Ústav informatiky a výpočetní techniky FEI VUT v Brně

Jaroslav Zendulka

Zabudované triggery a funkce Oracle Forms 4.5

Učební pomůcka

Tento manuál je výtahem originálního textu v angličtině z manuálu Forms Reference Manual, Release 4.5 firmy Oracle. Jeho účelem je poskytnout ve zhuštěné podobě přehled zabudovaných triggerů a funkcí prostředí Forms Designer. U jednotlivých kapitol i v textu jsou odkazy na kapitoly v originálu. Materiál byl vytvořen jako učební pomůcka pro řešení projektů předmětu Databázové systémy na oboru Výpočetní technika a informatika.

Prosinec 1998

Obsah

1. Triggers (CHAPTER 2.) 9
1.1 About Triggers and Processes 9
1.2 SQL Statements in Trigger Text 9
1.3 Trigger Tables 9
1.3.1 Block Processing Triggers 9
1.3.1.1 When-Clear-Block 9
1.3.1.2 When-Create-Record 9
1.3.1.3 When-Database-Record 9
1.3.1.4 When-Remove-Record 10
1.3.2 Interface Event Triggers 10
1.3.2.1 When-Button-Pressed 10
1.3.2.2 When-Checkbox-Changed 10
1.3.2.3 When-Custom-Item-Event 10
1.3.2.4 When-Image-Activated 10
1.3.2.5 When-Image-Pressed 10
1.3.2.6 When-List-Changed 10
1.3.2.7 When-Mouse-Click 10
1.3.2.8 When-Mouse-DoubleClick 10
1.3.2.9 When-Mouse-Down 10
1.3.2.10 When-Mouse-Enter 11
1.3.2.11 When-Mouse-Leave 11
1.3.2.12 When-Mouse-Move 11
1.3.2.13 When-Mouse-Up 11
1.3.2.14 When-Radio-Changed 11
1.3.2.15 When-Timer-Expired 11
1.3.2.16 When-Window-Activated 11
1.3.2.17 When-Window-Closed 11
1.3.2.18 When-Window-Deactivated 11
1.3.2.19 When-Window-Resized 11
1.3.3 Key Triggers 12
1.3.3.1 Function Key 12
1.3.3.2 Key-Fn 12
1.3.3.3 Key-Others 12
1.3.4 Master-Detail Triggers 12
1.3.4.1 On-Check-Delete-Master 12
1.3.4.2 On-Clear-Details 12
1.3.4.3 On-Populate-Details 12
1.3.5 Message-Handling Triggers 13
1.3.5.1 On-Error 13
1.3.5.2 On-Message 13
1.3.6 Navigational Triggers 13
1.3.6.1 Post-Block 13
1.3.6.2 Post-Form 13
1.3.6.3 Post-Record 13
1.3.6.4 Post-Text-Item 13
1.3.6.5 Pre-Block 13
1.3.6.6 Pre-Form 13
1.3.6.7 Pre-Record 13
1.3.6.8 Pre-Text-Item 13
1.3.6.9 When-Form-Navigate 13
1.3.6.10 When-New-Block-Instance 13
1.3.6.11 When-New-Form-Instance 13
1.3.6.12 When-New-Item-Instance 13
1.3.6.13 When-New-Record-Instance 14
1.3.7 Query-Time Triggers 14
1.3.7.1 Post-Query 14
1.3.7.2 Pre-Query 14
1.3.8 Transactional Triggers 14

1.3.8.1 On-Check-Unique 14
1.3.8.2 On-Close 14
1.3.8.3 On-Column-Security 14
1.3.8.4 On-Commit 14
1.3.8.5 On-Count 14
1.3.8.6 On-Delete 14
1.3.8.7 On-Fetch 14
1.3.8.8 On-Insert 14
1.3.8.9 On-Lock 15
1.3.8.10 On-Logon 15
1.3.8.11 On-Logout 15
1.3.8.12 On-Rollback 15
1.3.8.13 On-Savepoint 15
1.3.8.14 On-Select 15
1.3.8.15 On-Sequence-Number 15
1.3.8.16 On-Update 15
1.3.8.17 Post-Change 15
1.3.8.18 Post-Database-Commit 15
1.3.8.19 Post-Delete 15
1.3.8.20 Post-Forms-Commit 15
1.3.8.21 Post-Insert 16
1.3.8.22 Post-Logon 16
1.3.8.23 Post-Logout 16
1.3.8.24 Post-Select 16
1.3.8.25 Post-Update 16
1.3.8.26 Pre-Commit 16
1.3.8.27 Pre-Delete 16
1.3.8.28 Pre-Logon 16
1.3.8.29 Pre-Logout 16
1.3.8.30 Pre-Select 16
1.3.8.31 Pre-Update 16
1.3.9 Validation Triggers 16
1.3.9.1 When-Validate-Item 16
1.3.9.2 When-Validate-Record 17
1.3.9.3 User-Named Trigger 17
2. System Variables (CHAPTER 4.) 18
2.1 About System Variables 18
2.1.1 List of system variables 18
2.1.1.1 SYSTEM.BLOCK_STATUS 18
2.1.1.2 SYSTEM.COORDINATION_OPERATION 18
2.1.1.3 SYSTEM.CURRENT_BLOCK 18
2.1.1.4 SYSTEM.CURRENT_DATETIME 18
2.1.1.5 SYSTEM.CURRENT_FORM 18
2.1.1.6 SYSTEM.CURRENT_ITEM 18
2.1.1.7 SYSTEM.CURRENT_VALUE 19
2.1.1.8 SYSTEM.CURSOR_BLOCK 19
2.1.1.9 SYSTEM.CURSOR_ITEM 19
2.1.1.10 SYSTEM.CURSOR_RECORD 19
2.1.1.11 SYSTEM.CURSOR_VALUE 19
2.1.1.12 19
2.1.1.13 SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS 19
2.1.1.14 SYSTEM.DATE_THRESHOLD* 19
2.1.1.15 SYSTEM.EFFECTIVE_DATE* 19
2.1.1.16 SYSTEM.EVENT_WINDOW 19
2.1.1.17 SYSTEM.FORM_STATUS 20
2.1.1.18 SYSTEM.LAST_FORM 20
2.1.1.19 SYSTEM.LAST_QUERY 20
2.1.1.20 SYSTEM.LAST_RECORD 20
2.1.1.21 SYSTEM.MASTER_BLOCK 20
2.1.1.22 SYSTEM.MESSAGE_LEVEL* 20

2.1.1.23 SYSTEM.MODE 20
2.1.1.24 SYSTEM.MOUSE_BUTTON_PRESSED 20
2.1.1.25 SYSTEM.MOUSE_BUTTON_SHIFT_STATE 21
2.1.1.26 SYSTEM.MOUSE_CANVAS 21
2.1.1.27 SYSTEM.MOUSE_RECORD_OFFSET 21
2.1.1.28 SYSTEM.MOUSE_X_POS 21
2.1.1.29 SYSTEM.MOUSE_Y_POS 21
2.1.1.30 SYSTEM.RECORD_STATUS 21
2.1.1.31 SYSTEM.SUPPRESS_WORKING* 21
2.1.1.32 SYSTEM.TRIGGER_BLOCK 22
2.1.1.33 SYSTEM.TRIGGER_ITEM 22
2.1.1.34 SYSTEM.TRIGGER_RECORD 22
2.1.2 Date and Time System Default Values 22
2.1.2.1 $$DATE$$ 22
2.1.2.2 $$DATETIME$$ 22
2.1.2.3 $$DBDATE$$ 22
2.1.2.4 $$DBDATETIME$$ 23
2.1.2.5 $$DBTIME$$ 23
2.1.2.6 $$TIME$$ 23
2.1.3 Local Variables 23
3. Built-in Subprograms (CHAPTER 3.) 24
3.1 Overview 24
3.1.1 Syntax 24
3.1.2 Named Parameters 24
3.1.3 Object IDs 24
3.1.4 Form Coordinate Units 24
3.1.5 Uppercase Return Values 25
3.1.6 Restricted Built-In Subprograms 25
3.1.7 Constants 25
3.2 Built-in Subprograms Tables 25
3.2.1 Alert Built-ins 25
3.2.1.1 FIND_ALERT 25
3.2.1.2 ID_NULL 26
3.2.1.3 SET_ALERT_BUTTON_PROPERTY 26
3.2.1.4 SET_ALERT_PROPERTY 26
3.2.1.5 SHOW_ALERT 26
3.2.2 Application Built-ins 26
3.2.2.1 DO_KEY 26
3.2.2.2 GET_APPLICATION_PROPERTY 26
3.2.2.3 HOST 27
3.2.2.4 PAUSE 27
3.2.2.5 SET_APPLICATION_PROPERTY 27
3.2.2.6 USER_EXIT 27
3.2.3 Block Built-ins 27
3.2.3.1 BLOCK_MENU 27
3.2.3.2 CLEAR_BLOCK 27
3.2.3.3 FIND_BLOCK 27
3.2.3.4 GET_BLOCK_PROPERTY 28
3.2.3.5 GO_BLOCK 28
3.2.3.6 ID_NULL 28
3.2.3.7 NEXT_BLOCK 28
3.2.3.8 PREVIOUS_BLOCK 28
3.2.3.9 SET_BLOCK_PROPERTY 28
3.2.4 Canvas and View Built-ins 28
3.2.4.1 FIND_CANVAS 28
3.2.4.2 FIND_VIEW 29
3.2.4.3 GET_CANVAS_PROPERTY 29
3.2.4.4 GET_VIEW_PROPERTY 29
3.2.4.5 HIDE_VIEW 29
3.2.4.6 ID_NULL 29

3.2.4.7 PRINT 29
3.2.4.8 SCROLL_VIEW 29
3.2.4.9 SET_CANVAS_PROPERTY 30
3.2.4.10 SET_VIEW_PROPERTY 30
3.2.4.11 SHOW_VIEW 30
3.2.5 Form Built-ins 30
3.2.5.1 BELL 30
3.2.5.2 BREAK 30
3.2.5.3 CALL_FORM 31
3.2.5.4 CALL_INPUT 31
3.2.5.5 CLEAR_FORM 31
3.2.5.6 COMMIT_FORM 31
3.2.5.7 DEBUG_MODE 31
3.2.5.8 ENTER 32
3.2.5.9 ERASE 32
3.2.5.10 EXECUTE_TRIGGER 32
3.2.5.11 EXIT_FORM 32
3.2.5.12 FIND_FORM 33
3.2.5.13 FORM_FAILURE 33
3.2.5.14 FORM_FATAL 33
3.2.5.15 FORM_SUCCESS 33
3.2.5.16 GET_FORM_PROPERTY 34
3.2.5.17 HELP 34
3.2.5.18 ID_NULL 34
3.2.5.19 NEW_FORM 34
3.2.5.20 OPEN_FORM 34
3.2.5.21 POST 35
3.2.5.22 REDISPLAY 35
3.2.5.23 REPLACE_MENU 35
3.2.5.24 SET_FORM_PROPERTY 35
3.2.5.25 SHOW_KEYS 35
3.2.5.26 SHOW_MENU 35
3.2.5.27 SYNCHRONIZE 35
3.2.6 Item Built-ins 36
3.2.6.1 CHECKBOX_CHECKED 36
3.2.6.2 CLEAR_EOL 36
3.2.6.3 CLEAR_ITEM 36
3.2.6.4 CONVERT_OTHER_VALUE 36
3.2.6.5 COPY 36
3.2.6.6 COPY_REGION 36
3.2.6.7 CUT_REGION 37
3.2.6.8 DEFAULT_VALUE 37
3.2.6.9 DISPLAY_ITEM 37
3.2.6.10 DUPLICATE_ITEM 37
3.2.6.11 EDIT_TEXTITEM 37
3.2.6.12 FIND_ITEM 38
3.2.6.13 GET_ITEM_PROPERTY 38
3.2.6.14 GET_RADIO_BUTTON_PROPERTY 38
3.2.6.15 GO_ITEM 38
3.2.6.16 ID_NULL 38
3.2.6.17 IMAGE_ZOOM 38
3.2.6.18 NAME_IN 38
3.2.6.19 NEXT_ITEM 39
3.2.6.20 NEXT_KEY 39
3.2.6.21 PASTE_REGION 39
3.2.6.22 PREVIOUS_ITEM 39
3.2.6.23 READ_IMAGE_FILE 40
3.2.6.24 SELECT_ALL 40
3.2.6.25 SET_ITEM_PROPERTY 40
3.2.6.26 SET_RADIO_BUTTON_PROPERTY 40

