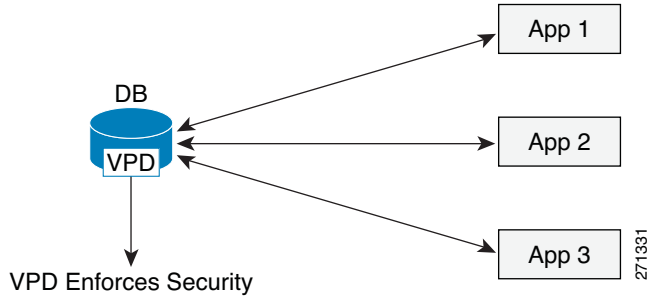
Scenario 1: Data Security Without Using Oracle VPD

When multiple applications such as App1, App2, and App3, you should replicate the data security logic in all the applications. Thus if you make any change in the security configuration, then you should replicate for each individual application.



Scenario 2: Data Security Using Oracle VPD

When the VPD is implemented in the database, then the database itself controls the security policies and all the applications using the data has the security policies automatically applied. Thus any change in the security configuration needs changes only at the database side, and no change is needed in the applications accessing the data.



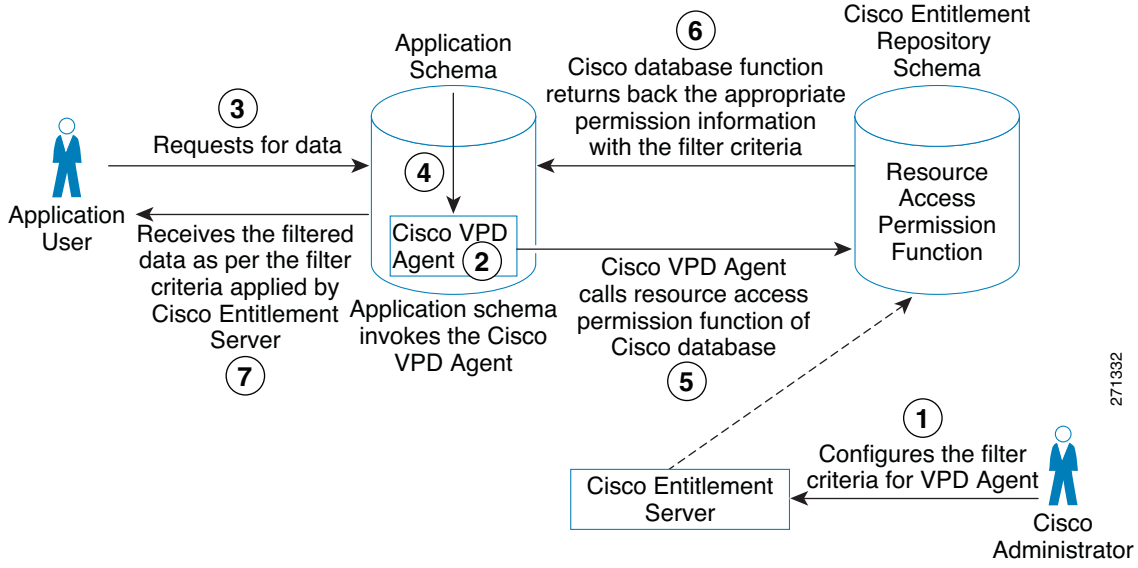
Here are more reasons why using VPD is beneficial for implementing security policies:

- Scalability: Consider a database table named Customers that contains 1,000 customer records. Suppose we want customers to access only their own records. Using views, we must create 1,000 views. Using VPD, this restriction can be done by creating a single policy function.
- Simplicity: Perhaps we have a table T, and many views are based on T. Suppose we want to restrict access to some information in table T. Without VPD, all view definitions have to be changed. Using VPD, the restriction can be done by creating one policy function and attaching it to table T. Thus, as the policy is enforced for table T, the policy also gets enforced for all the views that are based on table T.
- Security: For VPD, security is database server-enforced, as opposed to application-enforced.

How CEPM Helps to Implement Oracle VPD

CEPM provides a software component called VPD Agent that integrates with Oracle Database in such a way that the WHERE clause predicate is configured in the Policy Decision Point (PDP) for the required policy. Then this predicate is dynamically accessed by the VPD Agent deployed in the Oracle Database, which implements row-level security for the appropriate tables present in the Oracle Database.

PDP allows you to logically define the security scheme used in your application for integrating with Oracle database schema.



The VPD Agent mechanism works as follows.

VPD Agent Configuration Steps

Step 1 Configure security policies in the Policy Administration Point (PAP). This step involves creating appropriate policies and setting the policy attributes (predicates) for those policies, so as to provide the restricted access on a particular database table, view, or synonym for the required application users. (Refer to "Sample Use Case" for details about how the VPD Agent is configured in the PAP).

