HOW TO INTEGRATE APPLICATIONS RELEASE 11i WITH CUSTOM APPLICATIONS

OVERVIEW

This article contains information on how to integrate custom Applications with the APPS schema. It is not meant to fix and/or solve all issues, but it is a good guideline. If custom Applications are not integrated properly, problems can occur with alerts, reports, requests, programs, etc.

It is recommended that you develop the custom applications code by following the standards exactly as described in the Oracle Applications Developers Guide and the Oracle Applications User Interface Standards manual. Deviations from these standards can have unpredictable results.

Oracle Applications release 11i is a complete suite of over 150 products (AP, PO, GL etc.) that uses a java-based interface. Oracle Applications 11i is a multi-tier architecture that allows functions to be distributed among multiple tiers of servers (forms server, administration server, web server, concurrent processing server and database server).

You will want the following documentation for reference:

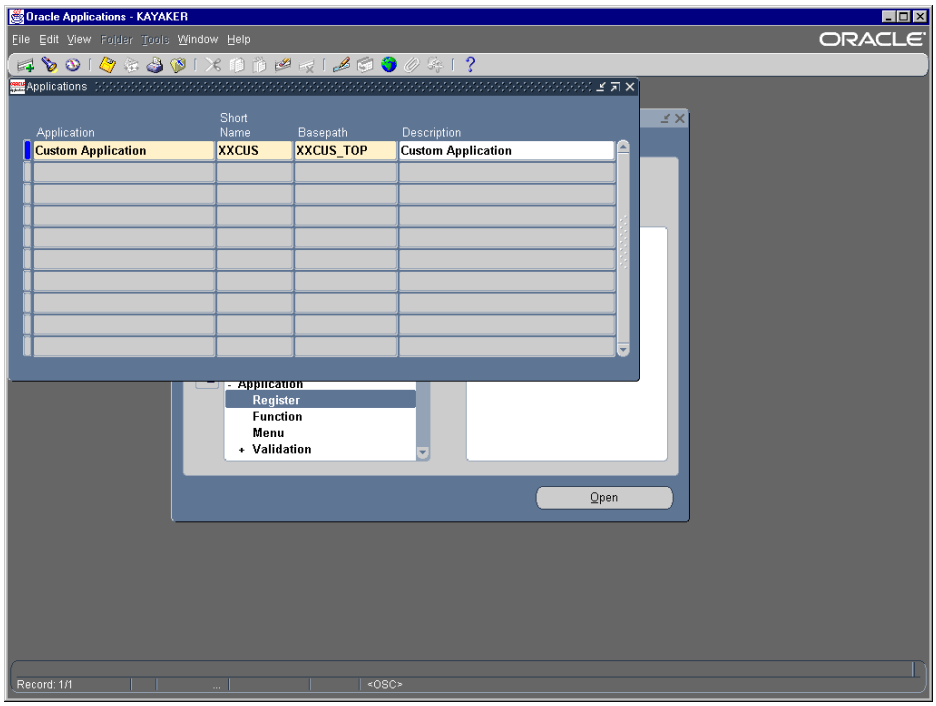
Oracle Applications Developers Guide, Volume 1 & 2 Release 11i
Oracle Applications System Administrator's Guide Release 11i
Oracle Applications Installing Oracle Applications Release 11i
Oracle Applications User Interface Standards Release 11i for Forms-Base Products
Oracle Alert User's Guide release 11i

PROCEDURE

Note: The following steps given are for a UNIX system, just make the appropriate changes for your specific operating system.

1) Register your custom application with the Application Object Library. It is recommended to use an XX as the preface to the custom schema short name so that it doesn't conflict with any future Oracle Application short names.

   Log into Applications as the System Administrator and navigate to Application → Register.



   Refer to "Oracle Applications Developers Guide, Volume 1 Release 11i" page 2-6.

2) Create a custom directory tree for your custom schema as the APPLMGR user. Use the basepath parameter from the Application registration for the top level directory. This top level directory will reside just under APPL_TOP. The subdirectories under the custom directory may vary depending on the server type (forms server, concurrent processing server, etc.). Make sure that the rights/protections are open for the world (rwx).

APPL_TOP (/u02/oracle/visappl)
|
XXCUS_TOP (/u02/oracle/visappl/XXCUS) → other product directories
|

| | | | | | | |
bin forms help lib log mesg out reports
| |
US US

Refer to "Oracle Applications Developers Guide, Volume 1 Release 11i" page 2-2.

3) Modify the applications environmental file (example: APPSORA.env) as the APPLMGR user to include the custom schema basepath.

XXCUS_TOP="/u02/oracle/visappl/XXCUS"
export XXCUS_TOP

4) Register the custom schema as an Oracle user.

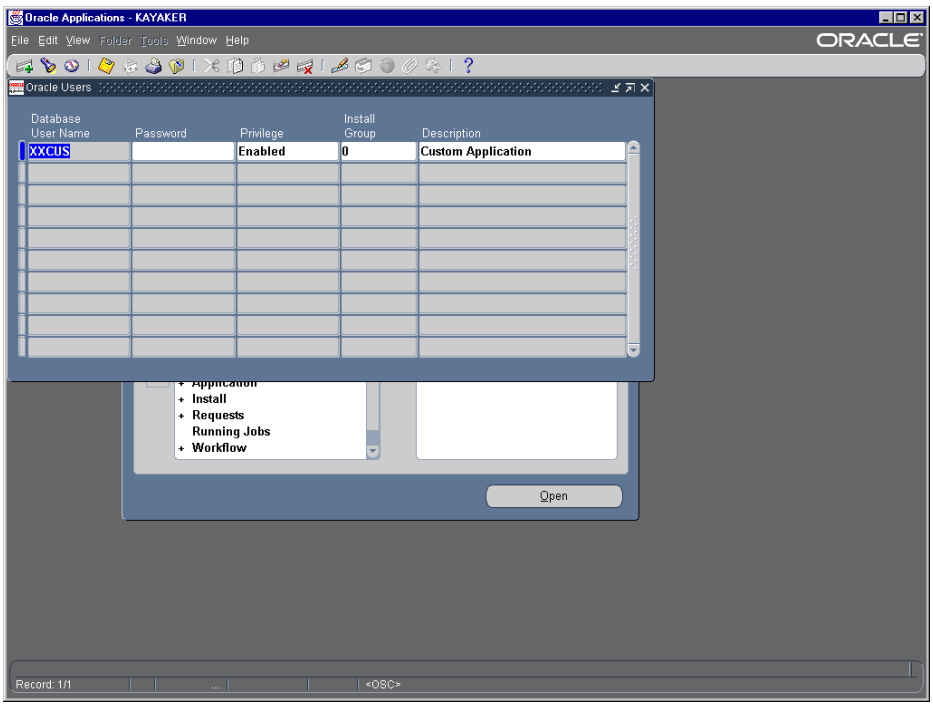
a) Create the user in the RDBMS database using SQL*Plus under the system account. Give the user a default and temporary tablespace with quotas and then grant the CONNECT role.