3.2.6.27 WRITE_IMAGE_FILE....40
3.2.7 List Item Built-ins....40
3.2.7.1 ADD_LIST_ELEMENT....40
3.2.7.2 CLEAR_LIST....41
3.2.7.3 DELETE_LIST_ELEMENT....41
3.2.7.4 GET_LIST_ELEMENT_COUNT....41
3.2.7.5 GET_LIST_ELEMENT_LABEL....41
3.2.7.6 GET_LIST_ELEMENT_VALUE....41
3.2.7.7 POPULATE_LIST....41
3.2.7.8 RETRIEVE_LIST....42
3.2.8 Menu Built-ins....42
3.2.8.1 APPLICATION_PARAMETER....42
3.2.8.2 BACKGROUND_MENU....42
3.2.8.3 FIND_MENU_ITEM....42
3.2.8.4 GET_MENU_ITEM_PROPERTY....42
3.2.8.5 HIDE_MENU....42
3.2.8.6 ITEM_ENABLED....43
3.2.8.7 MAIN_MENU....43
3.2.8.8 MENU_CLEAR_FIELD....43
3.2.8.9 MENU_NEXT_FIELD....43
3.2.8.10 MENU_PARAMETER....43
3.2.8.11 MENU_PREVIOUS_FIELD....43
3.2.8.12 MENU_REDISPLAY....43
3.2.8.13 MENU_SHOW_KEYS....44
3.2.8.14 NEXT_MENU_ITEM....44
3.2.8.15 PREVIOUS_MENU....44
3.2.8.16 PREVIOUS_MENU_ITEM....44
3.2.8.17 QUERY_PARAMETER....44
3.2.8.18 SET_INPUT_FOCUS....44
3.2.8.19 SET_MENU_ITEM_PROPERTY....44
3.2.8.20 SHOW_BACKGROUND_MENU....44
3.2.8.21 TERMINATE....45
3.2.8.22 WHERE_DISPLAY....45
3.2.9 Messages Built-ins....45
3.2.9.1 CLEAR_MESSAGE....45
3.2.9.2 DBMS_ERROR_CODE....45
3.2.9.3 DBMS_ERROR_TEXT....45
3.2.9.4 DISPLAY_ERROR....45
3.2.9.5 ERROR_CODE....45
3.2.9.6 ERROR_TEXT....46
3.2.9.7 ERROR_TYPE....46
3.2.9.8 GET_MESSAGE....46
3.2.9.9 MESSAGE....46
3.2.9.10 MESSAGE_CODE....46
3.2.9.11 MESSAGE_TEXT....47
3.2.9.12 MESSAGE_TYPE....47
3.2.10 Miscellaneous Built-ins....47
3.2.10.1 CREATE_TIMER....47
3.2.10.2 DELETE_TIMER....47
3.2.10.3 FIND_EDITOR....47
3.2.10.4 FIND_LOV....47
3.2.10.5 FIND_TIMER....47
3.2.10.6 GET_LOV_PROPERTY....48
3.2.10.7 ID_NULL....48
3.2.10.8 LIST_VALUES....48
3.2.10.9 SET_LOV_COLUMN_PROPERTY....48
3.2.10.10 SET_LOV_PROPERTY....48
3.2.10.11 SET_TIMER....48
3.2.10.12 SHOW_EDITOR....48
3.2.10.13 SHOW_LOV....48

3.2.10.14 VALIDATE....48
3.2.11 Multiple-form Application Built-ins....48
3.2.11.1 CLOSE_FORM....48
3.2.11.2 GO_FORM....49
3.2.11.3 NEXT_FORM....49
3.2.11.4 OPEN_FORM....49
3.2.11.5 PREVIOUS_FORM....49
3.2.12 OLE Built-ins....49
3.2.13 Parameter List Built-ins....50
3.2.14 Query Built-ins....50
3.2.14.1 ABORT_QUERY....50
3.2.14.2 COUNT_QUERY....50
3.2.14.3 ENTER_QUERY....50
3.2.14.4 EXECUTE_QUERY....50
3.2.15 Record Built-ins....51
3.2.15.1 CHECK_RECORD_UNIQUENESS....51
3.2.15.2 CLEAR_RECORD....51
3.2.15.3 CREATE_QUERIED_RECORD....51
3.2.15.4 CREATE_RECORD....51
3.2.15.5 DELETE_RECORD....51
3.2.15.6 DOWN....52
3.2.15.7 DUPLICATE_RECORD....52
3.2.15.8 FIRST_RECORD....52
3.2.15.9 GENERATE_SEQUENCE_NUMBER....52
3.2.15.10 GET_RECORD_PROPERTY....52
3.2.15.11 GO_RECORD....52
3.2.15.12 INSERT_RECORD....53
3.2.15.13 LAST_RECORD....53
3.2.15.14 LOCK_RECORD....53
3.2.15.15 NEXT_RECORD....53
3.2.15.16 NEXT_SET....53
3.2.15.17 PREVIOUS_RECORD....53
3.2.15.18 SCROLL_DOWN....54
3.2.15.19 SCROLL_UP....54
3.2.15.20 SELECT_RECORDS....54
3.2.15.21 SET_RECORD_PROPERTY....54
3.2.15.22 UP....54
3.2.15.23 UPDATE_RECORD....54
3.2.16 Record Group Built-ins....55
3.2.16.1 ADD_GROUP_COLUMN....55
3.2.16.2 ADD_GROUP_ROW....55
3.2.16.3 CREATE_GROUP....55
3.2.16.4 CREATE_GROUP_FROM_QUERY....55
3.2.16.5 DELETE_GROUP....55
3.2.16.6 DELETE_GROUP_ROW....56
3.2.16.7 FIND_COLUMN....56
3.2.16.8 FIND_GROUP....56
3.2.16.9 GET_GROUP_CHAR_CELL....56
3.2.16.10 GET_GROUP_DATE_CELL....56
3.2.16.11 GET_GROUP_NUMBER_CELL....57
3.2.16.12 GET_GROUP_RECORD_NUMBER....57
3.2.16.13 GET_GROUP_ROW_COUNT....57
3.2.16.14 GET_GROUP_SELECTION....57
3.2.16.15 GET_GROUP_SELECTION_COUNT....57
3.2.16.16 ID_NULL....57
3.2.16.17 POPULATE_GROUP....57
3.2.16.18 POPULATE_GROUP_WITH_QUERY....58
3.2.16.19 RESET_GROUP_SELECTION....58
3.2.16.20 SET_GROUP_CHAR_CELL....58
3.2.16.21 SET_GROUP_DATE_CELL....58

3.2.16.22 SET_GROUP_NUMBER_CELL 58
3.2.16.23 SET_GROUP_SELECTION 59
3.2.16.24 UNSET_GROUP_SELECTION 59
3.2.17 Relation Built-ins 59
3.2.17.1 FIND_RELATION 59
3.2.17.2 GET_RELATION_PROPERTY 59
3.2.17.3 ID_NULL 59
3.2.17.4 SET_RELATION_PROPERTY 59
3.2.18 Transactional Built-ins 60
3.2.18.1 CHECK_RECORD_UNIQUENESS 60
3.2.18.2 DELETE_RECORD 60
3.2.18.3 ENFORCE_COLUMN_SECURITY 60
3.2.18.4 FETCH_RECORDS 60
3.2.18.5 FORMS_DDL 60
3.2.18.6 GENERATE_SEQUENCE_NUMBER 60
3.2.18.7 INSERT_RECORD 60
3.2.18.8 ISSUE_ROLLBACK 60
3.2.18.9 ISSUE_SAVEPOINT 61
3.2.18.10 LOGON 61
3.2.18.11 LOGON_SCREEN 61
3.2.18.12 LOGOUT 61
3.2.18.13 SELECT_RECORDS 61
3.2.18.14 UPDATE_RECORD 61
3.2.19 VBX Control Built-ins 61
3.2.20 Window Built-ins 61
3.2.20.1 FIND_WINDOW 61
3.2.20.2 ID_NULL 62
3.2.20.3 MOVE_WINDOW 62
3.2.20.4 REPLACE_CONTENT_VIEW 62
3.2.20.5 RESIZE_WINDOW 62
3.2.20.6 SHOW_WINDOW 62

1. Triggers (CHAPTER 2.)

1.1 About Triggers and Processes

Triggers are blocks of PL/SQL code that you write to perform specific tasks. There are pre-defined runtime events for which you can create triggers. Trigger names correspond to these events. For ease of description, the terms events and triggers are synonymous in this chapter. In effect, an Oracle Forms trigger is an event-handler written in PL/SQL to augment (or occasionally replace) the default processing behavior.

A process is a series of individual, related events that occurs during a specific Oracle Forms Runform operation. Oracle Forms includes navigational, validation, and database transaction processes. To see a visual representation of Oracle Forms processes showing where each trigger fires, refer to the flowchart for the process named in the "Fires In" section for each trigger. All the flowcharts are in the Oracle Forms Reference Manual, Vol. 2, Chapter 8, "Processing Flow Charts".

1.2 SQL Statements in Trigger Text

The trigger descriptions in this chapter include a section called "Legal Commands." This section lists the types of statements that are valid in the indicated trigger type, including:

- restricted built-in subprograms
- unrestricted built-in subprograms
- SELECT statements
- Data Manipulation Language (DML) statements

Restricted built-in subprograms initiate navigation. They include built-ins that move the input focus from one item to another, such as NEXT_ITEM , and those that involve database transactions, such as CREATE_RECORD. Restricted built-ins are illegal in triggers that fire in response to navigation, such as Pre- and Post- navigational triggers. Each built-in description includes a "Built-in Type" section that indicates whether the built-in is restricted or unrestricted. For more information, refer to the Oracle Forms Developers Guide, Chapter 7, "Writing Event Triggers."

While you can write a trigger that uses any DML statement, Oracle Corporation advises that you follow the recommendations that are stated for each trigger type. Using DML statements in certain triggers can desynchronize the state of records in Oracle Forms and rows in the database, and can cause unexpected results.

1.3 Trigger Tables

The following cross references are included to help you locate the triggers you need in each category.

1.3.1 Block Processing Triggers

1.3.1.1 When-Clear-Block

Fires just before Oracle Forms clears the data from the current block.
Note that the When-Clear-Block trigger does not fire when Oracle Forms clears the current block during the CLEAR_FORM event.

1.3.1.2 When-Create-Record

Fires whenever Oracle Forms creates a new record. For example, when the operator presses the [Insert] key, or navigates to the last record in a set while scrolling down, Oracle Forms fires this trigger.

1.3.1.3 When-Database-Record

Fires when Oracle Forms first marks a record as an insert or an update. That is, the trigger fires as soon as Oracle Forms determines through validation that the record should be processed by the next post or commit as an insert or update. This generally occurs only when the operator modifies the first item in a record, and after the operator attempts to navigate out of the item.

1.3.1.4 When-Remove-Record

Fires whenever the operator or the application clears or deletes a record.

1.3.2 Interface Event Triggers

1.3.2.1 When-Button-Pressed

Fires when an operator selects a button, either by way of a key, or by clicking with a mouse.

1.3.2.2 When-Checkbox-Changed

Fires whenever an operator changes the state of a check box, either by clicking with the mouse, or through keyboard interaction.

1.3.2.3 When-Custom-Item-Event

Fires whenever a VBX control sends an event to Oracle Forms.

1.3.2.4 When-Image-Activated

Fires when an operator double-clicks on an image item with the mouse.
Note that When-Image-Pressed also fires on a double-click.

1.3.2.5 When-Image-Pressed

Fires when an operator uses the mouse to:

- single-click on an image item
- double-click on an image item (note that When-Image-Activated also fires on a double-click)

1.3.2.6 When-List-Changed

Fires when an operator selects a different element in a list item or de-selects the currently selected element. In addition, if a When-List-Changed trigger is attached to a combo box style list item, it fires each time the operator enters or modifies entered text.

1.3.2.7 When-Mouse-Click

Fires after the operator clicks the mouse if one of the following events occurs:

- if attached to the form, when the mouse is clicked within any canvas-view or item in the form
- if attached to a block, when the mouse is clicked within any item in the block
- if attached to an item, when the mouse is clicked within the item

Three events must occur before a When-Mouse-Click trigger will fire:

- Mouse down
- Mouse up
- Mouse click

Any trigger that is associated with these events will fire before the When-Mouse-Click trigger fires.

1.3.2.8 When-Mouse-DoubleClick

Fires after the operator double-clicks the mouse if one of the following events occurs:

- if attached to the form, when the mouse is double-clicked within any canvas-view or item in the form
- if attached to a block, when the mouse is double-clicked within any item in the block
- if attached to an item, when the mouse is double-clicked within the item

Six events must occur before a When-Mouse-DoubleClick trigger will fire:

- Mouse down
- Mouse up
- Mouse click
- Mouse down
- Mouse up
- Mouse double-click

Any trigger that is associated with these events will fire before the When-Mouse-DoubleClick trigger fires.

1.3.2.9 When-Mouse-Down

Fires after the operator presses down the mouse button if one of the following events occurs:

- if attached to the form, when the mouse is pressed down within any canvas-view or item in the form
- if attached to a block, when the mouse is pressed down within any item in the block

- if attached to an item, when the mouse is pressed within the item

Note: The mouse down event is always followed by a mouse up event.