Step 2 Configure the VPD component in the application schema that needs to be protected. This step involves creating and compiling the policy function, and then executing Oracle's in-built package function, DBMS_RLS.ADD_POLICY(), at the SQL prompt with appropriate function parameters. (Refer to "Sample Use Case" for details about how the VPD Agent is configured in the application schema).

VPD Agent Execution Steps

Step 3 Application user requests data from the application database.

Step 4 Application database invokes the VPD Agent that was configured in Step 2.

Step 5 As per the configuration, the policy function of the VPD Agent calls the resource access permission function (the isUserAccessAllowed()) that is present in CEPM entitlement repository schema. Two parameter values, User Name and Resource Name, are passed to this function.

Step 6 The isUserAccessAllowed() function executes and returns the access permission information and the policy attribute (also called as predicate) enforces the filter criteria on the requested query, to the policy function of the VPD Agent.

Step 7 The policy function generates a dynamic SQL statement by appending the attribute (that is the predicate) to the original requested query, executes the new query, and returns the filtered data to the requesting user.

Note Refer to "Sample Use Case" for details about how the VPD Agent works.

Installing the VPD Agent

This section describes the installation process for the VPD Agent.

Prerequisites

- CEPM schema must be installed and the database instance must be running.
- Application schema installed and the database instance running.

Application Schema Configurations for VPD

You must configure the following database privileges for the application schema for the VPD Agent:

- CREATE SESSION
- ALTER SESSION
- UNLIMITED TABLESPACE
- CREATE TABLE
- CREATE SYNONYM
- CREATE PROCEDURE
- EXECUTE ANY PROCEDURE
- CREATE TYPE

- CREATE ANY TYPE
- CREATE ANY CONTEXT
- EXECUTE ON DBMS_RLS

You must also provide the following permissions to create and drop database links to the application schema user, if CEPM schema and the application schema are installed on different machines.

- CREATE PUBLIC DATABASE LINK
- DROP PUBLIC DATABASE LINK

Executing Database Scripts for VPD Installation

VPD Agent installation requires executing a few database scripts into the application schema. To install the VPD Agent, follow these steps:

Step 1 Navigate to the following folder in CEPM server installation directory:
\CEPM-v33\db\scripts\oracle\Vpd\
This folder contains the database scripts (.sql files) for installing the VPD Agent.

Step 2 Execute the following first two scripts to install the VPD Agent.

CEPM_Vpd_Configure_3.3.sql

This database script that configures the tables of the application schema for VPD implementation. Execute this script from the sql prompt and when you are prompted to enter the comma-separated list of table names that exist in the application schema and need to be protected.

Note The inputs should always be given in single quotes.

For example, to configure two tables ‘PORTFOLIO' and 'ACCOUNTS' present in the application schema, for VPD implementation, enter the value at the sql prompt as: 'PORTFOLIO, ACCOUNTS’.

CEPM_Vpd_Package_3.3.sql

This database script contains the policy function that calls a database function present in CEPM Server schema that returns the policy information configured as part of the VPD implementation. Open this script in an editor and replace 'CURRENTSCHEMA' text with the name of the application schema, and replace 'CEPMUSER' text with the database schema name of CEPM Server database.

If CEPM Server database and Application database are running on different machines, then replace CEPMUSER.CALL_SECVPD' with CALL_SECVPD'@<dblink>', where <dblink> is the database link created for CEPM Server database.

Use the following SQL commands to create the database link.

```
create public database link <database link name>

connect to <CEPM Repository schema name> identified by <schema password>

using '<database name of CEPM Repository schema>'
```

CEPM_Vpd_UnConfigure_3.3.sql

This database script is to remove the existing implementation of VPD created for the database tables present in the application schema.

When you execute this script from the sql prompt, you are prompted to enter the comma-separated values of table names for which you want to remove the existing implementation of VPD.

For example, to remove the existing implementation of VPD for two tables 'PORTFOLIO' and 'ACCOUNTS' present in the application schema, enter the value at the sql prompt as: 'PORTFOLIO, ACCOUNTS'.

Sample Use Case

This section describes how the integration happens between CEPM Server and the database of the application that needs to be protected.

PORTFOLIO_ID	PORTFOLIO_NAME	PORTFOLIO_MANAGER_NAME	ASSOCIATE_NAME
1	Portfolio1	Tom	Tom
2	Portfolio2	Tom	Mary
3	Portfolio3	Tom	Mary
4	Portfolio4	Tom	Mary
5	Portfolio5	Tom	James
6	Portfolio6	Tom	James
7	Portfolio7	Tom	James

Consider a database table named Portfolio that exists in the application database. The Portfolio table stores the portfolios of various users. Three application users – Tom, Mary and James – Tom being the portfolio manager and Mary and James being the associate users. Let the data present in the Portfolio table be as shown below.