For example:

$ sqlplus system/systempword
SQL> create user XXCUS identified by CUST default tablespace USER_DATA temporary tablespace TEMP quota unlimited on USER_DATA quota unlimited on TEMP;
SQL> grant connect to XXCUS identified by CUST;

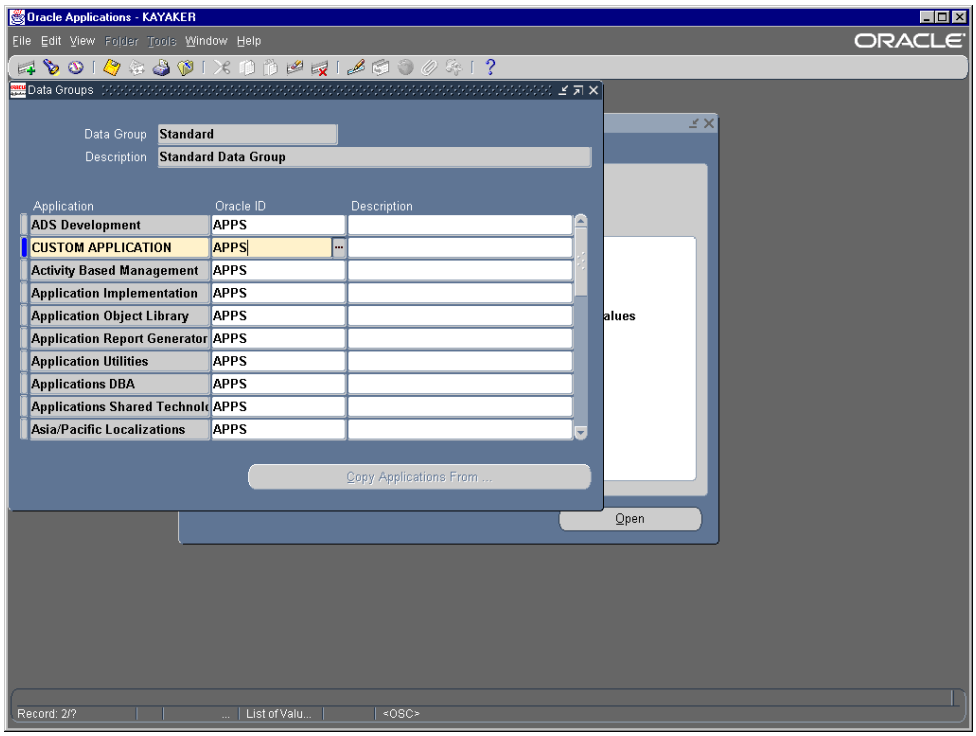
XXCUS is the product short name, CUST is the password for the custom schema, USER_DATA and TEMP are existing tablespaces.

b) Register the user with the Application Object Library. Log into Applications as the System Administrator and navigate to Security → ORACLE → Register.



Refer to "Oracle Applications System Administrator's Guide, Release 11i " page 8-9.

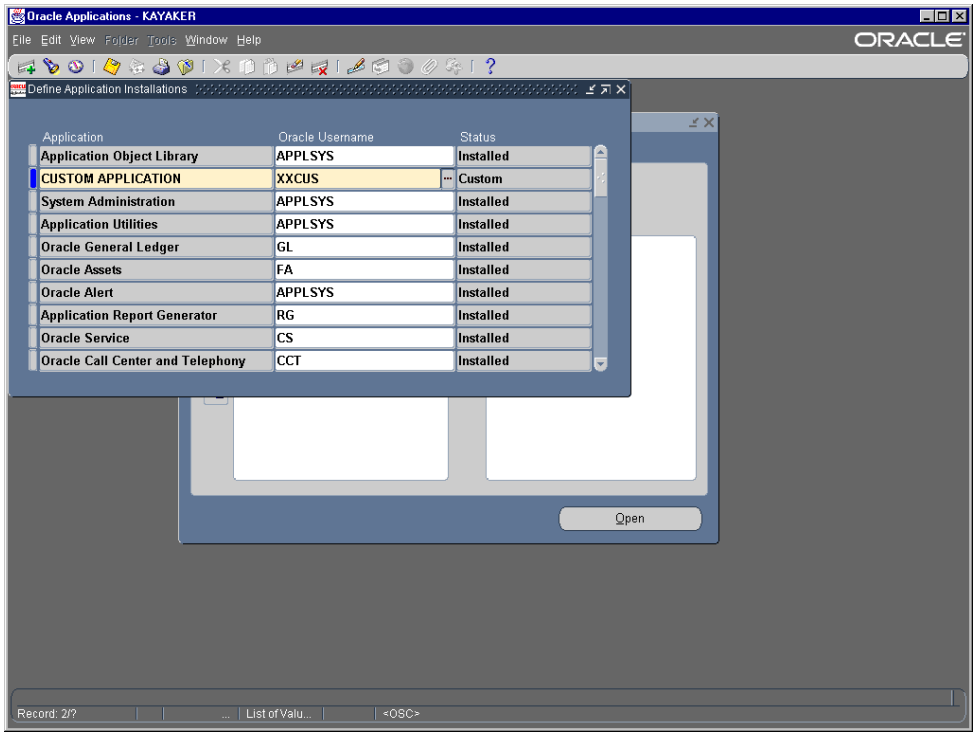
5) Add the custom schema to a data group. Log into Applications as the System Administrator and navigate to Security → ORACLE → DataGroup.



It is recommend that you use the STANDARD datagroup and pair the custom schema with APPS or you can add a new data group. This depends upon your own requirements.

Refer to " Oracle Applications System Administrator's Guide, Release 11i " page 4-78.

6) Register your custom application with Alerts. Log into Applications as the Alert Manager and navigate to System → Installations.



Refer to " Oracle Alert User's Guide, Release 11i " page 7-25.

