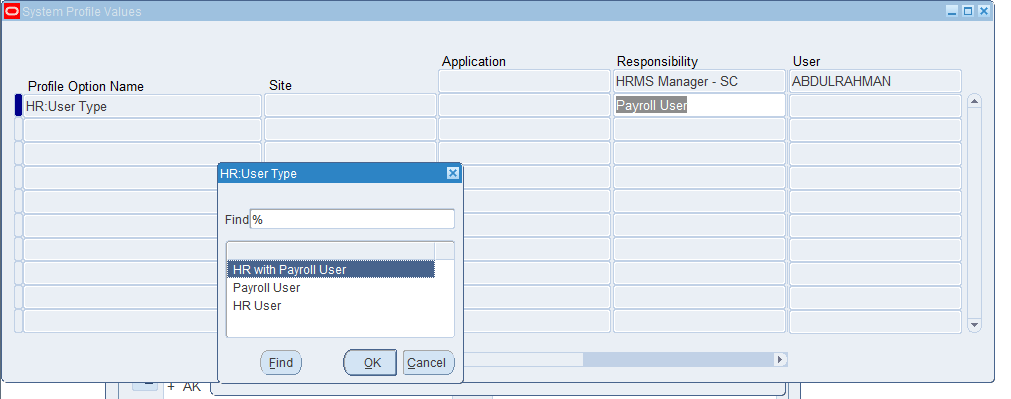
# How to Disable Access to Payroll Salary information from Apps (Backend user ) but still Enable Access Only from Authorized and Audited Front End users (Payroll user).

## Advantages of this Exercise

1. Sysadmin will be able to control (Add/ Delete) Access to Salary information and Assign Access from the Seeded profile HR: User Type for a particular Responsibility
2. Access to Payroll Table will be given only for this particular Responsibility and not to anyone Else even from Backend.
3. Responsibility Access can be Audited at any point



## About This Exercise

Need to create an Oracle Virtual Private Database policy that limits access to all Salary Information in the APPS.PER\_PAY\_PROPOSALS table. In essence, the policy translates the following statement:

SELECT \* FROM APPS.PER\_PAY\_PROPOSALS;

To the following statement:

SELECT \* FROM APPS.PER\_PAY\_PROPOSALS

WHERE fnd\_profile.value(‘HR\_USER\_TYPE’ ) =’Payroll User’

### Step 1: Ensure That the APPS User Account Is Active

1. Log on to SQL\*Plus as user SYSTEM with the SYSDBA privilege.
2. sqlplus apps
3. Enter password: *password*

## Step 2: Create a Policy Function

Create the following function, which will append the

fnd\_profile.value(‘HR\_USER\_TYPE’ ) =’Payroll User’

 Clause to any SELECT statement on the APPS.PER\_PAY\_PROPOSALS table. (You can copy and paste this text by positioning the cursor at the start of CREATE OR REPLACE in the first line.)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | CREATE OR REPLACE FUNCTION XXSC\_RESTRICT\_SALARY\_ACCESS(  schema\_var IN VARCHAR2,  table\_var IN VARCHAR2  )  RETURN VARCHAR2  IS  return\_val VARCHAR2 (400);  BEGIN  return\_val := 'fnd\_profile.value(''HR\_USER\_TYPE'' ) IN (''PAY'',''INT'') AND EXISTS ( SELECT 1 FROM V$SESSION where AUDSID = userenv(''SESSIONID'') and OSUSER=''applmgr'')';  RETURN return\_val;  END XXSC\_RESTRICT\_SALARY\_ACCESS; |

In this Exercise:

* **Lines 2–3:** Create input parameters to specify to store the schema name, APPS, and table name, PER\_PAY\_PROPOSALS . First, define the parameter for the schema, and then define the parameter for the object, in this case, a table. Always create them in this order. The Virtual Private Database policy you create will need these parameters to specify the APPS.PER\_PAY\_PROPOSALS table.
* **Line 5:** Returns the string that will be used for the WHERE predicate clause. Remember that return value is always a VARCHAR2 data type.
* **Lines 6–10:** Encompass the creation of the predicate.

fnd\_profile.value(‘HR\_USER\_TYPE’ ) =’Payroll User’