Let the business logic for access control of this table for the three users be as explained below.

If the portfolio manager accesses the data from this table, he should be able to view his as well as his associate's portfolio information, but if an associate accesses the data from this table, he should be able to view only his information.

For example:

When Tom logs in and types **select *** for the Portfolio table, then his query should be:

```
select * from Portfolio where portfolio_manager_name = Tom
```

When Mary logs in and types **select *** for the Portfolio table, then her query should be:

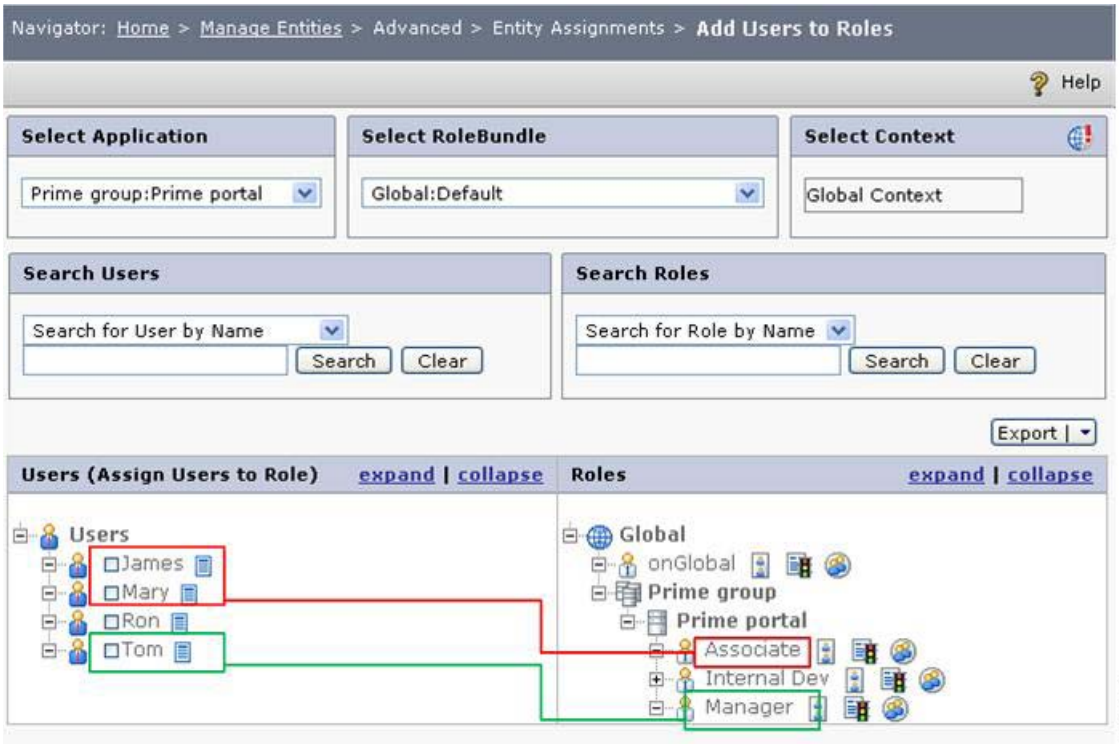
```
select * from Portfolio where associate_name = Mary
```