7) Create your custom tables, indexes, views and sequences. It is suggested that you add WHO columns to your custom tables so that Oracle Applications can keep track of customizations. Register your custom schema's tables (including flexfields) with the PL/SQL package AD_DD. You use the procedure AD_DD.register_table for the custom schema tables and AD_DD.register_column for the custom schema table columns.

execute ad_dd.register_table (appl short name, table name, table type, next extent, % free, % used)

For example:

$sqlplus apps/appspword
SQL> execute ad_dd.register_table (‘XXCUS’, 'CUST_TABLE' , 'T', 8, 10, 90)

where appl short name = ‘XXCUS’, table name = 'CUST_TABLE' , table type = 'T', next extent = 8, % free = 10, % used = 90

execute ad_dd.register_column (appl short name, table name, column name, column seq, column type, column width, null, translate,precision,scale)

For example:

$sqlplus apps/appspword
SQL> execute ad_dd.register_column (‘XXCUS’, 'CUST_TABLE' , 'CUST_NO',1, ‘NUMBER’,5,‘N’, ‘N’)

where appl short name = ‘XXCUS’, table name = 'CUST_TABLE' , column name = 'CUST_NO', column seq = 1, column type = ‘NUMBER’ , column width = 5, null = ‘N’, translate = ‘N’

Refer to "Oracle Applications Developers Guide Release 11i, Volume 1" page 3-2.

8) Run the custom schema against the APPS_DDL and APPS_ARRAY_DDL packages by running the scripts $AD_TOP/admin/sql/adaddls.pls, adaaddls.pls, adaddlb.pls and then adaaddlb.pls (in this order) under SQL*Plus:

$ sqlplus apps/appspword
SQL> @$AD_TOP/admin/sql/adaddls.pls system_pword custom_schema custom_schema_pword
SQL> @$AD_TOP/admin/sql/adaaddls.pls system_pword custom_schema custom_schema_pword
SQL> @$AD_TOP/admin/sql/adaddlb.pls system_pword custom_schema custom_schema_pword
SQL> @$AD_TOP/admin/sql/adaaddlb.pls system_pword custom_schema custom_schema_pword

9) Integrate your database objects with the APPS schema by granting APPS the access to your custom schema's objects.

a) Grant all privileges from each custom data object to APPS.

For example:

$ sqlplus xxcus/cust
SQL> grant all on CUST_TABLE to APPS

b) Create a synonym in APPS to each custom data object.

For example:

$ sqlplus apps/appspword
SQL> create synonym APPS.CUST_TABLE for XXCUS.CUST_TABLE

c) Create custom code objects in APPS

For example:

$ sqlplus apps/appspword
SQL> create function CUST_FUNCTION…

Refer to "Oracle Applications Developers Guide Release 11i, Volume 2" page 27-29.

10) Build your custom forms, menus and libraries as the APPLMGR user. Use the TEMPLATE form located in $AU_TOP/forms/US directory as the required starting point for your development work for your custom forms. Follow the form development steps (which incorporate the development standards) that are described in the Oracle Application Developers Guide. You will modify the form as needed, move the generated form to its proper directory, register the form with the Oracle Application Object Library and then add it to a menu.

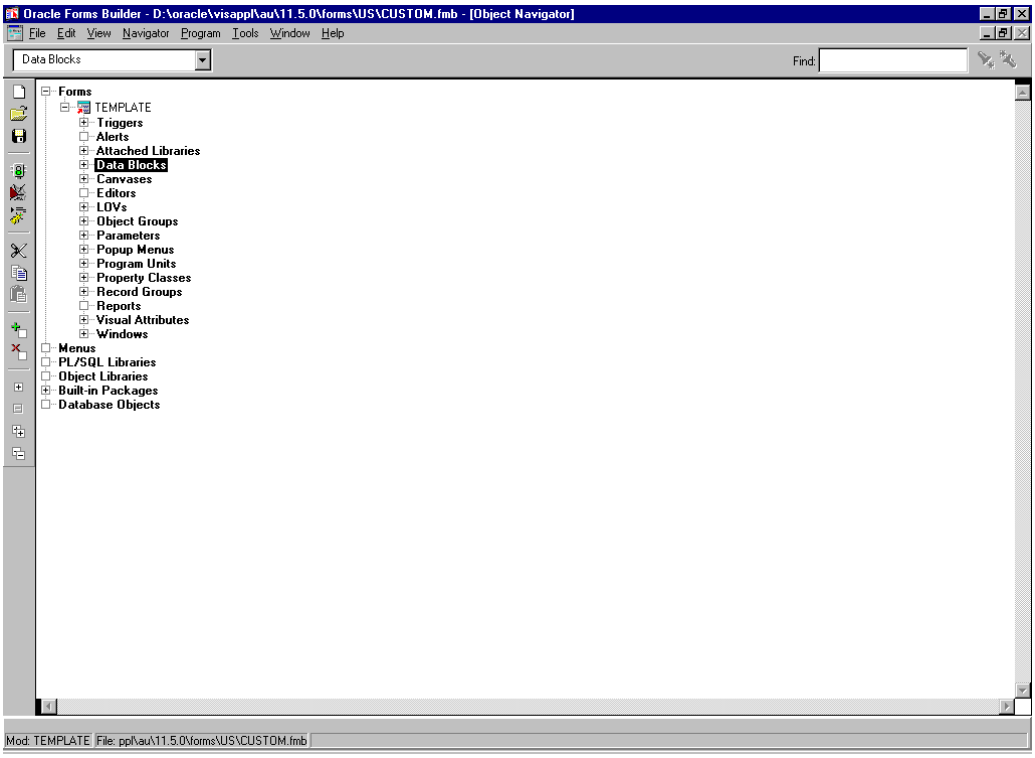
a) The following examples assume you are developing on the forms server. If the forms are being developed on a Windows client refer to Note:73880.1.

Make a copy of the TEMPLATE form and then rename it as the APPLMGR user.

