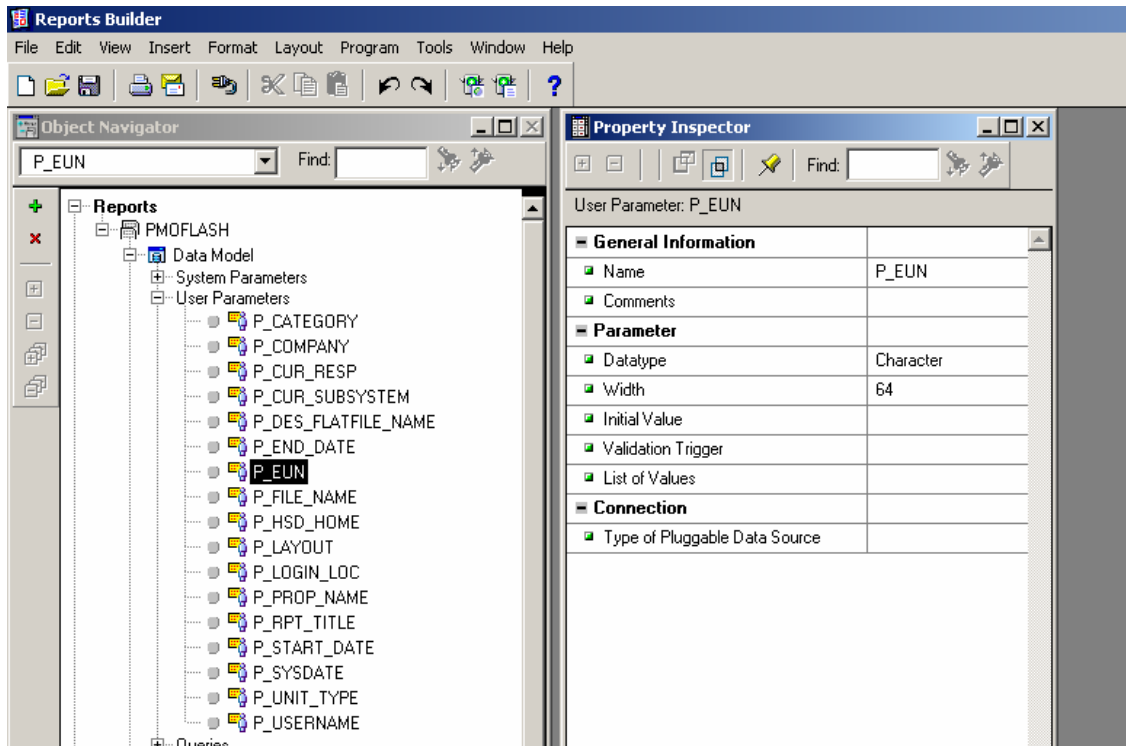


Create a new parameter called **P_EUN** which is an acronym for Encrypted User Name.



Add this code to the before parameter form trigger .

```
function BeforePForm return boolean is
p_decrypted_username varchar2(64);
begin
    select CO_SENSITIVE_DATA.decryptor(:P_EUN)into p_decrypted_username from dual;
    if p_decrypted_username is not null then :p_username := p_decrypted_username; end if;

    rep_global.set_username(:p_username);
    SELECT name INTO :P_company from co_company;
    return (TRUE);
end;
```

Theory:

Obtaining the URL string for a report:

To kick off a report we need to build a URL which is made up of many components. Some components are stored in the database and can be called using functions to help us build the URL.

This returns the initial part of the URL to run a report on the report server:

```
select co_get_param('REP_URL') from dual;
```

The character following this is a "?" (question mark).

This question mark indicates the first parameter you are sending to the report builder.

What database are we logged into?

```
select name from v$database
```

Once we have the result (DEV/TEST/ or PROD) we need to get the login id and password for the machine we are on.

We cannot simply pass sensitive information on the command line so we must hide it.