Configuring the VPD Agent Component in CEPM

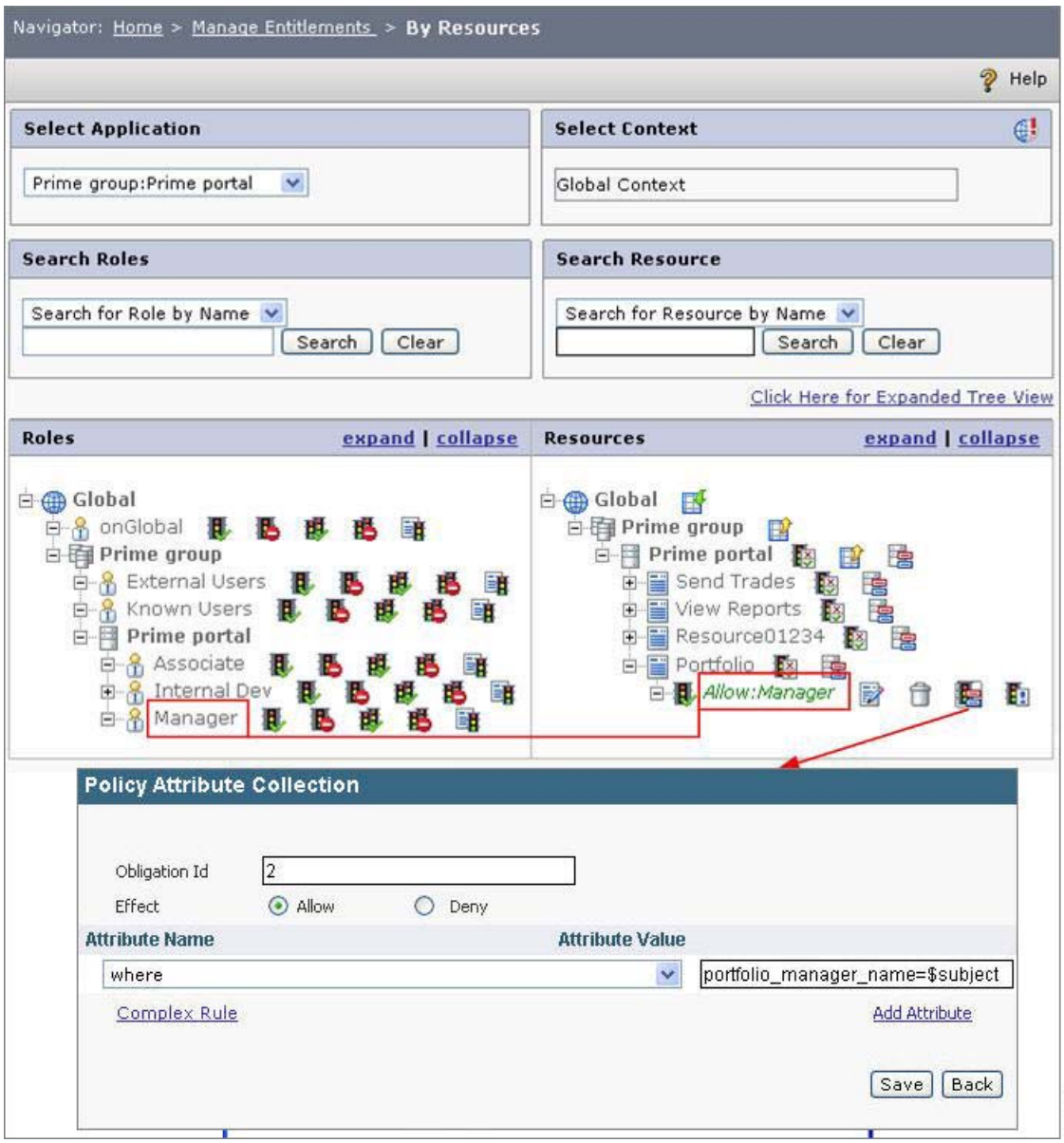
To configure the VPD Agent component in CEPM, follow these steps:

Step 1 In the PAP, create a resource with same name as the table name (For example, Portfolio) under the application for which the previously described authorization policy is to be implemented.

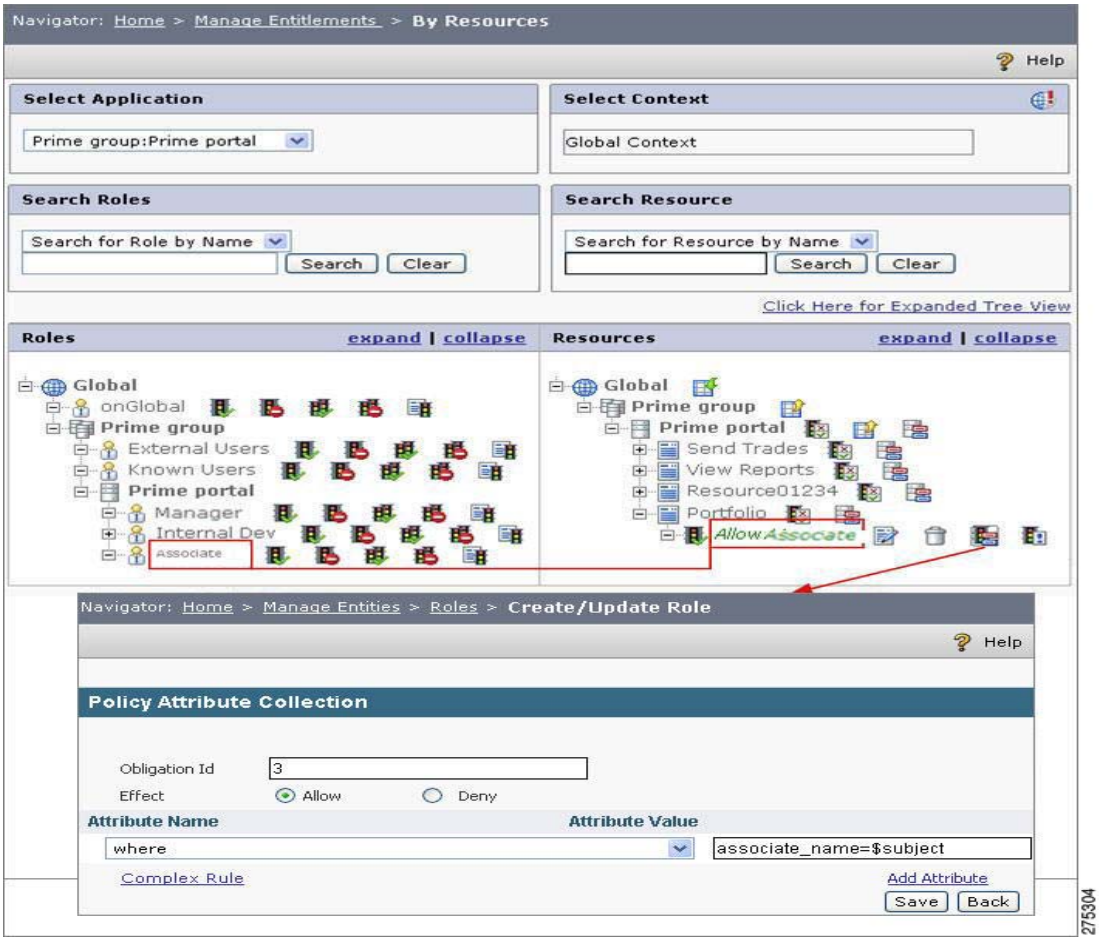
Step 2 Add user Tom to the Manager role and users Mary and James to the Associate role.