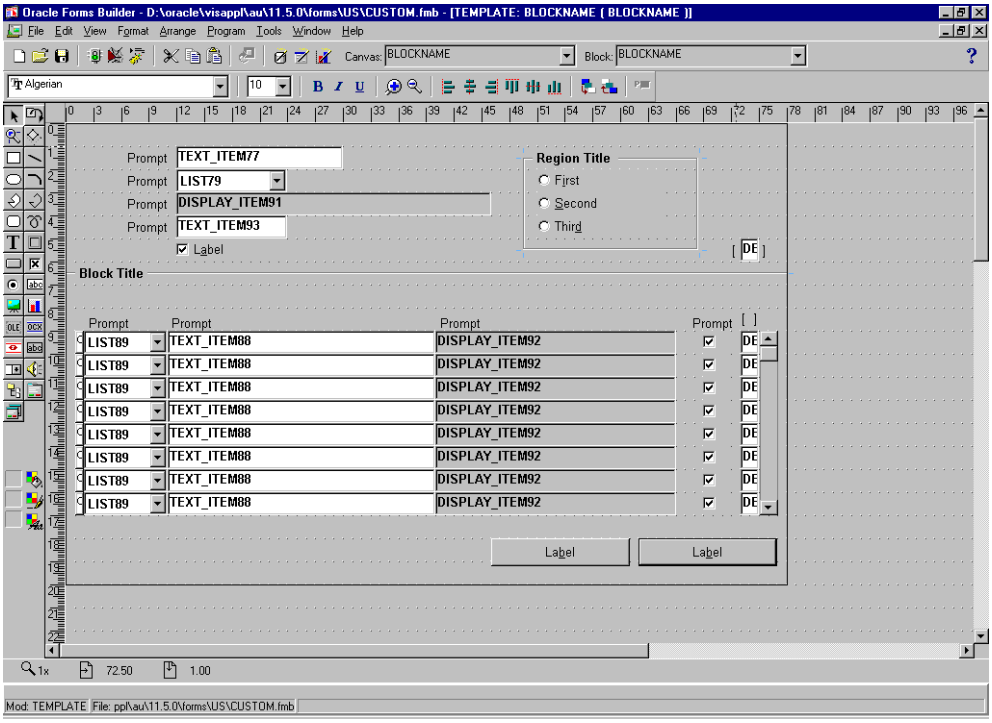
For example:

$ cd $AU_TOP/forms/US
$ cp TEMPLATE.frm CUSTOM.frm

b) In the forms designer attach any additional libraries to your custom form. The TEMPLATE form already has the APPCORE, APPDAYPK, FNDSQF, GLOBE and CUSTOM libraries attached. The only Oracle Applications library that you should modify is the CUSTOM library. All the libraries need to reside in directory $AU_TOP/resource and make sure that your FORM60_PATH includes $AU_TOP/resource so that your form can find the libraries, $AU_TOP/forms/US to locate forms, $AU_TOP/plsql and $ORACLE_HOME/forms60.



c) Modify the form as desired following the development standards. You will be setting the properties of container and widget objects, creating window layout, coding table handler, item handler, event handler and code logic, enabling querying behavior, coding messaging, adding flexfield logic and coding any other appropriate logic.



Refer to "Oracle Applications Developers Guide Release 11i, Volume 1" page 1-17.

d) Generate the form on the forms server as the APPLMGR user. Make sure that the $FORMS60_PATH is set and that the current directory is $AU_TOP/forms/US.

F60gen userid=apps/appspword module=<form name>.fmb output_file=<schema_top>/forms/<language>/<form name>.fmx module_type=form batch=no compile_all=special

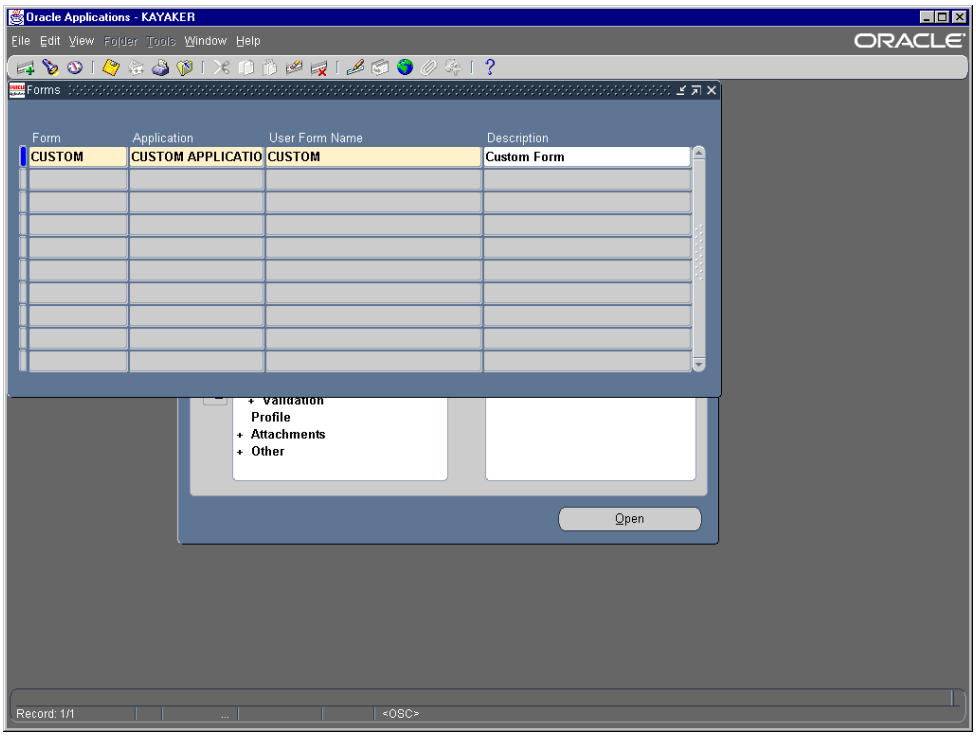
For example:

$ f60gen userid=apps/apps module=TEMP.fmb output_file=/u02/oracle/viappl/XXCUS/forms/US/TEMP.fmx module_type=form batch=no compile_all=special

where form name=TEMP,schema_top=/u02/oracle/visappl/XXCUS and language=US

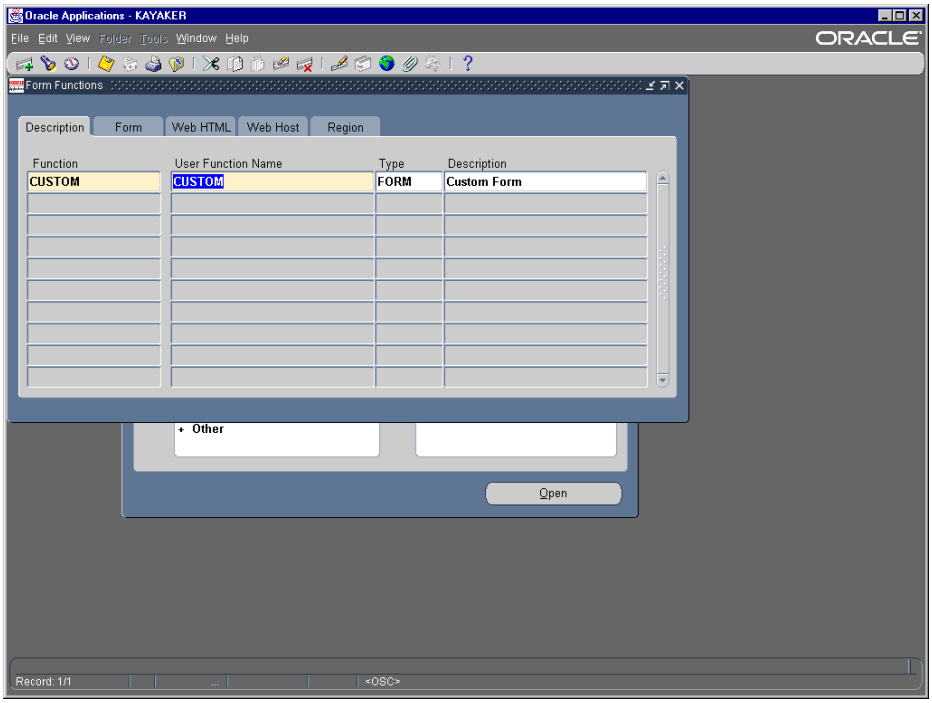
e) Test the form by itself.