1.3.2.10 When-Mouse-Enter

Fires when the mouse enters an item or canvas-view if one of the following events occurs:
- if attached to the form, when the mouse enters any canvas-view or item in the form
- if attached to a block, when the mouse enters any item in the block
- if attached to an item, when the mouse enters the item

1.3.2.11 When-Mouse-Leave

Fires after the mouse leaves an item or canvas-view if one of the following events occurs:
- if attached to the form, when the mouse leaves any canvas-view or item in the form
- if attached to a block, when the mouse leaves any item in the block
- if attached to an item, when the mouse leaves the item

1.3.2.12 When-Mouse-Move

Fires each time the mouse moves if one of the following events occurs:
- if attached to the form, when the mouse moves within any canvas-view or item in the form
- if attached to a block, when the mouse moves within any item in the block
- if attached to an item, when the mouse moves within the item*When-Mouse-Up*

1.3.2.13 When-Mouse-Up

Fires each time the operator presses down and releases the mouse button if one of the following events occurs:
- if attached to the form, when the mouse up event is received within any canvas-view or item in a form
- if attached to a block, when the mouse up event is received within any item in a block
- if attached to an item, when the mouse up event is received within an item

Two events must occur before a When-Mouse-Up trigger will fire:
- Mouse down
- Mouse up

1.3.2.14 When-Radio-Changed

Fires when an operator selects a different radio button in a radio group, or de-selects the currently selected radio button, either by clicking with the mouse, or by way of keyboard selection commands.

1.3.2.15 When-Timer-Expired

Fires when a timer expires.

Note: Timers are created programmatically by calling the CREATE_TIMER built-in procedure.

1.3.2.16 When-Window-Activated

Fires when a window is made the active window. This occurs at form startup and whenever a different window is given focus. Note that on some window managers, a window can be activated by, say, clicking on its title bar. This operation is independent of navigation to an item in the window. Thus, navigating to an item in a different window always activates that window, but window activation can also occur independently of navigation.

1.3.2.17 When-Window-Closed

Fires when an operator closes a window using a window-manager specific Close command.

1.3.2.18 When-Window-Deactivated

Fires when an operator deactivates a window by setting the input focus to another window.

1.3.2.19 When-Window-Resized

Fires when a window is resized, either by the operator or programmatically through a call to RESIZE_WINDOW or SET_WINDOW_PROPERTY. (Even if the window is not currently displayed, resizing the window programmatically fires the When-Window-Resized trigger.) This trigger also fires at form startup, when the root window is first drawn. It does not fire when a window is iconified.

1.3.3 Key Triggers

1.3.3.1 Function Key

Function key triggers are associated with individual Runform function keys. A function key trigger fires only when an operator presses the associated function key. The actions you define in a function key trigger replace the default action that the function key would normally perform.
The following paragraph shows all function key triggers and the corresponding Runform function keys
Key-CLRBLK [Clear Block], Key-CLRFRM [Clear Form], Key-CLRREC [Clear Record], Key-COMMIT [Accept], Key-CQUERY [Count Query Hits], Key-CREREC [Insert Record], Key-DELREC [Delete Record], Key-DOWN [Down], Key-DUP-ITEM [Duplicate Item], Key-DUPREC [Duplicate Record], Key-EDIT [Edit], Key-ENTQRY [Enter Query], Key-EXEQRY [Execute Query], Key-EXIT [Exit], Key-HELP [Help], Key-LISTVAL [List of Values], Key-MENU [Block Menu], Key-NXTBLK [Next Block], Key-NXT-ITEM [Next Item], Key-NXTKEY [Next Primary Key], Key-NXTREC [Next Record], Key-NXTSET [Next Set of Records], Key-PRINT [Print], Key-PRVBLK [Previous Block], Key-PRV-ITEM [Previous Item], Key-PRVREC [Previous Record], Key-SCRDOWN [Scroll Down], Key-SCRUP [Scroll Up], Key-UP [Up], Key-UPDREC (Equivalent to Record, Lock command on the default menu).
Note that you cannot redefine all Runform function keys with function key triggers. Specifically, you cannot ever redefine the following static function keys because they are often performed by the terminal or user interface management system and not by Oracle Forms.
[Clear Item], [Copy], [Cut], [Delete Character], [Delete Line], [Display Error], [End of Line], [First Line], [Insert Line], [Last Line], [Left], [Paste], [Refresh], [Right], [Scroll Left], [Scroll Right], [Search], [Select], [Show Keys], [Toggle Insert/Replace], [Transmit].
The default functionality performed by the following keys is not allowed in Enter Query mode:
[Clear Block], [Clear Form], [Clear Record], [Accept], [Insert Record], [Delete Record], [Down], [Duplicate Item], [Duplicate Record], [Block Menu], [Next Block], [Next Primary Key], [Next Record], [Next Set of Records], [Previous Block], [Previous Record], [Up], [Lock Record].

1.3.3.2 Key-Fn

A Key-Fn trigger fires when an operator presses the associated key.
You can attach Key-Fn triggers to 10 keys or key sequences that normally do not perform any Oracle Forms operations. These keys are referred to as Key-F0 through Key-F9. Before you can attach key triggers to these keys, you or the DBA must use Oracle Terminal to map the keys to the appropriate functions.

1.3.3.3 Key-Others

A Key-Others trigger fires when an operator presses the associated key.
A Key-Others trigger is associated with all keys that can have key triggers associated with them but are not currently defined by function key triggers (at any level).
A Key-Others trigger overrides the default behavior of a Runform function key (unless one of the restrictions apply). When this occurs, however, Oracle Forms still displays the function key's default entry in the Show Keys screen.

1.3.4 Master-Detail Triggers

1.3.4.1 On-Check-Delete-Master

Oracle Forms creates this trigger automatically when you define a master-detail relation and set the Master Deletes property to Non-Isolated. It fires when there is an attempt to delete a record in the master block of a master-detail relation.

1.3.4.2 On-Clear-Details

Fires when a coordination-causing event occurs in a block that is a master block in a master-detail relation. A coordination-causing event is any event that makes a different record the current record in the master block.

1.3.4.3 On-Populate-Details

Oracle Forms creates this trigger automatically when you define a master-detail relation. It fires when Oracle Forms would normally need to populate the detail block in a master-detail relation.

1.3.5 Message-Handling Triggers

1.3.5.1 On-Error

An On-Error trigger fires whenever Oracle Forms would normally cause an error message to display.

1.3.5.2 On-Message

Fires whenever Oracle Forms would normally cause a message to display.

1.3.6 Navigational Triggers

1.3.6.1 Post-Block

Fires during the Leave the Block process.

1.3.6.2 Post-Form

Fires during the Leave the Form process, when a form is exited.

1.3.6.3 Post-Record

Fires during the Leave the Record process. Specifically, the Post-Record trigger fires whenever the operator or the application moves the input focus from one record to another. The Leave the Record process can occur as a result of numerous operations, including INSERT_RECORD, DELETE_RECORD, NEXT_RECORD, NEXT_BLOCK, CREATE_RECORD, PREVIOUS_BLOCK, etc.

1.3.6.4 Post-Text-Item

Fires during the Leave the Item process for a text item. Specifically, this trigger fires when the input focus moves from a text item to any other item.

1.3.6.5 Pre-Block

Fires during the Enter the Block process, during navigation from one block to another.

1.3.6.6 Pre-Form

Fires during the Enter the Form event, at form startup.

1.3.6.7 Pre-Record

Fires during the Enter the Record process, during navigation to a different record.

1.3.6.8 Pre-Text-Item

Fires during the Enter the Item process, during navigation from an item to a text item.

1.3.6.9 When-Form-Navigate

Fires whenever any peer form navigation takes place.

1.3.6.10 When-New-Block-Instance

Fires when the input focus moves to an item in a block that is different than the block that previously had input focus. Specifically, it fires after navigation to an item, when Oracle Forms is ready to accept input in a block that is different than the block that previously had input focus.

1.3.6.11 When-New-Form-Instance

At form start-up, Oracle Forms navigates to the first navigable item in the first navigable block. A When-New-Form-Instance trigger fires after the successful completion of any navigational triggers that fire during the initial navigation sequence.
This trigger does not fire when control returns to a calling form from a called form.
In a multiple-form application, this trigger does not fire when focus changes from one form to another.

1.3.6.12 When-New-Item-Instance

Fires when the input focus moves to an item. Specifically, it fires after navigation to an item, when Oracle Forms is ready to accept input in an item that is different than the item that previously had input focus.

1.3.6.13 When-New-Record-Instance

Fires when the input focus moves to an item in a record that is different than the record that previously had input focus. Specifically, it fires after navigation to an item in a record, when Oracle Forms is ready to accept input in a record that is different than the record that previously had input focus.
Fires whenever Oracle Forms instantiates a new record.

1.3.7 Query-Time Triggers

1.3.7.1 Post-Query

When a query is open in the block, the Post-Query trigger fires each time Oracle Forms fetches a record into a block. The trigger fires once for each record placed on the block's list of records.

1.3.7.2 Pre-Query

Fires during Execute Query or Count Query processing, just before Oracle Forms constructs and issues the SELECT statement to identify rows that match the query criteria.

1.3.8 Transactional Triggers

1.3.8.1 On-Check-Unique

During a commit operation, the On-Check-Unique trigger fires when Oracle Forms normally checks that primary key values are unique before inserting or updating a record in a base table. It fires once for each record that has been inserted or updated.

1.3.8.2 On-Close

Fires when an operator or the application causes a query to close. By default, Oracle Forms closes a query when all of the records identified by the query criteria have been fetched, or when the operator or the application aborts the query.
The On-Close trigger augments the normal Oracle Forms "close cursor" phase of

1.3.8.3 On-Column-Security

Fires when Oracle Forms would normally enforce column-level security for each block that has the Column Security block property set On.

1.3.8.4 On-Commit

Fires whenever Oracle Forms would normally issue a database commit statement to finalize a transaction. By default, this operation occurs after all records that have been marked as updates, inserts, and deletes have been posted to the database.

1.3.8.5 On-Count

Fires when Oracle Forms would normally perform default Count Query processing to determine the number of rows in the database that match the current query criteria. When the On-Count trigger completes execution, Oracle Forms issues the standard query hits message: FRM-40355: Query will retrieve <n> records.

1.3.8.6 On-Delete

Fires during the Post and Commit Transactions process. Specifically, it fires after the Pre-Delete trigger fires and before the Post-Delete trigger fires, replacing the actual database delete of a given row. The trigger fires once for each row that is marked for deletion from the database.

1.3.8.7 On-Fetch

When a query is first opened, fires immediately after the On-Select trigger fires, when the first records are fetched into the block. While the query remains open, fires again each time a set of rows must be fetched into the block.

1.3.8.8 On-Insert

Fires during the Post and Commit Transactions process. Specifically, it fires after the Pre-Insert trigger fires and before the Post-Insert trigger fires, when Oracle Forms would normally insert a record in the database. It fires once for each row that is marked for insertion into the database.

1.3.8.9 On-Lock

Fires whenever Oracle Forms would normally attempt to lock a row, such as when an operator presses a key to modify data in an item. The trigger fires between the keypress and the display of the modified data.

1.3.8.10 On-Logon

Fires once per logon when Oracle Forms normally initiates the logon sequence.

1.3.8.11 On-Logout

Fires when Oracle Forms normally initiates a logout procedure.

1.3.8.12 On-Rollback

Fires when Oracle Forms would normally issue a ROLLBACK statement, to roll back a transaction to the last savepoint that was issued.

1.3.8.13 On-Savepoint

Fires when Oracle Forms would normally issue a Savepoint statement. By default, Oracle Forms issues savepoints at form startup, and at the start of each Post and Commit Transaction process.

1.3.8.14 On-Select

Fires when Oracle Forms would normally execute the selection phase of a query, to identify the records in the database that match the current query criteria.

1.3.8.15 On-Sequence-Number

Fires when Oracle Forms would normally perform the default processing for generating sequence numbers for default item values.

1.3.8.16 On-Update

Fires during the Post and Commit Transactions process. Specifically, it fires after the Pre-Update trigger fires and before the Post-Update trigger fires, when Oracle Forms would normally update a record in the database. It fires once for each row that is marked for update in the form.

1.3.8.17 Post-Change

Fires when any of the following conditions exist:

- The Validate the Item process determines that an item is marked as Changed and is not NULL.
- An operator returns a value into an item by making a selection from a list of values, and the item is not NULL.
- Oracle Forms fetches a non-NULL value into an item. In this case, the When-Validate-Item trigger does not fire. If you want to circumvent this situation and effectively get rid of the Post-Change trigger, you must include a Post-Query trigger in addition to your When-Validate-Item trigger. See "Usage Notes" below.

1.3.8.18 Post-Database-Commit

Fires once during the Post and Commit Transactions process, after the database commit occurs. Note that the Post-Forms-Commit trigger fires after inserts, updates, and deletes have been posted to the database, but before the transaction has been finalized by issuing the Commit. The Post-Database-Commit Trigger fires after Oracle Forms issues the Commit to finalize the transaction.