Step 3 Map the Manager role to the Portfolio resource in the Resource Based Entitlements page to create an Allow policy. Click the **Policy Attribute** button, and in the Policy Attributes window, enter WHERE in the Name box and portfolio_manager_name = $subject in the Value box. Click **Create** to save the policy attribute.



Step 4 Map the Associate role to the Portfolio resource in the Resource Based Entitlements page to create an Allow policy. Click the **Policy Attribute** button, and in the Policy Attributes window, enter WHERE in the Name box and associate_name = $subject in the Value box. Click **Create** to save the policy attribute.



Note The portfolio_manager_name and associate_name values entered in the Name boxes should match the corresponding column names of the Portfolio table in the database.

Configuring the VPD Component in the Application Database

Oracle implements VPD on any table using a package function called DBMS_RLS.ADD_POLICY(). The syntax of this function is as shown here:

```
DBMS_RLS.ADD_POLICY (
   object_schema            IN VARCHAR2 NULL,
   object_name              IN VARCHAR2,
   policy_name              IN VARCHAR2,
   function_schema          IN VARCHAR2 NULL,
   policy_function          IN VARCHAR2,
   statement_types          IN VARCHAR2 NULL,
   update_check             IN BOOLEAN  FALSE,
```

```
   enable                   IN BOOLEAN  TRUE,
   static_policy            IN BOOLEAN  FALSE,
   policy_type              IN BINARY_INTEGER NULL,
   long_predicate           IN BOOLEAN  FALSE,
   sec_relevant_cols        IN VARCHAR2,
   sec_relevant_cols_opt    IN BINARY_INTEGER NULL);
```

To configure the VPD component in the application database, follow these steps:

Step 1 Connect to Oracle client for the application database instance with the user credentials having administrator privileges.

Step 2 Execute the DBMS_RLS.ADD_POLICY() package function in the SQL prompt as shown here:

```
BEGIN
  DBMS_RLS.ADD_POLICY (object_schema     => 'scott',
                       object_name       => 'portfolio',
                       policy_name       => 'sp_job',
                       function_schema   => 'cepm',
                       policy_function   => 'pf_job' );
END;
/
```

Note The **DBMS_RLS.ADD_POLICY()** package function takes a total of 13 parameters as input. For our use case we need to set only the following five parameters.

```
Object schema   =>   'scott',    (schema name of the table)
Object name     =>   'portfolio',(table name)
Policy name     =>   'sp_job',   (user defined name for the policy)
Function schema =>   'cepm', (schema name of the policy function)
Policy function =>   'pf_job'    (name of the function)
```

The policy function (pf_job) is the user defined function, which appends the WHERE predicate dynamically to the query. The policy function (pf_job) has to be written and compiled first.

Use Case Operation

Step 1 Log in to the application that needs to be secured using the credentials of user Tom. Execute the query - Select * from the Portfolio table. The DBMS_RLS package written for the Portfolio table executes the function pf_job(). The pf_job() function internally calls CEPM isUserAccessAllowed() method passing user name as Tom and resource name as Portfolio.

CEPM returns the decision for the user Tom and the policy attribute ('WHERE predicate') to the VPD Agent function pf_job().

In our use case, the decision for Tom on the Portfolio table is Allow and the policy attribute is *portfolio_manager_name = $subject*, which evaluates to portfolio_manager_name = 'Tom'.

VPD Agent function **pf_job()** takes this policy attribute and dynamically appends to Tom's requested query:

```
Select * from portfolio.
```

Now the new query becomes:

```
Select * from portfolio WHERE portfolio_manager_name = 'Tom'.
```

The query returns all the records from Portfolio table where the value for portfolio_manager_name column is Tom.