f) Register your form. Log into Applications as the Application Developer responsibility and navigate to Application → Form.

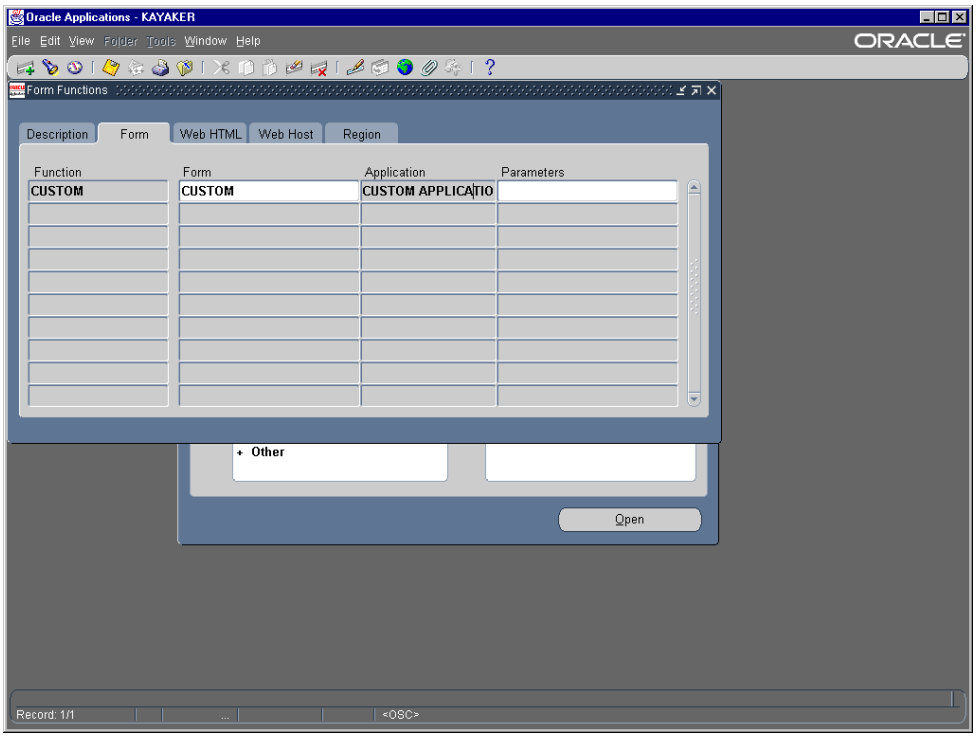


Refer to "Oracle Applications Developers Guide Release 11i, Volume 1" page 11-20.

g) Register the form as a function. If needed register subfunctions per the functionality that you require. Log into Application as the Application Developer responsibility and navigate to Application → Function.

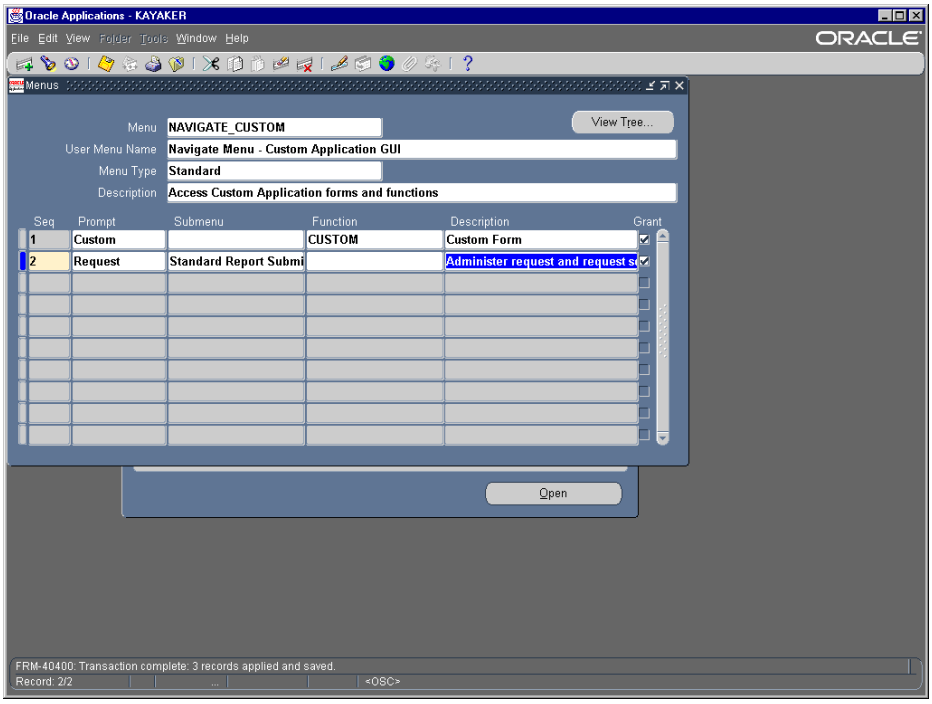


Now click on the "Form" tab.



Refer to "Oracle Applications Developers Guide Release 11i, Volume 1" page 11-22.

h) Add your custom functions (forms and subfunctions) to an existing menu or create a new one. The menu will be tied to a responsibility. Log into Applications as the Application Developer responsibility and navigate to Application → Menu.