Hiding the login string in the URL:

System administrators need to maintain a file on the report server called cgicmd.dat. This file contains a relationship between a key and a value. We use this key/value combination to hide the login strings hsdappspub/hsdappspub@dev for test and prod databases also. While any **key/value** combinations may be kept in the cgicmd.dat file on the server we will be using it to hid the login/password for the machine we are on when we build a URL to kick off a report. To get the report server to see the cgicmd.dat file, the RWSERVERLET.PROPERTIES must be set up by the administrator.

The **Key** is simply a text word in the cgicmd.dat file which relates to a value or group of values (such as an entire set of parameters). For our sake we have limited relationship of key/value to only return the login/password@database values when we pas a key of DEV, TEST, or PROD. When we pass DEV we get the associated login/password@database for the DEV database. Selecting the key from the file requires assigning assigning a variable called cmdkey to our key selection. In this case we assign cmdkey='DEV'. The variable cmdkey must be populated when building your URL. There is one exception when we do not actually need to make the assignment of 'DEV' to the cmdkey. That exception is when we assign the value of 'DEV' in the **position of the first parameter following the "?" on the URL**. We simply make pass ?DEV followed by all the other parameters we are passing. When using cmdkey to make the assignment cmdkey='DEV' it does not appear that parameter position is important. TEST and PROD are also names of keys which return the login string which will be hidden on the URL. All must be uppercase.

Passing the server name on the URL:

The example below (taken from the format trigger above) is saying the report server will be taken from a function stored in the database.

```
'?server=' || co_get_param('REP_SERVER')
```

It also indicates (by the question mark) that this is the first in a set of parameters being passed to the report server.

All other parameters that follow will be prefixed with "&" (ampersand).

Encrypting P_USERNAME:

The example below (taken from the format trigger above) is saying to encrypt the user name on the URL and pass it through the new parameter we created called P_EUN (Encrypted User Name).

```
'&p_eun=' || CO_SENSITIVE_DATA.encryptor(:p_username)
```

Decrypting P_USERNAME:

The username will be decrypted in the **before parameter form trigger** if P_EUN contains data.

Turning Off The Hyperlink For Printing:

Previous use of button objects on reports never printed on the paper. Because hyperlinks are text objects they will show up on paper.

A parameter will be required to do mute them when printing.

Once the PDF has been created, the report has stopped running. All hyperlinks will have been created inside the PDF which will be passed to the browser if clicked. The URL contains the instruction for the report server to start a report with the parameters on the URL.

Because hyperlinks are text, they will print on the report unless turned off. Here a new parameter called :p_turn_off_hyperlinks is being coded to control hyperlinks.

Please notice that the :p_turn_off_hyperlinks parameter lives in two places in the format trigger. One place turns off the hyperlink and the other place passes it if the hyperlink is left on.

The screenshot displays the Reports Builder interface with several key components:

- Object Navigator:** Shows a tree view of the report structure. Under 'User Parameters', the parameter `P_TURN_OFF_HYPERLINKS` is selected.
- Property Inspector:** Shows the configuration for the selected parameter:

User Parameter: p_turn_off_hyperlinks	
General Information	
Name	p_turn_off_hyperlinks
Comments	
Parameter	
Datatype	Character
Width	3
Initial Value	NO
Validation Trigger	
List of Values	...
Connection	
Type of Pluggable Data Source	
- Format Trigger Editor:** Shows the PL/SQL code for the `B_11FORMATTRIGGER` object:


```
function B_11FormatTrigger return boolean is
  p_database v$database.name%TYPE;
begin
  if :p_turn_off_hyperlinks = 'YES' then return(false); end if;
  select name into p_database from v$database;
  SRW.SET_HYPERLINK(
    co_get_param('REP_URL')
    '?server='           || co_get_param('REP_SERVER')
    '&report='           || :p_hsd_home||'/rs/reports/pmoflas
    '&cmdkey='           || p_database
    '&paramform=no'     ||
    '&destype=cache'    ||
    '&desformat=pdf'    ||
    '&sp_turn_off_hyperlinks=' || :p_turn_off_hyperlinks
```
- Parameter List of Values Dialog:** A dialog box for defining the list of values for the parameter. It is set to 'Static Values' and shows a list containing 'YES' and 'NO'.