Step 2 Now log in to the application using the credentials of user Mary. Execute the query:

```
Select * from portfolio.
```

The DBMS_RLS package written for the portfolio table executes the function pf_job(). The **pf_job()** function internally calls CEPM **isUserAccessAllowed()** method passing user name as Mary and resource name as portfolio.

CEPM returns the decision for the user Mary and the policy attribute ('WHERE predicate') to **pf_job()** function.

In our use case, the decision for Mary on the Portfolio table is Allow and the policy attribute is *associate_name = $subject*, which evaluates to associate_name = 'Mary'.

Function pf_job() takes this policy attribute and dynamically appends to Mary's requested query:

```
Select * from portfolio.
```

Now the new query becomes:

```
Select * from portfolio WHERE associate_name = 'Mary'.
```

The query returns all the records from the Portfolio table where the value for associate_name column is 'Mary'.

Configuring the WHERE Clause Predicate

The WHERE clause predicate can be configured in the VPD Agent in two ways.

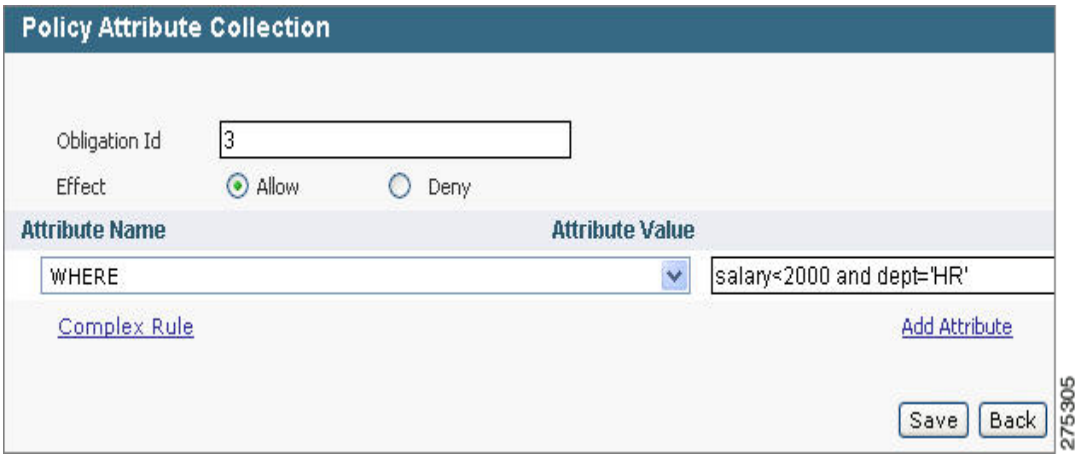
Using Policy Attributes in PAP to Configure the WHERE Clause Predicate

To configure the WHERE clause predicate using policy attributes, follow these steps:

Step 1 In PAP, create a resource with the same name as that of the database table (for example, Portfolio) for which VPD has to be implemented.

Step 2 Create the policy for this resource for the needed user role.

Step 3 Create the policy attribute on this policy so as to apply filter on columns, such as salary and dept, of the Portfolio table.



In this case, the application user will be able to access only those database records from the Portfolio table that have salary value less than 2000 and department name of HR.

Note Do not choose the values for the Attribute Name and Attribute Value from the drop- down list while creating a rule. Enter the fresh values for Attribute Name and Attribute Value. The fresh values should not belong to any of the entity types.

Note You can also use $subject as policy attribute value, which evaluates to the name of the user who is making the request for the resource.

For example, if policy attribute value is set as user=$subject, for a policy created for user Mary, then when Mary tries to access that resource, this policy attribute value evaluates to user=Mary.

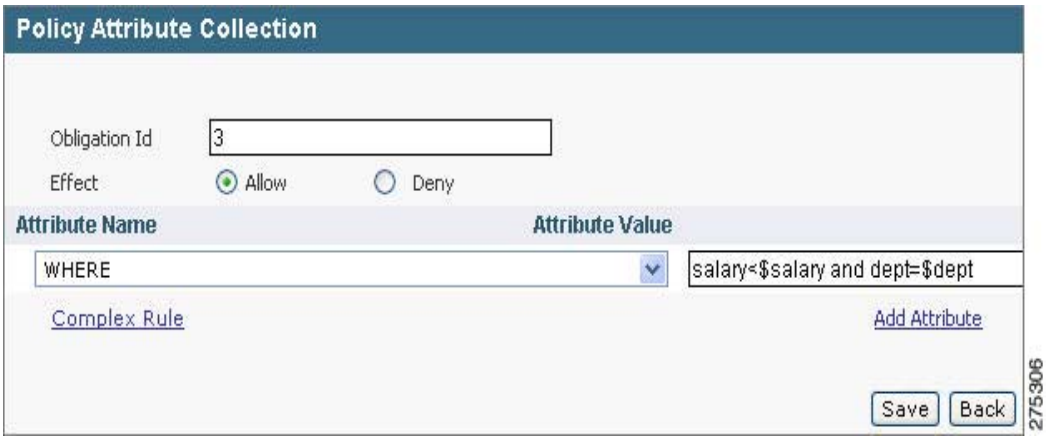
Using Oracle Application Context Attributes to Configure the WHERE Clause Predicate