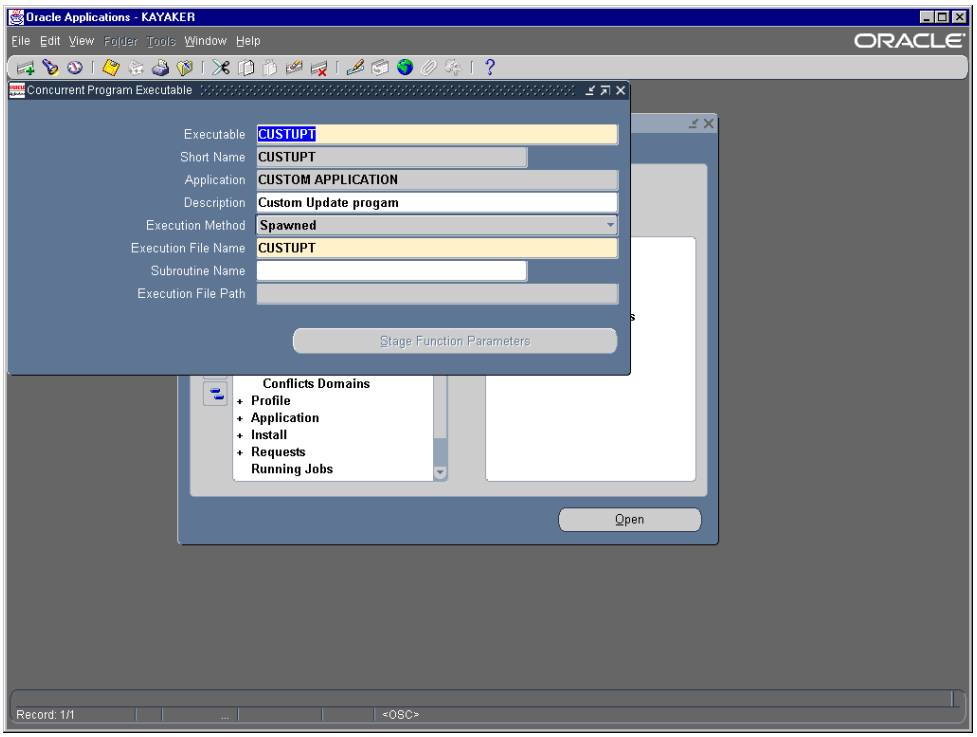


Refer to "Oracle Applications Developers Guide Release 11i, Volume 1" page 11-26.

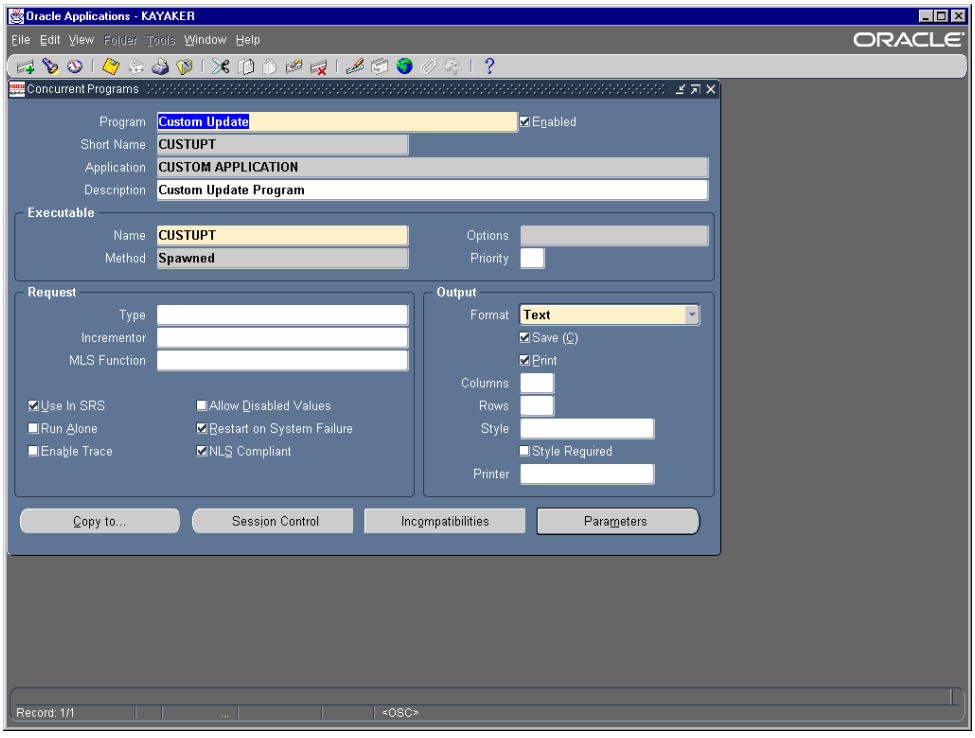
11) Set up your concurrent processing for your Custom Schema.

a) Write the concurrent program execution file and place it in the appropriate directory. You can use a variety of methods such as C, Pro*C, SQL*plus, PL/SQL, Oracle Reports or a host language program (a shell script).

b) Define the concurrent program executable with the Oracle Application Object Library. This links the execution file and the method used to execute it with a defined concurrent program. Log into Applications as the System Administrator and navigate to Concurrent → Program → Executable.



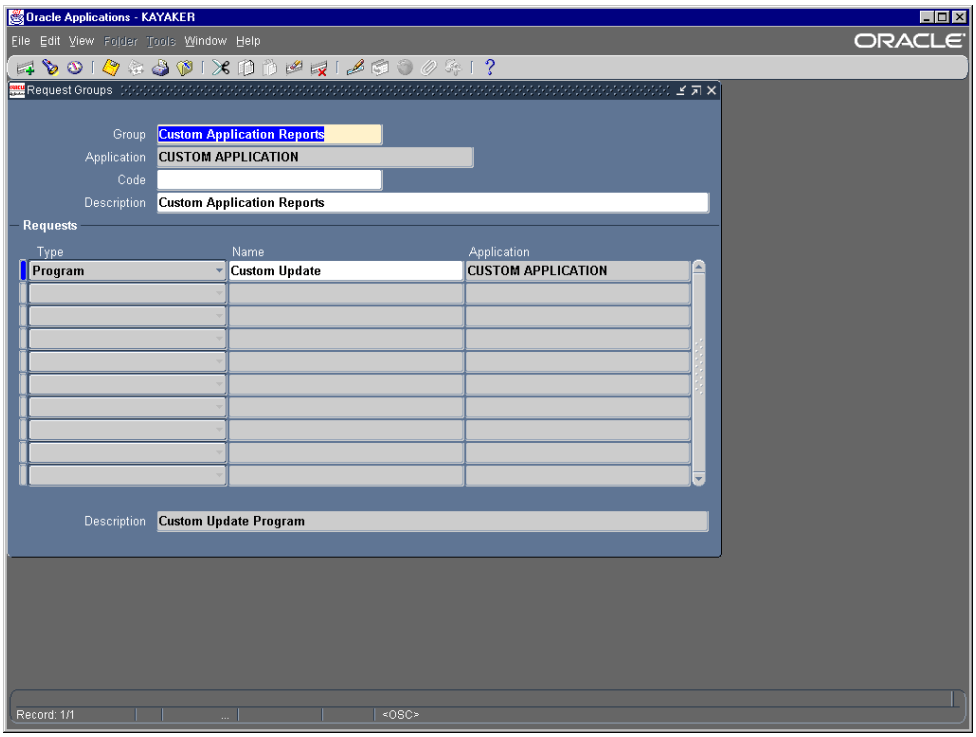
c) Define the concurrent program with the Oracle Application Object Library along with any program parameters or any incompatibilities. Log into Applications as the System Administrator and navigate to Concurrent → Program → Executable.