1.3.8.19 Post-Delete

Fires during the Post and Commit Transactions process, after a row is deleted. It fires once for each row that is deleted from the database during the commit process.

1.3.8.20 Post-Forms-Commit

Fires once during the Post and Commit Transactions process. If there are records in the form that have been marked as inserts, updates, or deletes, the Post-Forms-Commit trigger fires after these changes have been written to the database but before Oracle Forms issues the database Commit to finalize the transaction.
If the operator or the application initiates a Commit when there are no records in the form have been marked as inserts, updates, or deletes, Oracle Forms fires the Post-Forms-Commit trigger immediately, without posting changes to the database.

1.3.8.21 Post-Insert

Fires during the Post and Commit Transactions process, just after a record is inserted. It fires once for each record that is inserted into the database during the commit process.

1.3.8.22 Post-Logon

Fires after either of the following events:

- The successful completion of Oracle Forms default logon processing.
- The successful execution of the On-Logon trigger.

1.3.8.23 Post-Logout

Fires after either of the following events:

- Oracle Forms successfully logs out of ORACLE.
- The successful execution of the On-Logout trigger.

1.3.8.24 Post-Select

The Post-Select trigger fires after the default selection phase of query processing, or after the successful execution of the On-Select trigger. It fires before any records are actually retrieved through fetch processing.

1.3.8.25 Post-Update

Fires during the Post and Commit Transactions process, after a row is updated. It fires once for each row that is updated in the database during the commit process.

1.3.8.26 Pre-Commit

Fires once during the Post and Commit Transactions process, before Oracle Forms processes any records to change. Specifically, it fires after Oracle Forms determines that there are inserts, updates, or deletes in the form to post or commit. The trigger does not fire when there is an attempt to commit, but validation determines that there are no changed records in the form.

1.3.8.27 Pre-Delete

Fires during the Post and Commit Transactions process, before a row is deleted. It fires once for each record that is marked for delete.
Note: Oracle Forms creates a Pre-Delete trigger automatically for any master-detail relation that has the Master Deletes property set to Cascading.
Pre-Insert
Fires during the Post and Commit Transactions process, before a row is inserted. It fires once for each record that is marked for insert.

1.3.8.28 Pre-Logon

Fires just before Oracle Forms initiates a logon procedure to the data source.

1.3.8.29 Pre-Logout

Fires once before Oracle Forms initiates a logout procedure.

1.3.8.30 Pre-Select

Fires during Execute Query and Count Query processing, after Oracle Forms constructs the SELECT statement to be issued, but before the statement is actually issued. Note that the SELECT statement can be examined in a Pre-Select trigger by reading the value of the system variable SYSTEM.LAST_QUERY.

1.3.8.31 Pre-Update

Fires during the Post and Commit Transactions process, before a row is updated. It fires once for each record that is marked for update.

1.3.9 Validation Triggers

1.3.9.1 When-Validate-Item

Fires during the Validate the Item process. Specifically, it fires as the last part of item validation for items with the New or Changed validation status.

1.3.9.2 When-Validate-Record

Fires during the Validate the Record process. Specifically, it fires as the last part of record validation for records with the New or Changed validation status.

1.3.9.3 User-Named Trigger

A user-named trigger is a trigger that you define yourself in a form, and then call explicitly from other triggers or user-named subprograms. Each user-named trigger defined at the same definition level must have a unique name.
To execute a user-named trigger, you must call the EXECUTE_TRIGGER built-in procedure, as shown here:
Execute_Trigger('my_user_named_trigger');
Note: You can write user-named PL/SQL subprograms to perform almost any task for which you might use a user-named trigger.

2. System Variables (CHAPTER 4.)

2.1 About System Variables

A system variable is an Oracle Forms variable that keeps track of an internal Oracle Forms state. You can reference the value of a system variable to control the way an application behaves.
Oracle Forms maintains the values of system variables on a per form basis. That is, the values of all system variables correspond only to the current form. The following list presents the names of the available system variables

2.1.1 List of system variables

2.1.1.1 SYSTEM.BLOCK_STATUS

Represents the status of the block where the cursor is located, or the current block during trigger processing. The value can be one of three character strings:

- CHANGED -Indicates that the block contains at least one Changed record.
- NEW - Indicates that the block contains only New records.
- QUERY - Indicates that the block contains only Valid records that have been retrieved from the database.

2.1.1.2 SYSTEM.COORDINATION_OPERATION

This system variable works with its companion SYSTEM.MASTER_BLOCK to help an On-Clear-Details trigger determine what type of coordination-causing operation fired the trigger, and on which master block of a master-detail relation.

2.1.1.3 SYSTEM.CURRENT_BLOCK

The value that the SYSTEM.CURRENT_BLOCK system variable represents depends on the current navigation unit:

- If the current navigation unit is the block, record, or item (as in the Pre- and Post- Item, Record, and Block triggers), the value of SYSTEM.CURRENT_BLOCK is the name of the block that Oracle Forms is processing or that the cursor is in.
- If the current navigation unit is the form (as in the Pre- and Post-Form triggers), the value of SYSTEM.CURRENT_BLOCK is NULL.

The value is always a character string.
Note: SYSTEM.CURRENT_BLOCK is included for compatibility with previous versions. Oracle Corporation recommends that you use SYSTEM.CURSOR_BLOCK and SYSTEM.TRIGGER_BLOCK instead.

2.1.1.4 SYSTEM.CURRENT_DATETIME

Is a variable representing the operating system date. The value is a CHAR string in the following format: DD-MON-YYYY HH24:MI:SS

2.1.1.5 SYSTEM.CURRENT_FORM

Represents the name of the form that Oracle Forms is executing. The value is always a character string.

2.1.1.6 SYSTEM.CURRENT_ITEM

The value that the SYSTEM.CURRENT_ITEM system variable represents depends on the current navigation unit:

- If the current navigation unit is the item (as in the Pre- and Post-Item triggers), the value of SYSTEM.CURRENT_ITEM is the name of the item that Oracle Forms is processing or that the cursor is in. The returned item name does not include a block name prefix.
- If the current navigation unit is the record, block, or form (as in the Pre- and Post- Record, Block, and Form triggers), the value of SYSTEM.CURRENT_ITEM is NULL.

The value is always a character string.
Note: SYSTEM.CURRENT_ITEM is included for compatibility with previous versions. Oracle Corporation recommends that you use SYSTEM.CURSOR_ITEM or SYSTEM.TRIGGER_ITEM instead.

2.1.1.7 SYSTEM.CURRENT_VALUE

Represents the value of the item that is registered in SYSTEM.CURRENT_ITEM.
The value is always a character string.
Note: SYSTEM.CURRENT_VALUE is included for compatibility with previous versions. Oracle Corporation recommends that you use SYSTEM.CURSOR_ITEM and SYSTEM.CURSOR_VALUE instead.

2.1.1.8 SYSTEM.CURSOR_BLOCK

The value that the SYSTEM.CURSOR_BLOCK system variable represents depends on the current navigation unit:

- If the current navigation unit is the block, record, or item (as in the Pre- and Post- Item, Record, and Block triggers), the value of SYSTEM.CURSOR_BLOCK is the name of the block where the cursor is located. The value is always a character string.
- If the current navigation unit is the form (as in the Pre- and Post-Form triggers), the value of SYSTEM.CURSOR_BLOCK is NULL.

2.1.1.9 SYSTEM.CURSOR_ITEM

Represents the name of the block and item, block.item, where the input focus (cursor) is located.
The value is always a character string.

2.1.1.10 SYSTEM.CURSOR_RECORD

Represents the number of the record where the cursor is located. This number represents the record's current physical order in the block's list of records. The value is always a character string.

2.1.1.11 SYSTEM.CURSOR_VALUE

Represents the value of the item where the cursor is located. The value is always a character string.

2.1.1.12

SYSTEM.CUSTOM_ITEM_EVENT
Stores the name of the event fired by a VBX control.

2.1.1.13 SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS

Stores the supplementary arguments for an event fired by a VBX control.

2.1.1.14 SYSTEM.DATE_THRESHOLD*

Represents the database date requery threshold. This variable works in conjunction with the three system variables $$DBDATE$$, $$DBDATETIME$$, and $$DBTIME$$, and controls how often Oracle Forms synchronizes the database date with the RDBMS. The value of this variable must be specified in the following format:
MI:SS
Because frequent RDBMS queries can degrade performance, it is best to keep this value reasonably high. However, keep in mind that if the value is not synchronized often enough, some time discrepancy can occur. In addition, if you are building a client-server application, the performance implications of SYSTEM.DATE_THRESHOLD could vary depending on the complexity of your network configuration.

2.1.1.15 SYSTEM.EFFECTIVE_DATE*

Represents the effective database date. The variable value must always be in the following format:
DD-MON-YYYY HH24:MI:SS

2.1.1.16 SYSTEM.EVENT_WINDOW

The SYSTEM.EVENT_WINDOW system variable represents the name of the last window that was affected by an action that caused one of the window event triggers to fire. The following triggers cause this variable to be updated:

- WHEN-WINDOW-ACTIVATED
- WHEN-WINDOW-CLOSED
- WHEN-WINDOW-DEACTIVATED
- WHEN-WINDOW-RESIZED

From within these triggers, you can assign the value of the variable to any of the following:

- global variable
- parameter
- variable

- item, including a null canvas item

2.1.1.17 SYSTEM.FORM_STATUS

represents the status of the current form. The value can be one of three character strings:

- CHANGED - Indicates that the form contains at least one block with a Changed record. The value of SYSTEM.FORM_STATUS becomes CHANGED only after at least one record in the form has been changed and the associated navigation unit has also changed.
- NEW - Indicates that the form contains only New records.
- QUERY - Indicates that a query is open. The form contains at least one block with QUERY records and no blocks with CHANGED records.

2.1.1.18 SYSTEM.LAST_FORM

Represents the form module ID of the previous form in a multi-form application, where multiple forms have been invoked using OPEN_FORM. The value can be one of two character strings: either the form module ID or NULL.

2.1.1.19 SYSTEM.LAST_QUERY

Represents the query SELECT statement that Oracle Forms most recently used to populate a block during the current Runform session. The value is always a character string.

2.1.1.20 SYSTEM.LAST_RECORD

Indicates whether the current record is the last record in a block's list of records. The value is one of the following two CHAR values:

- TRUE - Indicates that the current record is the last record in the current block's list of records.
- FALSE - Indicates that the current record is not the last record in the current block's list of records.

2.1.1.21 SYSTEM.MASTER_BLOCK

This system variable works with its companion SYSTEM.COORDINATION_OPERATION to help an On-Clear-Details trigger determine what type of coordination-causing operation fired the trigger, and on which master block of a master-detail relation.
The values of the two system variables remain constant throughout the clearing phase of any block synchronization. SYSTEM.MASTER_BLOCK represents the name of the driving master block, and SYSTEM_COORDINATION_OPERATION represents the coordination-causing event that occurred on the master block.
More details are in the description for SYSTEM.COORDINATION_OPERATION.

2.1.1.22 SYSTEM.MESSAGE_LEVEL*

Represents one of the following message severity levels: 0, 5, 10, 15, 20, or 25. The value is always a character string.
During a Runform session, Oracle Forms suppresses all messages with a severity level that is the same or lower (less severe) than the indicated severity level.
Assign a value to the SYSTEM.MESSAGE_LEVEL system variable with standard PL/SQL syntax:

:System.Message_Level := value;

The legal values for SYSTEM.MESSAGE_LEVEL are 0, 5, 10, 15, 20, and 25. Oracle Forms does not suppress prompts or vital error messages, no matter what severity level you select.

2.1.1.23 SYSTEM.MODE

SYSTEM.MODE indicates whether the form is in Normal, Enter Query, or Fetch Processing mode. The value is always a character string.

- NORMAL - Indicates that the form is currently in normal processing mode.
- ENTER-QUERY - Indicates that the form is currently in Enter Query mode.
- QUERY - Indicates that the form is currently in fetch processing mode, meaning that a query is currently being processed.

2.1.1.24 SYSTEM.MOUSE_BUTTON_PRESSED

Indicates the number of the button that was clicked. Mouse button support is limited to buttons 1 and 2 (left or middle) on a three button mouse. The value is always a character string.

2.1.1.25 SYSTEM.MOUSE_BUTTON_SHIFT_STATE

Indicates the key that was pressed during the click, such as SHIFT, ALT, or CONTROL. The value is always a character string.

2.1.1.26 SYSTEM.MOUSE_CANVAS

If the mouse is in a canvas, SYSTEM.MOUSE_CANVAS represents the name of that canvas as a CHAR value. If the mouse is in an item, this variable represents the name of the canvas containing the item.
SYSTEM.MOUSE_CANVAS is NULL if:

- the mouse is not in a canvas
- the platform is non-GUI

SYSTEM.MOUSE_FORM
If the mouse is in a form module, SYSTEM.MOUSE_FORM represents the name of that form module as a CHAR value. For example, if the mouse is in Form_Module1, the value for SYSTEM.MOUSE_ITEM is FORM_MODULE1.
Note: SYSTEM.MOUSE_FORM is NULL if the platform is not a GUI platform.