To configure the WHERE clause predicate using Oracle application context attributes, follow these steps:

Step 1 In PAP, create a resource with the same name as that of the database table (for example, Portfolio) for which VPD has to be implemented.

Step 2 Create the policy for this resource for the needed user role.

Step 3 Create the policy attribute on this policy so as to apply filter on columns, such as salary and dept, of the Portfolio table.



Step 4 secpck.setsecctx

```
secpck.setsecctx('[username]', 'salary<2000, dept=''HR'' ')
```

The secpck.setsecctx package takes the parameters (username) and comma-separated values of attribute values. This package in turn calls the Oracle package, **dbms_session.set_context()**, to set the Oracle application context attributes for salary and dept. Give the appropriate name of the user in place of [username] in single quotes in this code.

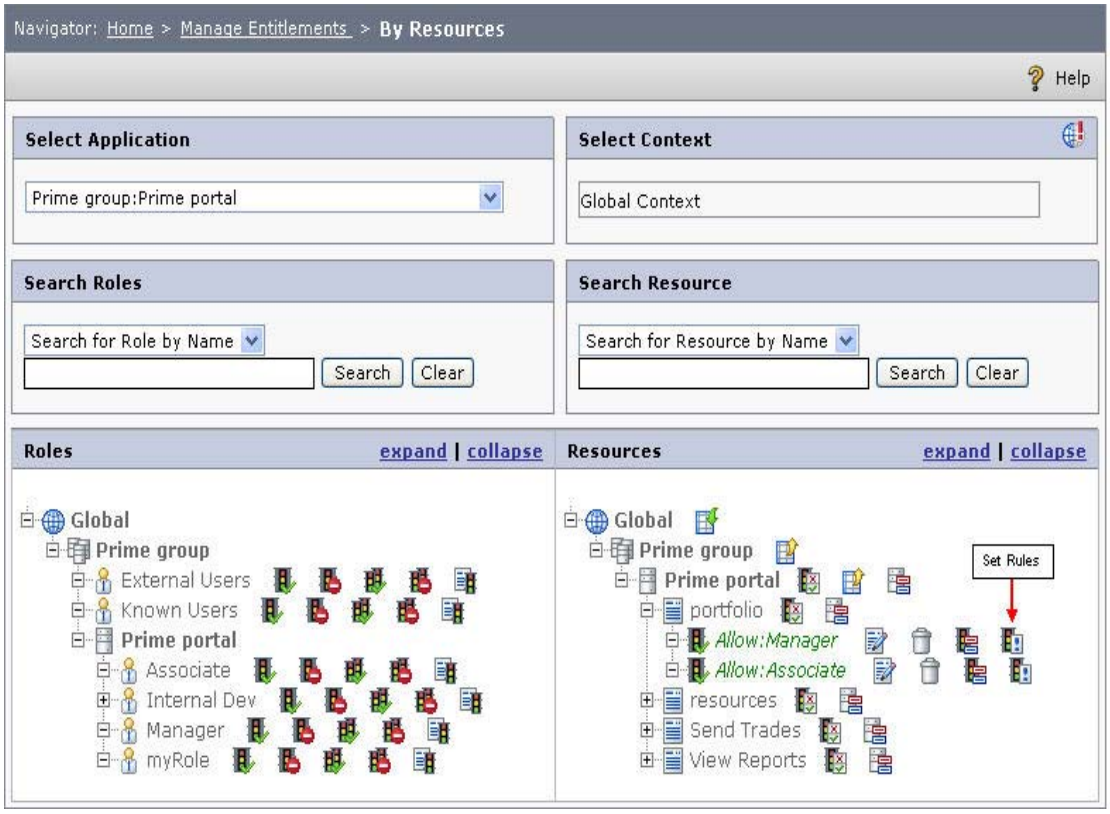
The application user will now be able to access only those database records from the Portfolio table that have salary value less than 2000 and department name of HR.

Note Do not select the values for the Attribute Name and Attribute Value from the drop- down list while creating a rule. Enter the fresh values for Attribute Name and Attribute Value. The fresh values should not belong to any of the entity types.

