



In this section the Developer will learn ...

- How to call another form module or a Report from Forms 9i.



Although Forms modules can be run independently of each other, it is normally desirable for modules to be grouped together to either make a logical workflow or to create a 'single login' system.

Calling other modules can help reduce the size of individual modules as well as create 'generic' modules which can be modified according to the value of the parameters passed to them.

In some systems a controlling Form module is created which gives users a 'menu' of all other modules available.



A Developer may need to call another module for a variety of reasons ...

- To replace the current module
- To display more details relevant to the current module without inhibiting input in the current module
- To display more details relevant to the current module greying out the current module until the called form is exited.
- To display graphical representation of data
- To create a report output of the current data
- To run independently of the calling module



Calling Modules in Forms 9i - Types of Calls

Ideally when calling a module, it should reside in the same directory as the calling form, otherwise the path will need to be included in the call.

To call a form from a form, the following Built-ins can be used...

- OPEN_FORM
- CALL_FORM
- NEW_FORM

To call a Report module

- RUN_REPORT_OBJECT



Open_Form can be used to create multiple-form applications.

A call from one form to another keeps the calling form active, but will change the focus to the called form. The calling form will remain in the background and can be returned to by either closing the called form or by selecting its window.

```
open_form('c:\course\stock')
```

The above is the simplest call to another Form using Open_Form.



Calling Modules in Forms 9i - Open_Form

The screenshot displays three overlapping Oracle Forms 9i windows, each titled 'WINDOW1'. The top window shows a list of items with columns 'Number' and 'Name'. The middle window shows a list with columns 'Number', 'Name', and 'Type'. The bottom window shows a list with columns 'Stock', 'Section', 'Description', 'Price', and 'In Stock'. A watermark 'OracleAS Forms Service' is visible in the background.

Number	Name
1	DOORS AND LOCKS
2	BATHROOM FITTINGS
3	BULBS AND BLOOMS
4	TOOLS AND MOWERS
5	TREES
6	CEMENT AND SAND
7	HAMMERS AND NAILS
8	IRONMONGRY

Number	Name	Type
1	DOORS AND LOCKS	HARDWARE
2	BATHROOM FITTINGS	HARDWARE
3	BULBS AND BLOOMS	GARDENING
4	TOOLS AND MOWERS	GARDENING
5	TREES	GARDENING

Stock	Section	Description	Price	In Stock
1	1	BRASS HANDLE	12.99	100
9	1	YALE LOCK	9.99	15
10	1	BRASS KNOCKER	15.99	10
25	1	HINGE PACK (LH)	6.99	10
26	1	HINGE PACK (RH)	6.99	0



There are several settings which can be included ...

Activate_Mode - Specifies if the called form is to receive the focus. Can be `ACTIVATE` or `NO_ACTIVATE`. The default is `ACTIVATE`.

Session_Mode - Specifies if a new database session is to be opened using the current connection information, enabling more than one transaction to be handled at the same time. Can be `NO_SESSION` or `SESSION`. The default is `NO_SESSION`.

Paramlist_Name or *Paramlist ID* - The name or ID of the parameter list if any, to be supplied.



Calling Modules in Forms 9i - Open_Form

This example calls a form named `stock_details`, make it active, and supply a parameter list called `paramlist1` without opening a new database session...

```
open_form('stock_details'  
         ,ACTIVATE  
         ,NO_SESSION  
         ,'paramlist1');
```



Call_Form can be used to call one form from another, the focus transfers to the new Form, until the called Form is closed the originating Form cannot be returned to.

Calling several forms using Call_Form creates a '*call form stack*' structure, this is due to the modal forms forming a hierarchy.

The simplest call to Call_Form is to specify the name of the form to open...

```
call_form('c:\course\sections');
```



Calling Modules in Forms 9i - Call_Form

The image shows two Oracle Forms 9i windows. The left window, titled 'WINDOW1', contains a list of items with the following data:

Number	Name	Type
1	DOORS AND LOCKS	HARDWARE
2	BATHROOM FITTINGS	HARDWARE
3	BULBS AND BLOOMS	GARDENING
4	TOOLS AND MOWERS	GARDENING
5	TREES	GARDENING
6	CEMENT AND SAND	BUILDING
7	HAMMERS AND NAILS	HARDWARE
8	IRONMONGRY	HARDWARE

The right window, also titled 'WINDOW1', displays a table with the following columns: Stock, Section, Description, Price, and In Stock. The table is currently empty.

Both windows have a menu bar with 'Action', 'Edit', 'Query', 'Block', 'Record', 'Field', 'Help', and 'Window'. The status bar at the bottom of each window shows 'Record: 1/?'.



There are a number of optional parameters...

Display - Specifies if the calling form is hidden when the called form is displayed. `No_Hide` displays calling form. The default is `Hide`. Can be `Hide` or `No_Hide`.

Switch_Menu - Specifies if the calling form's default menu should be replaced by the called form's default menu. The default is `No_Replace`. Can be `No_Replace` or `Do_Replace`.