d) Add the request functionality for your concurrent program. The program can be called from the run reports form, from a trigger within an application form or from a Pro*C program. To use the run reports form just add the submit request window to your custom menu so that you have access to the Standard Request Submission interface (SRS).

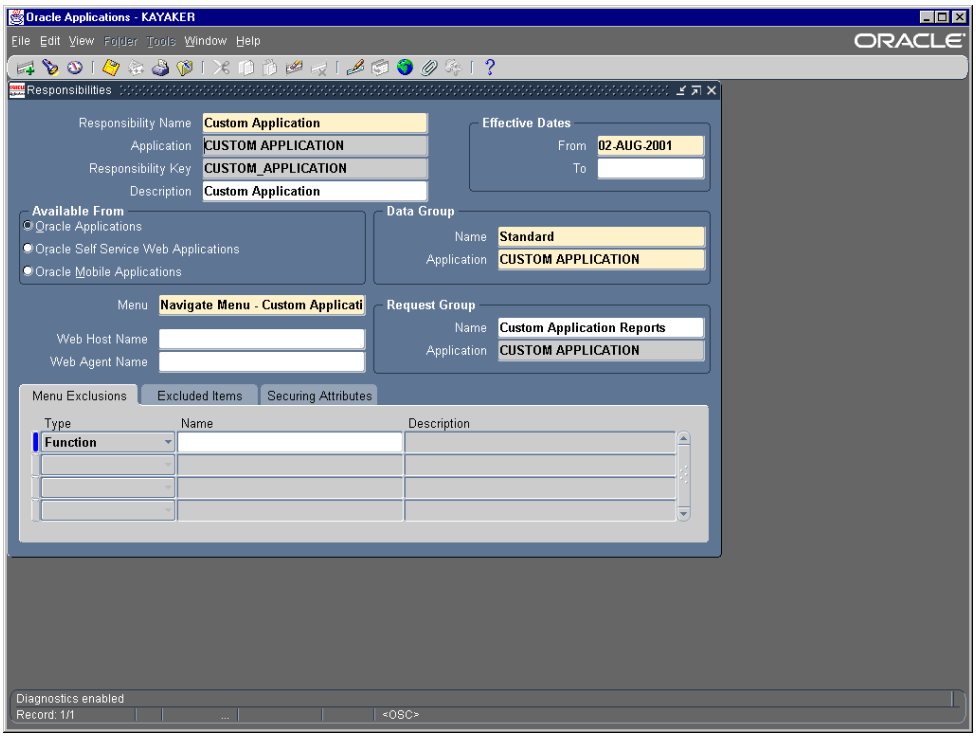
Refer to "Oracle Applications Developers Guide Release 11, Volume 2" page 15-2.

e) Create a custom request group that will hold your custom and/or standard requests for your custom responsibility. Log into Applications as the System Administrator and navigate to Security → Responsibility → Request.



Refer to "Oracle Applications System Administrator's Guide, Release 11i " page 4-58.

f) Create a custom responsibility for your custom schema. Log into Applications as the System Administrator and navigate to Security → Responsibility → Define. Be sure to add the responsibility to a user.



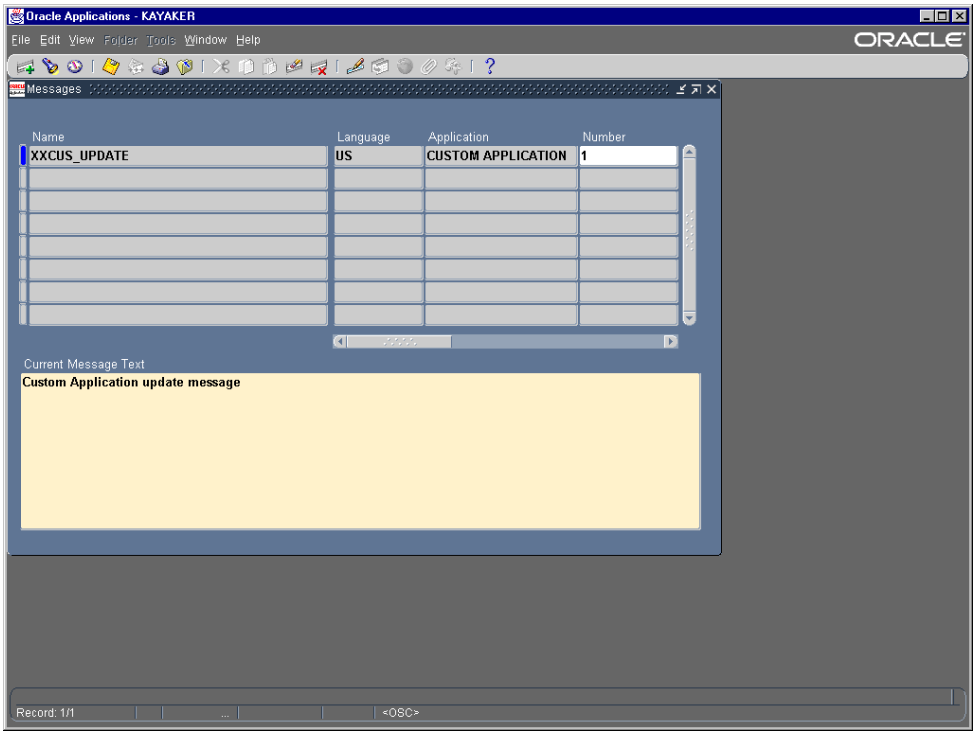
Refer to "Oracle Applications System Administrator's Guide, Release 11i " page 2-9.

g) Test the menu and forms from within Applications.