Creating Rules for Accessing the VPD Resource

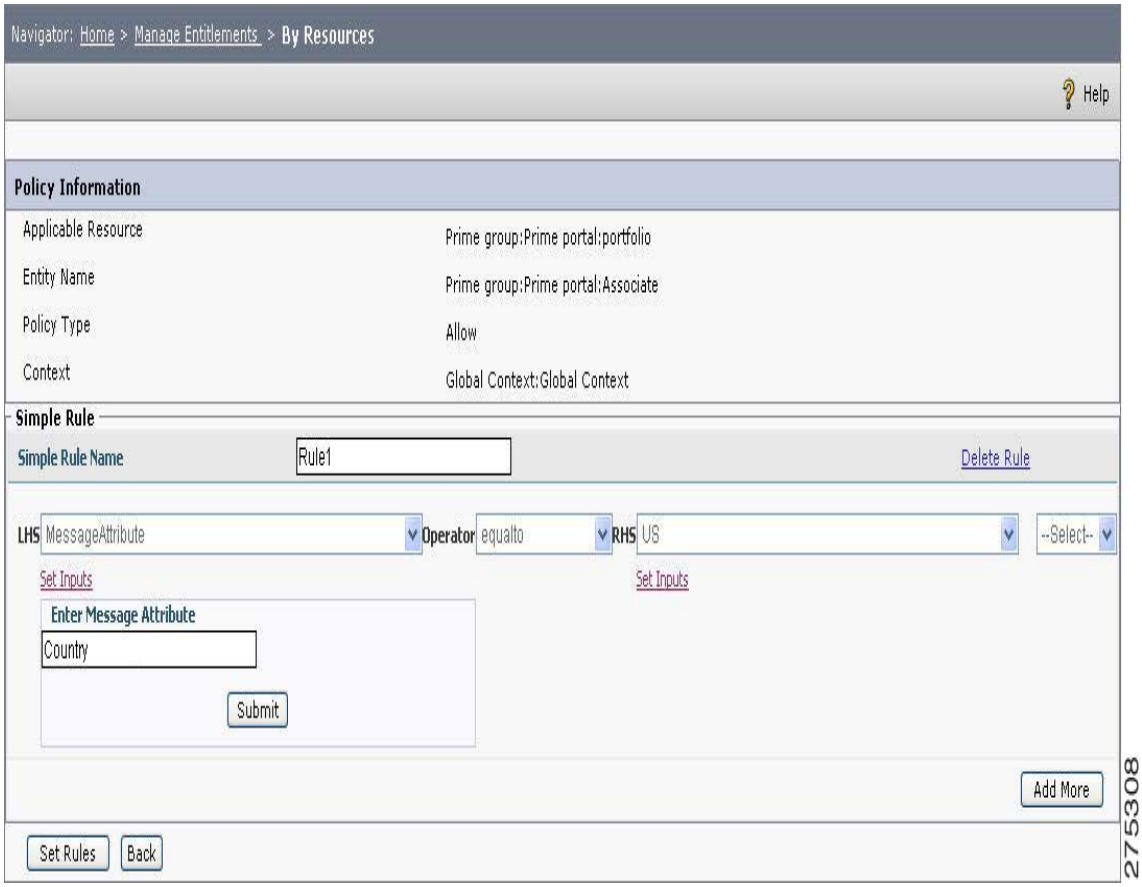
The database table for which the VPD has to be implemented should be created as a resource in PAP.

Thus multiple rules can also be created for the VPD resource for implementing finer level of authorization. The only way to create rules for the VPD resource is to use CEPM MessageAttributes.



Click the Set Rules icon to open the Rule Editor page.

In this example, when the user tries to access the VPD resource for which the above rule (Rule1) is configured, then the check is done if the application context attribute contains the Country attribute having the value of US. If YES, then the next step will be to execute the rest of the VPD function. If NO, then the user will not be allowed to access that resource.



Scope of Data Protection Using VPD Agent

Apart from simple select queries, the VPD Agent policy works similarly for queries containing table joins and subqueries, that is, only those records are fetched for queries having table joins and for subqueries that are allowed as per the policy.

The VPD Agent policies also work on insert, update, and delete statements. Thus updates and deletions can be carried out only on those database records that are not restricted by the policy. Also insertion operations are allowed for a user for a table, if the policy allows it.

If a synonym exists for a table for which the policy is to be implemented, then use the original table name in the VPD Agent configuration.

The VPD Agent policy also gets executed for a table when the table is referenced from a PL/SQL block or a database function or a database procedure.

In PAP, the resource name should be unique for the VPD agent to work.

In PAP, the resource name should be unique for the if no policy attributes are specified and if the decision for the table is true, then the user requesting can see the entire records of the table.

Rules on a policy for a VPD Agent-enabled resource will be evaluated only if the PAP application, LHS part of the rule is a message attribute and RHS part of the rule is a constant. Otherwise the decision will be given as deny.

Rules for a policy can have only one condition for a VPD Agent-protected resource.