Query_Mode - Specifies if the called form should be displayed in Query-Only mode. The default is `No_Query_Only`. Can be `No_Query_Only` or `Query_Only`..



Data_Mode - Specifies that called forms which have an identical attached library to the calling form can share data between the forms. The default is `NO_SHARE_LIBRARY_DATA`. Can be `NO_SHARE_LIBRARY_DATA` or `SHARE_LIBRARY_DATA`.

Paramlist_Name or *Paramlist ID* - The name or ID of the parameter list if any, to be supplied.

To call a form named `stock_details`, in query mode, and supply a parameter list called `paramlist1` ...

```
CALL_FORM('stock_details'  
        ,no_hide  
        ,no_replace  
        ,no_query_only  
        ,<parameter list>);
```



Using `New_Form` closes the calling Form and displays the new Form in its place.

This option tends to be used when the calling Form is no longer needed in the processing, or when the Developer wishes to give the user an uncluttered system.

As with the previous examples, the simplest call is thus ...

```
new_form('c:\course\stock');
```



The optional parameters that can be supplied are..

Rollback_Mode - Specifies if a rollback is to be performed on the form, and if so, up until what point. Can be *To_Savepoint*, *No_Rollback*, or *Full_Rollback*. The default is *To_Savepoint*.

Query_Mode - Specifies if the called form should be displayed in Query-Only mode or not. Can be *No_Query_Only* or *Query_Only*.

Data_Mode - Specifies if a called form that has an identical attached library to the calling form can share data between the forms. Can be *No_Share_Library_Data* or *Share_Library_Data*. The default is *No_Share_Library_Data*.



Calling Modules in Forms 9i - New_Form

Paramlist_Name or *Paramlist ID* - The name or ID of the parameter list if any, to be supplied.

To call a form named `stock_details` in query only mode...

```
new_form('stock_details'  
        ,query_only);
```



Calling Modules in Forms 9i – Run_Report_Object

Having registered a Report in the Form module, the Developer can call it using the Run_Report_Object, in all examples the Developer will use the registered name of the Report not its physical name on the filesystem.

The simplest call to a Report is as follows ...

```
run_report_object('stock_report');
```



Another method is to check that the Report is already registered using the following ...

declare

l_report REPORT_OBJECT;

l_run VARCHAR2(100);

begin

l_report := find_report_object('stock_report');

l_run := RUN_REPORT_OBJECT(*l_report*);

end;



The passing of Globals between Forms modules using Run_Report_Object doesn't work, therefore Parameter lists must be used.

Note that in the case of Forms modules, all Parameters passed must already exist in the called form, however, not all Parameters which exist need to be passed.

To append a Parameter to a call, use either the Parameter name or its ID

Calling Modules in Forms 9i – Run_Report_Object



Here the Report is sent a parameter called l_location ...

```
declare
    l_paraml  ParamList;
begin
    l_paraml := Create_Parameter_List('input_values');

    add_parameter(l_paraml
                  , 'l_location'
                  , Text_Parameter
                  , :store_locations.sl_location_id);

    Run_Report_Object('stock_report'
                     , l_paraml);
```



In the normal course of events closing a called form and returning to the calling form is taken care of by the normal Exit option on the menu.

To simulate this programmatically, use either the EXIT_FORM or CLOSE_FORM built-ins.



Exits the current form, committing or rolling back if necessary.

If the form is in Enter-Query mode, the form is not exited, instead it exits the Enter-Query mode.

```
exit_form(commit_mode  
          , rollback);
```



The following optional parameters can be specified

Commit_Mode

- *Ask_Commit* prompts the user to commit the form.
- *Do_Commit* performs a commit without prompting the user.
- *No_Commit* validates and exits it without committing.
- *No_Validate* exits the form without validating or committing.

The default is *Ask_Commit*.

Rollback - Specifies if a rollback is to be performed on the form, and if so, up until what point. Can be *To_Savepoint*, *No_Rollback*, or *Full_Rollback*. The default is *To_Savepoint*.

Calling Modules in Forms 9i – Exit_Form



To exit the current form without committing ...

```
exit_form(no_commit);
```

Calling Modules in Forms 9i – Close_Form



This function allows the closure of a module regardless whether it is the current (in focus) or otherwise ...

The form name can either be the string or the id of the Form.

```
close_form('sales_stock');
```



There are certain problems with calling modules, in particular when attempting to return values to the calling module.

Basically the call to the new module should be the last line of processing in the Program Unit, this is because the processing will not return to where it was called from, in other words calling another Module is almost like raising a `Form_Trigger_Error`.

In some circumstances this can cause problems because a Developer may wish to receive information back from the called module.

Calling Modules in Forms 9i – When-Window-Activated



Developers may need to create a When-Window-Activated trigger, this will detect when the focus has returned to the calling module and any reading of Globals etc should take place here.



Web.Show_Document allows the Developer to display web sites from Forms

```
web.show_document('http://www.seercomputing.com','_top');
```

_top signifies the position of the new Web page, in this case replacing the current, other options are

_self

_parent

_blank