SYSTEM.MOUSE_ITEM
If the mouse is in an item, SYSTEM.MOUSE_ITEM represents the name of that item as a CHAR value. For example, if the mouse is in Item1 in Block2, the value for SYSTEM.MOUSE_ITEM is :BLOCK2.ITEM1.
SYSTEM.MOUSE_ITEM is NULL if:

- the mouse is not in an item
- the platform is not a GUI platform

SYSTEM.MOUSE_RECORD
If the mouse is in a record, SYSTEM.MOUSE_RECORD represents that record's record number as a CHAR value.
Note: SYSTEM.MOUSE_RECORD is 0 if the mouse is not in an item (and thus, not in a record).

2.1.1.27 SYSTEM.MOUSE_RECORD_OFFSET

If the mouse is in a record, SYSTEM.MOUSE_RECORD_OFFSET represents the offset from the first visible record as a CHAR value.
For example, if the mouse is in the second of five visible records in a multi-record block, SYSTEM.MOUSE_RECORD_OFFSET is 2. (SYSTEM.MOUSE_RECORD_OFFSET uses a 1-based index).
Note: SYSTEM.MOUSE_RECORD_OFFSET is 0 if the mouse is not in an item (and thus, not in a record).

2.1.1.28 SYSTEM.MOUSE_X_POS

represents (as a CHAR value) the x coordinate of the mouse in the units of the current form coordinate system. If the mouse is in an item, the value is relative to the upper left corner of the item's bounding box. If the mouse is on a canvas, the value is relative to the upper left corner of the canvas.
Note: SYSTEM.MOUSE_X_POS is always NULL on character mode platforms.

2.1.1.29 SYSTEM.MOUSE_Y_POS

represents (as a CHAR value) the y coordinate of the mouse, using units of the current coordinate system. If the mouse is in an item, the value is relative to the upper left corner of the item's bounding box. If the mouse is on a canvas, the value is relative to the upper left corner of the canvas.
Note: SYSTEM.MOUSE_Y_POS is always NULL on character mode platforms.

2.1.1.30 SYSTEM.RECORD_STATUS

Represents the status of the record where the cursor is located. The value can be one of four character strings:

- CHANGED - Indicates that a queried record's validation status is Changed.
- INSERT - Indicates that the record's validation status is Changed and that the record does not exist in the database.
- NEW - Indicates that the record's validation status is New.
- QUERY - Indicates that the record's validation status is Valid and that it was retrieved from the database.

2.1.1.31 SYSTEM.SUPPRESS_WORKING*

suppresses the "Working..." message in Runform, in order to prevent the screen update usually caused by the display of the "Working..." message. The value of the variable is one of the following two CHAR values:

- TRUE - Prevents Oracle Forms from issuing the "Working..." message.
- FALSE - Allows Oracle Forms to continue to issue the "Working..." message.

2.1.1.32 SYSTEM.TRIGGER_BLOCK

Represents the name of the block where the cursor was located when the current trigger initially fired. The value is NULL if the current trigger is a Pre- or Post-Form trigger. The value is always a character string.

2.1.1.33 SYSTEM.TRIGGER_ITEM

Represents the item (BLOCK.ITEM) in the scope for which the trigger is currently firing. When referenced in a key trigger, it represents the item where the cursor was located when the trigger began. The value is always a character string.

2.1.1.34 SYSTEM.TRIGGER_RECORD

represents the number of the record that Oracle Forms is processing. This number represents the record's current physical order in the block's list of records. The value is always a character string.

All system variables, except the four indicated with an asterisk (*), are read-only variables. These four variables are the only system variables to which you can explicitly assign values.

2.1.2 Date and Time System Default Values

Oracle Forms also supplies six special default values--$$DATE$$, $$DATETIME$$, $$TIME$$, $$DBDATE$$, $$DBDATETIME$$, and $$DBTIME$$--that supply date and time information and have special restrictions on their use:

- If you're building client/server applications, consider the performance implications of going across the network to get date and time information.
- If you're accessing a non-ORACLE datasource, avoid using $$DBDATE$$ and $$DBDATETIME$$. Instead, use a When-Create-Record trigger to select the current date in a datasource-specific manner.
- Use $$DATE$$, $$DATETIME$$, and $$TIME$$ to obtain the local system date/time; use $$DBDATE$$, $$DBDATETIME$$, and $$DBTIME$$ to obtain the database date/time, which may differ from the local system date/time if, for example, you're connecting to a remote database in a different time zone.
- Use these variables only to set the value of the Default Value, Range Low Value or Range High Value property.

2.1.2.1 $$DATE$$

Retrieves the current operating system date. You can use $$DATE$$ to designate a default value or range for a text item using the Default or Range property. The text item must be of the CHAR, DATE, or DATETIME data type.
You also can use $$DATE$$ as a default value for form parameters. In this case, the parameter's value is computed once, at form startup.

2.1.2.2 $$DATETIME$$

Retrieves the current operating system date and time. You can use $$DATETIME$$ to designate a default value or range for a text item using the Default or Range property. The text item must be of the CHAR or DATETIME data type.
You also can use $$DATETIME$$ as a default value for form parameters. In this case, the parameter's value is computed once, at form startup.
The difference between $$DATE$$ and $$DATETIME$$ is that the time component for $$DATE$$ is always fixed to 00:00:00, compared to $$DATETIME$$, which includes a meaningful time component, such as 09:17:59.
Note: Do not use $$DATETIME$$ instead of $$DATE$$ unless you plan to specify the time component. If, for example, you use $$DATETIME$$ with the default DATE format mask of DD-MON-YY, you would be committing values to the database that the user would not see, because the format mask does not include a time component. Then, because you had committed specific time information, when you later queried on date, the values would not match and you would not return any rows.

2.1.2.3 $$DBDATE$$

Retrieves the current database date. You can use $$DBDATE$$ to designate a default value or range for a text item using the Default or Range property. The text item must be of the CHAR, DATE, or DATETIME data type.

2.1.2.4 $$DBDATETIME$$

Retrieves the current date and time from the local database. You can use $$DBDATETIME$$ to designate a default value or range for a text item using the Default or Range property. The text item must be of the CHAR or DATETIME data type.

2.1.2.5 $$DBTIME$$

Retrieves the current time from the local database. You can use $$DBTIME$$ to designate a default value or range for a text item using the Default or Range property. The text item must be of the CHAR or TIME data type.

2.1.2.6 $$TIME$$

Retrieves the current operating system time. You can use $$TIME$$ to designate a default value or range for a text item using the Default or Range property. The text item must be of the CHAR or TIME data type.
You also can use $$TIME$$ as a default value for form parameters. In this case, the parameter's value is computed once, at form startup.

2.1.3 Local Variables

Because system variables are derived, if the value is not expected to change over the life of the trigger, you can save the system value in a local variable and use the local variable multiple times.

3. Built-in Subprograms (CHAPTER 3.)

This chapter includes information about Oracle Forms built-in subprograms included in the STANDARD Extensions, FORMS_OLE, and VBX packages.
For information on the following packages, refer to the Online Help system or the Oracle Procedure Builder User's Guide and Reference: OLE2, TOOL_ENV, ORA_NLS, TOOL_RES, ORA_FFI, ORA_DE, STPROC, TEXT_IO.

3.1 Overview

Oracle Forms provides built-in subprograms that you can call from triggers and user-named subprograms that you write yourself. Built-ins provide programmatic control over standard application functions, including navigation, interface control, and transaction processing.

3.1.1 Syntax

Refer to the Preface in this book for typographic conventions. Named parameters are shown in an italic monospaced font. You can replace any named parameter with the actual parameter, which can be a constant, a literal, a bind variable, or a number.

```
SET_TIMER(timer_name, milliseconds, iterate);
```

3.1.2 Named Parameters

You can directly use any of the named parameters shown in the syntax diagrams in this chapter. The named parameter should be followed with the equal/greater than signs (=>), which point to the actual parameter that follows the named parameter. For example, if you intend to change the milliseconds in the SET_TIMER Built-in you can directly use that parameter with the following syntax:

```
SET_TIMER(timer_name => 'my_timer', milliseconds => 12000, iterate => NO_REPEAT);
```

Also, you can continue to call the built-in with the following syntax:

```
SET_TIMER('my_timer', 12000, NO_REPEAT);
```

3.1.3 Object IDs

Some built-in subprograms accept object IDs as actual parameters. An object ID is an internal, opaque handle that is assigned to each object when created in the Designer. Object IDs are internally managed and cannot be externally viewed by the user. The only method you can use to retrieve the ID is to define a local or global variable and assign the return value of the object to the variable.
You make the assignment by way of the FIND_ built-in functions. Once you have used FIND_ within a PL/SQL block, you can use the variable as an object ID while still in that block. The valid PL/SQL type for each object is included in the syntax descriptions for each parameter. The description for the FIND_BLOCK built-in provides an example of how to obtain an object ID.

3.1.4 Form Coordinate Units

Many built-in subprograms allow you to specify size and position coordinates, using properties such as:

HEIGHT
WIDTH
DISPLAY_POSITION
DISPLAY_X_POS
DISPLAY_Y_POS
VIEW_SIZE
X_POS_ON_CANVAS
Y_POS_ON_CANVAS

When you specify coordinates or width and height, you express these measurements in units of the current form coordinate system, set on the Form Module property sheet. The form coordinate system defines the units for specifying size and position coordinates of objects in the Designer. Use the form module property, Coordinate Information, to set the form's coordinate units:

character cells or

real units:
inches
centimeters
pixels
points

When you design in the character cell coordinate system, all object dimensions and position coordinates are expressed in character cells, so Oracle Forms accepts only whole numbers for size and position properties. When you design using real units (inches, centimeters, or points), all object dimensions and position coordinates are expressed in the units you specify, so Oracle Forms will accept decimals as well as whole numbers for size and position properties. The precision of real units is three digits, so you can specify coordinates to thousandths. If you use pixels or character cells, coordinates are truncated to whole numbers.

3.1.5 Uppercase Return Values

The GET_X_PROPERTY built-ins, such as GET_FORM_PROPERTY, return CHAR arguments as uppercase values. This will affect the way you compare results in IF statements.

3.1.6 Restricted Built-In Subprograms

Restricted built-ins affect navigation in your form, either external screen navigation, or internal navigation. You can call these built-ins only from triggers while no internal navigation is occurring.
Restricted built-ins cannot be called from the Pre and Post triggers, which fire when Oracle Forms is navigating from object to another.
Restricted built-ins can be called from the When triggers that are specific to interface items, such as When-Button-Pressed or When-Checkbox-Changed. Restricted built-ins can also be called from any of the When-New-"object"-Instance triggers and from key triggers.
Unrestricted built-ins do not affect logical or physical navigation and can be called from any trigger.
The descriptions of built-in subprograms in this chapter include a heading, Built-In Type, that indicates if the built-in is restricted or unrestricted.

3.1.7 Constants

Many of the built-in subprograms take numeric values as arguments. Often, constants have been defined for these numeric arguments. A constant is a named numeric value. When passing a constant to a built-in do not enclose the constant value in quotation marks.
Constants can only appear on the right side of an operator in an expression.
In some cases, a built-in can take a number of possible constants as arguments. Possible constants are listed in the descriptions for each parameter.
In the following example, BLOCK_SCOPE is a constant that can be supplied for the parameter constant VALIDATION_UNIT. Other constants listed in the description are FORM, RECORD, and ITEM.

```
SET_FORM_PROPERTY('my_form', VALIDATION_UNIT, BLOCK_SCOPE);
```

3.2 Built-in Subprograms Tables

The following cross reference list organizes the built-in subprograms by object type. In some cases, these objects are externalized interface objects, such as windows or canvases. In other cases, they are data structures such as records, record groups, or parameter lists.

3.2.1 Alert Built-ins

3.2.1.1 FIND_ALERT

Syntax: FIND_ALERT(alert_name);
Built-in Type: unrestricted function
Returns: Alert
Enter Query Mode: yes
Description:
Searches the list of valid alerts in Oracle Forms. When the given alert is located, the subprogram returns an alert ID. You must return the ID to an appropriately typed variable. Define the variable with a type of Alert.
Parameters:
alert_name Specifies the CHAR alert name.