12) Create your message dictionary.

a) Make sure you have a message (mesg) subdirectory under your custom applications basepath.

b) Define your messages following the message standards. Login as the Application Developer and navigate to Application → Messages.



c) Create the message files. From the operating system run the Message Dictionary Generator program (FNDMDGEN) as the APPLMGR user.

FNDMDGEN apps/appspword 0 Y <Language> <Application Short Name> DB_TO_RUNTIME

For example:

$ FNDMDGEN apps/apps 0 Y US XXCUS DB_TO_RUNTIME

where Language=US and Application Short Name=XXCUS

The FNDMDGEN program will generate a message file named <Language>.msb in place it in the custom applications basepath mesg directory.

d) If needed make a copy of the generated file (located on the server) and transfer the copy to the appropriate mesg directory for the custom application on the client system.

e) Code the logic to set up messages. You will use calls to the FND_MESSAGE package to retrieve and set up messages for display.

f) Code the logic to display messages. You will either display the message to a user on the client or write it to a file on the server. You will use calls to the FND_MESSAGE package to display messages.

Refer to "Oracle Applications Developers Guide Release 11i, Volume 1" page 12-2.

13) Build the online help for your custom applications.

a) Make sure that your custom forms refer to your custom application short name in the call to the FND_STANDARD.FORM_INFO routine in the PRE-FORM trigger. Open your custom form and navigate to Triggers → PRE-FORM.

FND_STANDARD.FORM_INFO ('$Revision: <Number>', '<Form Name>','<Application Short Name>','$Date: <YY/MM/DD HH24:MI:SS>','$Author: <Developers Name>');

For example:

FND_STANDARD.FORM_INFO ('$Revision: 115.1', 'CUSTFORM', 'XXCUS', '$Date: 01/08/15 14:01:01', '$Author: A. Developer');

Where form name=CUSTFORM and application short name=XXCUS

Refer to "Oracle Applications Developers Guide, Volume 2 Release 11i" page 27-27.

b) In the directory <Custom Schema Top>/help/<LANGUAGE> create your HTML help files using an HTML editor. Include the HTML anchor tags of the form near the beginning of the help file so that the form can call the help file.

<A NAME="form_name.window_name"></A>

For example:

<A NAME="custform.blockname"></A>

Where form name=custform and the window name=blockname

Refer to "Oracle Applications Developers Guide, Volume 2 Release 11i" page 27-27.

c) Upload the help file into the database using the FNDGFU utility.

FNDGFU <apps/appspword> 0 Y PROGRAM_NAME=FND_HELP PROGRAM_TAG=<application>:<custom level> CONTENT_TYPE=<mime_type> LANGUAGE=<language code> <filename>

For customizations use 100 or above for the custom_level parameter

For example:
FNDGFU apps/apps 0 Y PROGRAM_NAME=FND_HELP PROGRAM_TAG=XXCUS:100 CONTENT_TYPE=text/html LANGUAGE=US CUSTHELP01.htm

Where application=XXCUS

Refer to " Oracle Applications System Administrator's Guide, Release 11i " page 7-2.

d) Link your help files if needed using hypertext links. The links allow you to link to other help files in other applications, link to other help files within the custom application, link within the same help file and link to multiple topics and files.

For example to link to another help file:

For info on Custom Apps, see <A HREF="@custapps#custapps">Info on Custom App</A>.

Refer to " Oracle Applications System Administrator's Guide, Release 11i " page 7-5.

e) Update the search index after uploading the custom help files.

sqlplus <apps/appspassword> @$FND_TOP/sql/aflobbld.sql

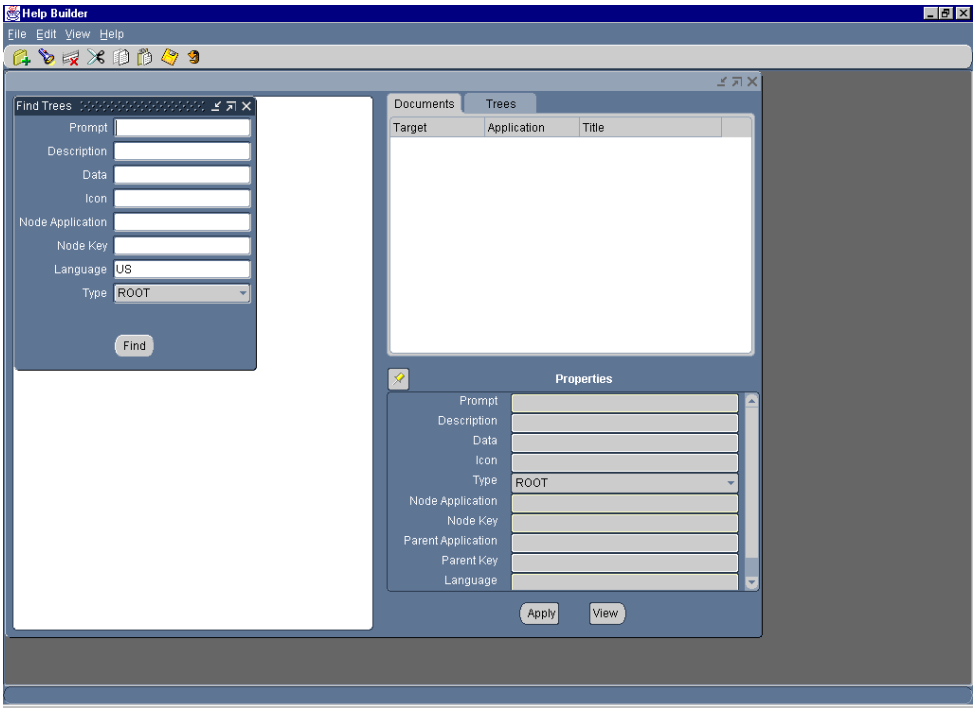
Refer to " Oracle Applications System Administrator's Guide, Release 11i " page 7-9.

f) Create a help navigation tree for the custom application if needed using the Help Builder applet. Trees are composed of root, branch and leaf nodes.

Help Builder Tasks
==========

Open a tree for editing
Add new help files to a tree
Add new nodes to a tree
Add nodes from another tree to a tree
Change the organization of a tree
Create a new navigation tree

To access the Help Builder navigate to Self Service Web Applications > System Administration > Help Builder



Refer to " Oracle Applications System Administrator's Guide, Release 11i " page 7-10 and "Oracle Applications Developers Guide, Volume 2 Release 11i" page 27-28.