### Step 3: Create the Oracle Virtual Private Database Policy

1. Log on to SQL\*Plus as user SYSTEM with the SYSDBA privilege.
2. sqlplus sys as sysdba
3. Enter password: *password*

Next, create the following policy by using the ADD\_POLICY procedure in the DBMS\_RLS package. (You can copy and paste this text by positioning the cursor at the start of BEGIN in the first line.)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | BEGIN  DBMS\_RLS.ADD\_POLICY (  object\_schema => 'HR',  object\_name => 'PER\_PAY\_PROPOSALS' ,  policy\_name => 'XXSC\_RESTRICT\_SALARY\_ACCESS',  function\_schema => 'apps',  policy\_function => 'XXSC\_RESTRICT\_SALARY\_ACCESS',  statement\_types => 'select, insert, update, delete'  );  END;  / |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | BEGIN  DBMS\_RLS.ADD\_POLICY (  object\_schema => 'HR',  object\_name => 'PAY\_COSTS' ,  policy\_name => 'XXSC\_RESTRICT\_PAY\_COSTS\_ACCESS',  function\_schema => 'APPS',  policy\_function => 'XXSC\_RESTRICT\_SALARY\_ACCESS',  statement\_types => 'select, insert, update, delete'  );  end;  / |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | BEGIN  DBMS\_RLS.ADD\_POLICY (  object\_schema => 'HR',  object\_name => 'PAY\_BALANCE\_FEEDS\_F' ,  policy\_name => 'XXSC\_RESTRICT\_BALANCES\_ACCESS',  function\_schema => 'APPS',  policy\_function => 'XXSC\_RESTRICT\_SALARY\_ACCESS',  statement\_types => 'select, insert, update, delete'  );  END;  / |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | BEGIN  DBMS\_RLS.ADD\_POLICY (  object\_schema => 'HR',  OBJECT\_NAME => 'PAY\_RUN\_RESULTS' ,  policy\_name => 'XXSC\_RESTRICT\_RUN\_RES\_ACCESS',  function\_schema => 'APPS',  policy\_function => 'XXSC\_RESTRICT\_SALARY\_ACCESS',  statement\_types => 'select, insert, update, delete'  );  end;  / |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | BEGIN  DBMS\_RLS.ADD\_POLICY (  object\_schema => 'HR',  object\_name => 'PAY\_ELEMENT\_ENTRY\_VALUES\_F' ,  policy\_name => 'XXSC\_RESTRICT\_ENTRY\_VALUES',  function\_schema => 'APPS',  policy\_function => 'XXSC\_RESTRICT\_SALARY\_ACCESS',  statement\_types => 'select, insert, update, delete'  );  END;  / |

In this example:

* **Line 3:** Specifies the schema that you want to protect, that is, Apps.
* **Line 4:** Specifies the object within the schema to protect, that is, the PER\_PAY\_PROPOSALS table.
* **Line 5:** Names this policy Salary Information policy.
* **Line 6:** Specifies the schema in which the XXSC\_RESTRICT\_SALARY\_ACCESS function was created. In this example, XXSC\_RESTRICT\_SALARY\_ACCESS was created in the SYS schema. But typically, it should be created in the schema of a security administrator.
* **Line 7:** Specifies a function to enforce the policy. Here, you specify the XXSC\_RESTRICT\_SALARY\_ACCESS function that you created in [Step 2: Create a Policy Function](http://docs.oracle.com/cd/E28271_01/network.1111/e16543/vpd.htm#CIHCAACD).
* **Line 8:** Specifies the operations to which the policy applies. In this example, the policy applies to all SELECT, INSERT, UPDATE, and DELETE statements the user may perform.

### Step 4: Test the Policy

After you create the Oracle Virtual Private Database policy, it goes into effect immediately. The next time a user, including the owner of the schema, performs a SELECT on APPS.PER\_PAY\_PROPOSALS, only the Salary Information Log on as user apps.

1. CONNECT apps
2. Enter password: password
3. Enter the following SELECT statement:
4. SELECT COUNT(\*) FROM PER\_PAY\_PROPOSALS ;

The following output should appear:

COUNT(\*)

---------

0

The policy is in effect for user apps : As you cannot see any rows in the Salary Information table.

But users with administrative privileges (Responsibility with Profile HR:User Type) still have access to all the rows in the table.

1. Log back on as application user Janine
2. Enter password: password
3. Query from the Front End you should be able to see information for all user from Janine Login

### Step 5: Remove the Components for This Exercise

1. As user SYS, remove the function and policy as follows:
2. DROP FUNCTION XXSC\_RESTRICT\_SALARY\_ACCESS;
3. as SYS user run

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | EXEC DBMS\_RLS.DROP\_POLICY('HR','PER\_PAY\_PROPOSALS','XXSC\_RESTRICT\_SALARY\_ACCESS');  EXEC DBMS\_RLS.DROP\_POLICY('HR','PAY\_COSTS' ,'XXSC\_PAY\_COSTS\_ACCESS');  EXEC DBMS\_RLS.DROP\_POLICY('HR','PAY\_ELEMENT\_ENTRY\_VALUES\_F' ,'XXSC\_RESTRICT\_ENTRY\_VALUES');  EXEC DBMS\_RLS.DROP\_POLICY('HR', 'PAY\_BALANCE\_FEEDS\_F' ,'XXSC\_RESTRICT\_BALANCES\_ACCESS');  EXEC DBMS\_RLS.DROP\_POLICY('HR','PAY\_RUN\_RESULTS','XXSC\_RESTRICT\_RUN\_RES\_ACCESS'); |