3.2.1.2 ID_NULL

Syntax: ID_NULL(object_id);
Built-in Type: unrestricted function
Returns: BOOLEAN
Enter Query Mode: yes
Description: Returns a BOOLEAN value that indicates whether the object ID is available.
Parameters and usage notes: see the manual or online help

3.2.1.3 SET_ALERT_BUTTON_PROPERTY

Syntax: SET_ALERT_BUTTON_PROPERTY(alert_id, button, property, value);
SET_ALERT_BUTTON_PROPERTY(alert_name, button, property, value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Changes the label on one of the buttons in an alert.
Parameters and usage notes: see the manual or online help

3.2.1.4 SET_ALERT_PROPERTY

Syntax: SET_ALERT_PROPERTY(alert_id, property, message);
SET_ALERT_PROPERTY(alert_name, property, message);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Changes the message text for an existing alert.
Parameters and usage notes: see the manual or online help

3.2.1.5 SHOW_ALERT

Syntax: SHOW_ALERT(alert_id);
SHOW_ALERT(alert_name);
Built-in Type: unrestricted function
Returns: A numeric constant corresponding to the button the operator selected from the alert. Button mappings are specified in the alert design.
Enter Query Mode: yes
Description: Displays the given alert, and returns a numeric value when the operator selects one of three alert buttons.
Parameters and usage notes: see the manual or online help

3.2.2 Application Built-ins

3.2.2.1 DO_KEY

Syntax: DO_KEY(built-in_subprogram_name);
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Executes the key trigger that corresponds to the specified built-in subprogram. If no such key trigger exists, then the specified subprogram executes. This behavior is analogous to pressing the corresponding function key.
Parameters:
built-in_subprogram_name Specifies the name of a valid built-in subprogram.
Restrictions: DO_KEY accepts built-in names only, not key names: DO_KEY(ENTER_QUERY). To accept a specific key name, use the EXECUTE_TRIGGER built-in: EXECUTE_TRIGGER('KEY_F11').

3.2.2.2 GET_APPLICATION_PROPERTY

Syntax: GET_APPLICATION_PROPERTY(property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: The GET_APPLICATION_PROPERTY built-in returns information about the current Oracle Forms application. You must call this built-in once for each value you want to retrieve.
Parameters and usage notes: see the manual or online help

3.2.2.3 HOST

Syntax: HOST(system_command_string);
HOST(system_command_string, screen_action);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Executes an indicated operating system command.
Parameters and usage notes: see the manual or online help

3.2.2.4 PAUSE

Syntax: PAUSE;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Suspends processing until the operator presses a function key. PAUSE might display an alert.
Parameters: none

3.2.2.5 SET_APPLICATION_PROPERTY

Syntax: SET_APPLICATION_PROPERTY(property, value)
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the application property for the current application.
Parameters and usage notes: see the manual or online help

3.2.2.6 USER_EXIT

Syntax: USER_EXIT(user_exit_string);
USER_EXIT(user_exit_string, error_string);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Calls the user exit named in the user_exit_string.
For more information, refer to Oracle Forms Advanced Techniques, Ch. 3, "Writing User Exits."
Parameters:
user_exit_string Specifies the name of the user exit you want to call, including any parameters.
error_string Specifies a user-defined error message that Oracle Forms should display if the user exit fails.

3.2.3 Block Built-ins

3.2.3.1 BLOCK_MENU

Syntax: BLOCK_MENU;
Built-in Type: restricted procedure
Enter Query Mode: yes; however, it is illegal to navigate out of the current block in Enter Query mode
Description: Displays a list of values (LOV) containing the sequence number and names of valid blocks in your form. Oracle Forms sets the input focus to the first enterable item in the block you select from the LOV.
Parameters: none

3.2.3.2 CLEAR_BLOCK

Syntax: CLEAR_BLOCK;
CLEAR_BLOCK(commit_mode);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Causes Oracle Forms to remove all records from, or "flush," the current block.
Parameters and usage notes: see the manual or online help

3.2.3.3 FIND_BLOCK

Syntax: FIND_BLOCK(block_name);
Built-in Type: unrestricted function
Returns: Block
Enter Query Mode: yes
Description: Searches the list of valid blocks and returns a unique block ID. You must define an appropriately typed variable to accept the return value. Define the variable with a type of Block.
Parameters:

block_name Specifies the CHAR block name.

3.2.3.4 GET_BLOCK_PROPERTY

Syntax: GET_BLOCK_PROPERTY(block_id, property);
GET_BLOCK_PROPERTY(block_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns information about a specified block. You must issue a call to the built-in once for each property value you want to retrieve.
Parameters and usage notes: see the manual or online help

3.2.3.5 GO_BLOCK

Syntax: GO_BLOCK(block_name);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: GO_BLOCK navigates to an indicated block. If the target block is non-enterable, an error occurs.
Parameters:
block_name Specifies the name you gave the block when defining it. The data type of the name is CHAR.

3.2.3.6 ID_NULL

See Alert Built-ins

3.2.3.7 NEXT_BLOCK

Syntax: NEXT_BLOCK;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the first navigable item in the next enterable block in the navigation sequence. By default, the next block in the navigation sequence is the block with the next higher sequence number, as defined by the order of blocks in the Object Navigator. However, the Next Navigation Block block property can be set to specify a different block as the next block for navigation purposes.
If there is no enterable block with a higher sequence, NEXT_BLOCK navigates to the enterable block with the lowest sequence number.
Parameters: none

3.2.3.8 PREVIOUS_BLOCK

Syntax: PREVIOUS_BLOCK;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the first navigable item in the previous enterable block in the navigation sequence. By default, the previous block in the navigation sequence is the block with the next lower sequence number, as defined by the block order in the Object Navigator. However, the Previous Navigation Block block property can be set to specify a different block as the previous block for navigation purposes.
If there is no enterable block with a lower sequence, PREVIOUS_BLOCK navigates to the enterable block with the highest sequence number.
Parameters: none

3.2.3.9 SET_BLOCK_PROPERTY

Syntax: SET_BLOCK_PROPERTY(block_id, property, value);
SET_BLOCK_PROPERTY(block_name, property, value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the given block characteristic of the given block.
Parameters and usage notes: see the manual or online help

3.2.4 Canvas and View Built-ins

3.2.4.1 FIND_CANVAS

Syntax: FIND_CANVAS(canvas_name);
Built-in Type: unrestricted function

Returns: Canvas
Enter Query Mode: yes
Description: Searches the list of canvases and returns a canvas ID when it finds a valid canvas with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of Canvas.
Parameters:
canvas_name Specifies the CHAR canvas name you gave the canvas when defining it.

3.2.4.2 FIND_VIEW

Syntax: FIND_VIEW(viewcanvas_name);
Built-in Type: unrestricted function
Returns: ViewPort
Enter Query Mode: yes
Description: Searches the list of canvas-views and returns a view ID when it finds a valid canvas-view with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of ViewPort.
Parameters:
viewcanvas_name Specifies the CHAR name of the canvas-view.

3.2.4.3 GET_CANVAS_PROPERTY

Syntax: GET_CANVAS_PROPERTY(canvas_id, property);
GET_CANVAS_PROPERTY(canvas_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the given canvas property for the given canvas. .
Parameters and usage notes: see the manual or online help

3.2.4.4 GET_VIEW_PROPERTY

Syntax: GET_VIEW_PROPERTY(view_id, property);
GET_VIEW_PROPERTY(view_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the indicated property setting for the indicated canvas-view.
Parameters and usage notes: see the manual or online help

3.2.4.5 HIDE_VIEW

Syntax: HIDE_VIEW(view_id);
HIDE_VIEW(view_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Hides the indicated canvas-view.
Parameters and usage notes: see the manual or online help

3.2.4.6 ID_NULL

See Alert Built-ins

3.2.4.7 PRINT

Syntax: PRINT;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Prints the current window to a file or to the printer.
Parameters: none

3.2.4.8 SCROLL_VIEW

Syntax: SCROLL_VIEW(view_id, x, y);
SCROLL_VIEW(view_name, x, y);
Built-in Type: unrestricted procedure
Enter Query Mode: yes

Description: Moves the view to a different position on its canvas by changing the X Position on Canvas and Y Position on Canvas properties. Moving the view makes a different area of the canvas visible to the operator, but does not change the position of the view within the window.
Note: For a content or toolbar canvas-view, the window in which the canvas-view is displayed represents the view for that canvas. For a stacked canvas-view, the view size is controlled by setting the View Width and View Height properties.
Parameters and usage notes: see the manual or online help

3.2.4.9 SET_CANVAS_PROPERTY

Syntax: SET_CANVAS_PROPERTY(canvas_id, property, value);
SET_CANVAS_PROPERTY(canvas_id, property, x);
SET_CANVAS_PROPERTY(canvas_id, property, x, y);
SET_CANVAS_PROPERTY(canvas_name, property, value);
SET_CANVAS_PROPERTY(canvas_name, property, x);
SET_CANVAS_PROPERTY(canvas_name, property, x, y);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the given canvas property for the given canvas.
Parameters and usage notes: see the manual or online help

3.2.4.10 SET_VIEW_PROPERTY

Syntax: SET_VIEW_PROPERTY(view_id, property, value);
SET_VIEW_PROPERTY(view_id, property, x, y);
SET_VIEW_PROPERTY(view_name, property, value);
SET_VIEW_PROPERTY(view_name, property, x, y);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets a property for the indicated canvas-view. You can set only one property per call to the built-in. In other words, you cannot split the argument in such a way that the x coordinate applies to X_POS and the y coordinate applies to the HEIGHT.
Parameters: see Forms Reference Manual

3.2.4.11 SHOW_VIEW

Syntax: SHOW_VIEW(view_id);
SHOW_VIEW(view_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays the indicated canvas-view at the coordinates specified by the canvas-view's X Position and Y Position property settings. If the view is already displayed, SHOW_VIEW raises it in front of any other views in the same window.
Parameters:
view_id Specifies the unique ID that Oracle Forms assigns the view when it creates the object. Use the FIND_VIEW built-in to return the ID to an appropriately typed variable. The data type of the ID is ViewPort.
view_name Specifies the name that you gave the view when defining it. The data type of the name is CHAR.

3.2.5 Form Built-ins

3.2.5.1 BELL

Syntax: BELL;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the terminal bell to ring the next time the terminal screen synchronizes with the internal state of the form. This synchronization can occur as the result of internal processing or as the result of a call to the SYNCHRONIZE built-in subprogram.
Parameters: none

3.2.5.2 BREAK

Syntax: BREAK;
Built-in Type: unrestricted procedure

Enter Query Mode: yes
Description: Halts form execution and displays the Debugger, while the current form is running in debug mode. From the Debugger you can make selections to view the values of global and system variables. The BREAK built-in is primarily useful when you need to inspect the state of a form during trigger execution.
Parameters: none

3.2.5.3 CALL_FORM

Syntax: CALL_FORM(formmodule_name);
CALL_FORM(formmodule_name, display);
CALL_FORM(formmodule_name, display, switch_menu);
CALL_FORM(formmodule_name, display, switch_menu, query_mode)
CALL_FORM(formmodule_name, display, switch_menu, query_mode, paramlist_id);
CALL_FORM(formmodule_name, display, switch_menu, query_mode, paramlist_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Runs an indicated form while keeping the parent form active. Oracle Forms runs the called form with the same Runform preferences as the parent form. When the called form is exited Oracle Forms processing resumes in the calling form at the point from which you initiated the call to CALL_FORM.
Parameters: see Forms Reference Manual

3.2.5.4 CALL_INPUT

Syntax: CALL_INPUT;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Accepts and processes function key input from the operator. When CALL_INPUT is terminated, Oracle Forms resumes processing from the point at which the call to CALL_INPUT occurred.
Parameters: none
Restrictions: CALL_INPUT is included for compatibility with previous versions. You should not include this built-in in new applications.

3.2.5.5 CLEAR_FORM

Syntax: CLEAR_FORM;
CLEAR_FORM(commit_mode);
CLEAR_FORM(commit_mode, rollback_mode);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Causes Oracle Forms to remove all records from, or flush, the current form, and puts the input focus in the first item of the first block.
Parameters:
If the operator has made changes to records in the current form or any called form, and those records have not been posted or committed, Oracle Forms processes the records, following the directions indicated by the argument supplied for the following parameter: see Forms Reference Manual.

3.2.5.6 COMMIT_FORM

Syntax: COMMIT_FORM;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Causes Oracle Forms to update data in the database to match data in the form. Oracle Forms first validates the form, then, for each block in the form, deletes, inserts, and updates to the database, and performs a database commit. As a result of the database commit, the database releases all row and table locks.
If the operator has posted data to the database during the current Runform session, a call to the COMMIT_FORM built-in commits this data to the database.
Following a commit operation, Oracle Forms treats all records in all base-table blocks as if they are queried records from the database. Oracle Forms does not recognize changes that occur in triggers that fire during commit processing.
Restrictions: If you use a PL/SQL COMMIT statement in an anonymous block or a form-level procedure, Oracle Forms interprets that statement as a call to the COMMIT_FORM built-in.

3.2.5.7 DEBUG_MODE

Syntax: DEBUG_MODE;

Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Toggles debug mode on and off in a menu. When debug mode is True for a menu, Oracle Forms issues an appropriate message when a menu item command executes.
Parameters: none
Restrictions: The DEBUG_MODE applies only to a menu module. It does not place the form in Debug Mode.

3.2.5.8 ENTER

Syntax: ENTER;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Validates data in the current validation unit. (The default validation unit is Item.)
Parameters: none

3.2.5.9 ERASE

Syntax: ERASE(global_variable_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Removes an indicated global variable, so that it no longer exists, and releases the memory associated with the global variable. Globals always allocate 255 bytes of storage. To ensure that performance is not impacted more than necessary, always erase any global variable when it is no longer needed.
Parameters:
global_variable_name Specifies the name of a valid global variable.

3.2.5.10 EXECUTE_TRIGGER

Syntax: EXECUTE_TRIGGER(trigger_name);
Built-in Type: restricted procedure (if the user-defined trigger calls any restricted built-in subprograms)
Enter Query Mode: yes
Description: EXECUTE_TRIGGER executes an indicated trigger.
Note: EXECUTE_TRIGGER is not the preferred method for executing a user-named trigger: writing a user-named subprogram is the preferred method.
Parameters:
trigger_name Specifies the name of a valid user-named trigger.
Restrictions: Although you can use EXECUTE_TRIGGER to execute a built-in trigger as well as a user-named trigger, this usage is not recommended, because the default fail behavior follows a different rule than when invoked automatically by Oracle Forms as part of default processing. For example, in default processing, if the When-Validate-Item trigger fails, it raises an exception and stops the processing of the form. However, if the When-Validate-Item trigger fails when it is invoked by EXECUTE_TRIGGER, that failure does not stop the processing of the form, but only sets Form_Failure to FALSE on return from the EXECUTE_TRIGGER built-in.
Usage Notes: Because you cannot specify scope for this built-in, Oracle Forms always looks for the trigger starting at the lowest level, then working up.
To execute a built-in associated with a key, use the DO_KEY built-in instead of EXECUTE_TRIGGER. For example, rather than:
Execute_Trigger ('KEY-NEXT-ITEM');
Use instead:
Do_Key('NEXT_ITEM');

3.2.5.11 EXIT_FORM

Syntax: EXIT_FORM;
EXIT_FORM(commit_mode);
EXIT_FORM(commit_mode, rollback_mode);
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: The behavior of EXIT_FORM depends on the current mode:

- In most contexts, EXIT_FORM navigates "outside" the form. If there are changes in the current form that have not been posted or committed, Oracle Forms prompts the operator to commit before continuing EXIT_FORM processing.

- If the operator is in Enter Query mode, EXIT_FORM navigates out of Enter Query mode, not out of the form.

- During a CALL_INPUT, EXIT_FORM terminates the CALL_INPUT function.

Parameters and usage notes: see the manual or online help

3.2.5.12 FIND_FORM

Syntax: FIND_FORM(formmodule_name);
Built-in Type: unrestricted function
Returns: FormModule
Enter Query Mode: yes
Description: Searches the list of forms and returns a form module ID when it finds a valid form with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of Formmodule.
Parameters:
formmodule_name Specifies a valid CHAR form name.

3.2.5.13 FORM_FAILURE

Syntax: FORM_FAILURE;
Built-in Type: unrestricted function
Returns: BOOLEAN
Enter Query Mode: yes
Description: Returns a value that indicates the outcome of the action most recently performed during the current Runform session.

success	FALSE
failure	TRUE
fatal error	FALSE

If no action has executed in the current Runform session, FORM_FAILURE returns FALSE.
Use FORM_FAILURE to test the outcome of a built-in to determine further processing within any trigger. To get the correct results, you must perform the test immediately after the action executes. That is, another action should not occur prior to the test.
Note: "Another action" includes both built-ins and PL/SQL assignment statements. If another action occurs, FORM_FAILURE may not reflect the status of the built-in you are testing, but of the other, more recently executed action. A more accurate technique is, for example, when performing a COMMIT_FORM, to check that the SYSTEM.FORM_STATUS variable is set to 'QUERY' after the operation is done.
Parameters: none

3.2.5.14 FORM_FATAL

Syntax: FORM_FATAL;
Built-in Type: unrestricted function
Return Type: BOOLEAN
Enter Query Mode: yes
Description: Returns the outcome of the action most recently performed during the current Runform session.

success	FALSE
failure	FALSE
fatal error	TRUE

Use FORM_FATAL to test the outcome of a built-in to determine further processing within any trigger. To get the correct results, you must perform the test immediately after the action executes. That is, another action should not occur prior to the test.
Note: "Another action" includes both built-ins and PL/SQL assignment statements. If another action occurs, FORM_FATAL may not reflect the status of the built-in you are testing, but of the other, more recently executed action. A more accurate technique is, for example, when performing a COMMIT_FORM, to check that the SYSTEM.FORM_STATUS variable is set to 'QUERY' after the operation is done.
Parameters: none

3.2.5.15 FORM_SUCCESS

Syntax: FORM_SUCCESS;
Built-in Type: unrestricted function
Return Type: BOOLEAN
Enter Query Mode: yes
Description: Returns the outcome of the action most recently performed during the current Runform session. Use FORM_SUCCESS to test the outcome of a built-in to determine further processing within any trigger. To

get the correct results, you must perform the test immediately after the action executes. That is, another action should not occur prior to the test. "Another action" includes both built-ins and PL/SQL assignment statements. If another action occurs, FORM_SUCCESS may not reflect the status of the built-in you are testing, but of the other, more recently executed action.
Note: FORM_SUCCESS should not be used to test whether a COMMIT_FORM or POST built-in has succeeded. Because COMMIT_FORM may cause many other triggers to fire, when you evaluate FORM_SUCCESS it may not reflect the status of COMMIT_FORM but of some other, more recently executed built-in. A more accurate technique is to check that the SYSTEM.FORM_STATUS variable is set to 'QUERY' after the operation is done.
Parameters: none

3.2.5.16 GET_FORM_PROPERTY

Syntax: GET_FORM_PROPERTY(formmodule_id, property);
GET_FORM_PROPERTY(formmodule_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns information about the given form. If your application is a multi-form application, then you can call this built-in to return information about the calling form, as well as about the current, or called form.
Parameters: see the manual or on-line help

3.2.5.17 HELP

Syntax: HELP;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Displays the current item's hint message on the message line. If the hint message is already displayed, HELP displays the detailed help screen for the item.
Parameters: none

3.2.5.18 ID_NULL

See Alert Built-ins

3.2.5.19 NEW_FORM

Syntax: NEW_FORM(formmodule_name);
NEW_FORM(formmodule_name, rollback_mode);
NEW_FORM(formmodule_name, rollback_mode, query_mode);
NEW_FORM(formmodule_name, rollback_mode, query_mode, paramlist_id);
NEW_FORM(formmodule_name, rollback_mode, query_mode, paramlist_name);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Exits the current form and enters the indicated form. The calling form is terminated as the parent form. If the calling form had been called by a higher form, Oracle Forms keeps the higher call active and treats it as a call to the new form. Oracle Forms releases memory (such as database cursors) that the terminated form was using.
Oracle Forms runs the new form with the same Runform options as the parent form. If the parent form was a called form, Oracle Forms runs the new form with the same options as the parent form.
Parameters: see the manual or on-line help

3.2.5.20 OPEN_FORM

Syntax: OPEN_FORM(form_name);
OPEN_FORM(form_name,activate_mode);
OPEN_FORM(form_name,activate_mode,session_mode);
OPEN_FORM(form_name,activate_mode,session_mode, paramlist_name);
OPEN_FORM(form_name,activate_mode,session_mode, paramlist_id);
Built-in Type: restricted procedure (cannot be called in Enter Query mode)
Enter Query Mode: no
Description: Opens the indicated form. Call OPEN_FORM to create multiple-form applications, that is, applications that open more than one form at the same time.
Parameters: see the manual or on-line help

3.2.5.21 POST

Syntax: POST;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Writes data in the form to the database, but does not perform a database commit. Oracle Forms first validates the form. If there are changes to post to the database, for each block in the form Oracle Forms writes deletes, inserts, and updates to the database.
Any data that you post to the database is committed to the database by the next COMMIT_FORM that executes during the current Runform session. Alternatively, this data can be rolled back by the next CLEAR_FORM.
Parameters: none

3.2.5.22 REDISPLAY

Syntax: REDISPLAY;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Redraws the screen. This clears any existing system messages displayed on the screen.
Parameters: none

3.2.5.23 REPLACE_MENU

Syntax: REPLACE_MENU;
REPLACE_MENU(menu_module_name);
REPLACE_MENU(menu_module_name, menu_type);
REPLACE_MENU(menu_module_name, menu_type, starting_menu_name);
REPLACE_MENU(menu_module_name, menu_type, starting_menu, group_name);
REPLACE_MENU(menu_module_name, menu_type, starting_menu, group_name, use_file);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Replaces the current menu with the specified menu, but does not make the new menu active. REPLACE_MENU also allows you to change the way the menu displays and the role.
Because REPLACE_MENU does not make the new menu active, Oracle Forms does not allow the menu to obscure any part of the active canvas. Therefore, all or part of the menu does not appear on the screen if the active canvas would cover it.
Usage Notes: REPLACE_MENU replaces the menu for all windows in the application. If you are using CALL_FORM, REPLACE_MENU will replace the menu for both the calling form and the called form with the specified menu.
Parameters: see the manual or on-line help

3.2.5.24 SET_FORM_PROPERTY

3.2.5.25 SHOW_KEYS

Syntax: SHOW_KEYS;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays the Show Keys screen. When the operator presses a function key, Oracle Forms redisplays the form as it was before invoking the SHOW_KEYS built-in.
Parameters: none

3.2.5.26 SHOW_MENU

Syntax: SHOW_MENU;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays the current menu if it is not currently displayed. It does not make the menu active. Because SHOW_MENU does not make the menu active, Oracle Forms does not allow the menu to obscure any part of the current canvas. Therefore, all or part of the menu does not appear on the screen if the current canvas would cover it.
Parameters: none
Restrictions: Only for use in character mode environments.

3.2.5.27 SYNCHRONIZE

Syntax: SYNCHRONIZE;

Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Synchronizes the terminal screen with the internal state of the form. That is, SYNCHRONIZE updates the screen display to reflect the information that Oracle Forms has in its internal representation of the screen.
Parameters: none
Restrictions: see the manual or on-line help

3.2.6 Item Built-ins

3.2.6.1 CHECKBOX_CHECKED

Syntax: CHECKBOX_CHECKED(item_id);
CHECKBOX_CHECKED(item_name);
Built-in Type: unrestricted function
Returns: BOOLEAN
Enter Query Mode: yes
Description: A call to the CHECKBOX_CHECKED function returns a BOOLEAN value indicating the state of the given check box. If the item is not a check box, Oracle Forms returns an error. A call to GET_ITEM_PROPERTY(item_name, ITEM_TYPE) can be used to verify the item type before calling CHECKBOX_CHECKED.
To set the value of a check box programmatically, assign a valid value to the check box using standard bind variable syntax.
Parameters: see the manual or no-line help

3.2.6.2 CLEAR_EOL

Syntax: CLEAR_EOL;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Clears the current text item's value from the current cursor position to the end of the line.

3.2.6.3 CLEAR_ITEM

Syntax: CLEAR_ITEM;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Clears the value from the current text item, regardless of the current cursor position, and changes the text item value to NULL.

3.2.6.4 CONVERT_OTHER_VALUE

Syntax: CONVERT_OTHER_VALUE(item_id);
CONVERT_OTHER_VALUE(item_name);
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Converts the current value of a check box, radio group, or list item to the value associated with the current check box state (Checked/Unchecked), or with the current radio group button or list item element.
Parameters: see the manual or on-line help

3.2.6.5 COPY

Syntax: COPY(source, destination);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Copies a value from one item or variable into another item or global variable. Use specifically to write a value into an item that is referenced through the NAME_IN built-in. COPY exists for two reasons:

- You cannot use standard PL/SQL syntax to set a referenced item equal to a value.
- You might intend to programmatically place characters such as relational operators in NUMBER and DATE fields while a form is in Enter Query mode.

Parameters: see the manual or on-line help

3.2.6.6 COPY_REGION

Syntax: COPY_REGION;
Built-in Type: restricted procedure

Enter Query Mode: yes
Description: Copies the selected text from the screen and stores it in the paste buffer until you cut or copy another selected region.
Parameters: none
Usage Notes: Use COPY_REGION, as well as the other editing functions, on text only. The cut and copy functions transfer the selected region of text into the system clipboard until you indicate the paste target. At that time, the cut or copied text is pasted onto the target location.

3.2.6.7 CUT_REGION

Syntax: CUT_REGION;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Removes a selected region of text from the screen and stores it in the paste buffer until you cut or copy another selected region.
Parameters: none
Usage Notes: Use CUT_REGION, as well as the other editing functions, on text only. The cut and copy functions transfer the selected region of text into the system clipboard until you indicate the paste target. At that time, the cut or copied text is pasted onto the target location.

3.2.6.8 DEFAULT_VALUE

Syntax: DEFAULT_VALUE(value_string,variable_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Copies an indicated value to an indicated variable if the variable's current value is NULL. If the variable's current value is not NULL, DEFAULT_VALUE does nothing. Therefore, for text items this built-in works identically to using the COPY built-in on a NULL item. If the variable is an undefined global variable, Oracle Forms creates the variable.
Parameters: see the manual or on-line help

3.2.6.9 DISPLAY_ITEM

Syntax: DISPLAY_ITEM(item_id, attribute);
DISPLAY_ITEM(item_name, attribute);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Modifies an item's appearance by assigning a specified display attribute to the item.
You can reference any item in the current form. Note that DISPLAY_ITEM only affects the display of the current instance of the item; other instances of the specified item are not affected. This means that if you specify a display change for an item that exists in a multi-record block, DISPLAY_ITEM only changes the instance of that item that belongs to the block's current record. If you want to change all instances of an item in a multi-record block, use SET_ITEM_PROPERTY.
Any change made by a DISPLAY_ITEM built-in is effective until another DISPLAY_ITEM references the same item, or that instance of the item is removed (e.g., through a CLEAR_RECORD or a query), or the current form is exited.
Parameters: see the manual or on-line help

3.2.6.10 DUPLICATE_ITEM

Syntax: DUPLICATE_ITEM;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Assigns the current item the same value as the instance of this item in the previous record.
Parameters: none
Restrictions: A previous record must exist in your current session, or Oracle Forms returns error FRM-41803: No previous record to copy value from.

3.2.6.11 EDIT_TEXTITEM

Syntax: EDIT_TEXTITEM;
EDIT_TEXTITEM(x, y);
EDIT_TEXTITEM(x, y, width, height);
Built-in Type: restricted procedure
Enter Query Mode: yes

Description: Invokes the Runform item editor for the current text item and puts the form in Edit mode.
Parameters: see the manual or on-line help

3.2.6.12 FIND_ITEM

Syntax: FIND_ITEM(block.item_name);
Built-in Type: unrestricted function
Returns: Item
Enter Query Mode: yes
Description: Searches the list of items in a given block and returns an item ID when it finds a valid item with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of Item.
Parameters: see the manual or on-line help

3.2.6.13 GET_ITEM_PROPERTY

Syntax: GET_ITEM_PROPERTY(item_id, property);
GET_ITEM_PROPERTY(item_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns information about a specified item. You may be able to get but not set certain object properties.
Parameters: see the manual or on-line help

3.2.6.14 GET_RADIO_BUTTON_PROPERTY

Syntax: GET_RADIO_BUTTON_PROPERTY(item_id, button_name, property);
GET_RADIO_BUTTON_PROPERTY(item_name, button_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns information about a specified radio button.
Parameters: see the manual or on-line help

3.2.6.15 GO_ITEM

Syntax: GO_ITEM(item_id);
GO_ITEM(item_name);
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: GO_ITEM navigates to an indicated item. GO_ITEM succeeds even if the target item has the Navigable property set to False.
Parameters: see the manula or on-line help

3.2.6.16 ID_NULL

See Alert Built-ins

3.2.6.17 IMAGE_ZOOM

Syntax: IMAGE_ZOOM(image_id, zoom_type);
IMAGE_ZOOM(image_name, zoom_type);
IMAGE_ZOOM(image_id, zoom_type, zoom_factor);
IMAGE_ZOOM(image_name, zoom_type, zoom_factor);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Zooms the image in or out using the effect specified in zoom_type and the amount specified in zoom_factor.
Parameters: see the manual or on-line help

3.2.6.18 NAME_IN

Syntax: NAME_IN(variable_name);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes

Description: Returns the value of the indicated variable. The returned value is in the form of a character string. However, you can use NAME_IN to return numbers and dates as character strings and then convert those strings to the appropriate data types. You can use the returned value as you would use any value within an executable statement.
If you nest the NAME_IN function, Oracle Forms evaluates the individual NAME_IN functions from the innermost one to the outermost one.
Parameters: see the manual or on-line help

3.2.6.19 NEXT_ITEM

Syntax: NEXT_ITEM;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Navigates to the navigable item with the next higher sequence number than the current item. If there is no such item, NEXT_ITEM navigates to the item with the lowest sequence number. If there is no such item, NEXT_ITEM navigates to the current item.
If the validation unit is the item, NEXT_ITEM validates any fields with sequence numbers greater than the current item or less than the target item.
The function of NEXT_ITEM from the last navigable item in the block depends on the setting of the Navigation Style block property. The valid settings for Navigation Style include:
Same Record (Default): A Next Item operation from a block's last item moves the input focus to the first navigable item in the block, in that same record.
Change Record: A Next Item operation from a block's last item moves the input focus to the first navigable item in the block, in the next record. If the current record is the last record in the block and there is no open query, Oracle Forms creates a new record. If there is an open query in the block (the block contains queried records), Oracle forms retrieves additional records as needed.
Change Block: A Next Item operation from a block's last item moves the input focus to the first navigable item in the first record of the next block.
Parameters: none

3.2.6.20 NEXT_KEY

Syntax: NEXT_KEY;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Navigates to the enabled and navigable primary key item with the next higher sequence number than the current item. If there is no such item, NEXT_KEY navigates to the enabled and navigable primary key item with the lowest sequence number. If there is no primary key item in the current block, an error occurs.
If the validation unit is the item, NEXT_KEY validates any fields with sequence numbers greater than the current item or less than the target item.
Parameters: none

3.2.6.21 PASTE_REGION

Syntax: PASTE_REGION;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Pastes the contents of the clipboard (i.e., the selected region that was cut or copied most recently), positioning the upper left corner of the pasted area at the cursor position.
Parameters: none

3.2.6.22 PREVIOUS_ITEM

Syntax: PREVIOUS_ITEM;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Navigates to the navigable item with the next lower sequence number than the current item. If there is no such item, PREVIOUS_ITEM navigates to the navigable item with the highest sequence number. If there is no such item, PREVIOUS_ITEM navigates to the current item.
The function of PREVIOUS_ITEM from the first navigable item in the block depends on the setting of the Navigation Style block property. The valid settings for Navigation Style include:
Same Record (Default): A Previous Item operation from a block's first item moves the input focus to the last navigable item in the block, in that same record.

Change Record: A Previous Item operation from a block's first item moves the input focus to the last navigable item in the block, in the previous record.
Change Block: A Previous Item operation from a block's first item moves the input focus to the last navigable item in the current record of the previous block.
Parameters: none

3.2.6.23 READ_IMAGE_FILE

Syntax: READ_IMAGE_FILE(file_name, file_type, item_id);
READ_IMAGE_FILE(file_name, file_type, item_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Reads an image of the given type from the given file and displays it in the Oracle Forms image item.
Parameters: see the manual or on-line help

3.2.6.24 SELECT_ALL

Syntax: SELECT_ALL;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Selects the text in the current item. Call this procedure prior to issuing a call to CUT_REGION or COPY_REGION, when you want to cut or copy the entire contents of a text item.
Parameters: none

3.2.6.25 SET_ITEM_PROPERTY

Syntax: SET_ITEM_PROPERTY(item_id, property, value);
SET_ITEM_PROPERTY(item_name, property, value);
SET_ITEM_PROPERTY(item_id, property, x, y);
SET_ITEM_PROPERTY(item_id, property, x);
SET_ITEM_PROPERTY(item_name, property, x, y);
SET_ITEM_PROPERTY(item_name, property, x);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Modifies all instances of an item in a block by changing a specified item property. You may be able to get but not set certain object properties.
Parameters: see the manual or on-line help

3.2.6.26 SET_RADIO_BUTTON_PROPERTY

Syntax: SET_RADIO_BUTTON_PROPERTY(item_id, button_name, property, value);
SET_RADIO_BUTTON_PROPERTY(item_id, button_name, property, x, y);
SET_RADIO_BUTTON_PROPERTY(item_name, button_name, property, x, y);
SET_RADIO_BUTTON_PROPERTY(item_name, button_name, property,value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the given property for a radio button that is part of the given radio group specified by the item_name or item_id.
Parameters: see the manual or on-line help

3.2.6.27 WRITE_IMAGE_FILE

Syntax: WRITE_IMAGE_FILE(file_name, file_type, item_id);
WRITE_IMAGE_FILE(file_name, file_type, item_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Writes the image stored in an Oracle Forms image item into the file.
Parameters: see the manual or on-line help

3.2.7 List Item Built-ins

3.2.7.1 ADD_LIST_ELEMENT

Syntax: ADD_LIST_ELEMENT(list_name, list_index, list_label, list_value);
ADD_LIST_ELEMENT(list_id, list_index, list_label, list_value);

Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Adds a single element to a list item.
Parameters: see the manual or on-line help

3.2.7.2 CLEAR_LIST

Syntax: CLEAR_LIST(list_id);
CLEAR_LIST(list_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Clears all elements from a list item. After Oracle Forms clears the list, the list will contain only one element (the null element), regardless of the item's Required property.
Parameters: see the manual or on-line help

3.2.7.3 DELETE_LIST_ELEMENT

Syntax: DELETE_LIST_ELEMENT(list_name, list_index);
DELETE_LIST_ELEMENT(list_id, list_index);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Deletes a specific list element from a list item.
Parameters: see the manual or on-line help

3.2.7.4 GET_LIST_ELEMENT_COUNT

Syntax: GET_LIST_ELEMENT_COUNT(list_id);
GET_LIST_ELEMENT_COUNT(list_name);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the total number of list item elements in a list, including elements with NULL values.
Parameters: see the manual or on-line help

3.2.7.5 GET_LIST_ELEMENT_LABEL

Syntax: GET_LIST_ELEMENT_LABEL(list_id, list_name, list_index);
GET_LIST_ELEMENT_LABEL(list_name, list_index);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns information about the requested list element label.
Parameters: see the manual or on-line help

3.2.7.6 GET_LIST_ELEMENT_VALUE

Syntax: GET_LIST_ELEMENT_VALUE(list_id, list_index);
GET_LIST_ELEMENT_VALUE(list_name, list_index);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the value associated with the specified list item element.
Parameters: see the manual or on-line help

3.2.7.7 POPULATE_LIST

Syntax: POPULATE_LIST(list_id, recgrp_id);
POPULATE_LIST(list_id, recgrp_name);
POPULATE_LIST(list_name, recgrp_id);
POPULATE_LIST(list_name, recgrp_name);
Built-in Type unrestricted procedure
Enter Query Mode: yes
Description: Removes the contents of the current list and populates the list with the values from a record group. The record group must be created at runtime and it must have the following two column (CHAR) structure:

Column 1:	Column 2:
the list label	the list value

Parameters: see the manual or on-line help

3.2.7.8 RETRIEVE_LIST

Syntax: RETRIEVE_LIST(list_id, recgrp_name);
RETRIEVE_LIST(list_id, recgrp_id);
RETRIEVE_LIST(list_name, recgrp_id);
RETRIEVE_LIST(list_name, recgrp_name);
Built-in Type: unrestricted procedure
Returns: CHAR
Enter Query Mode: yes
Description: Retrieves and stores the contents of the current list into the specified record group. The target record group must have the following two-column (CHAR) structure:

Column 1:	Column 2:
the list label	the list value

Storing the contents of a list item allows you to restore the list with its former contents.
Parameters: see the manual or on-line help

3.2.8 Menu Built-ins

3.2.8.1 APPLICATION_PARAMETER

Syntax: APPLICATION_PARAMETER;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays all the parameters associated with the current menu, and their current values, in the Enter Parameter Values dialog box.
Failure: If no parameters are defined for the current menu, Oracle Forms issues error message FRM-10201: No parameters needed.

3.2.8.2 BACKGROUND_MENU

Syntax: BACKGROUND_MENU{1|2|3|4|5|6|7|8|9|10};
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Executes the designer-specified menu item n from the background menu.
Parameters: see the manual or on-line help

3.2.8.3 FIND_MENU_ITEM

Syntax: FIND_MENU_ITEM(menu_name.menu_item_name);
Built-in Type: unrestricted function
Returns: MenuItem
Enter Query Mode: yes
Description: Searches the list of menu items and returns a menu item ID when it finds a valid menu item with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of MenuItem.
Parameters: see the manual or on-line help

3.2.8.4 GET_MENU_ITEM_PROPERTY

Syntax: GET_MENU_ITEM_PROPERTY(menuitem_id, property);
GET_MENU_ITEM_PROPERTY(menu_name.menuitem_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the state of the menu item given the specific property. You can use this built-in function to get the state and then you can change the state of the property with the SET_MENU_ITEM_PROPERTY built-in.
Parameters: see the manual or on-line help

3.2.8.5 HIDE_MENU

Syntax: HIDE_MENU;
Built-in Type: unrestricted procedure
Enter Query Mode: yes

Description: On character mode platforms, makes the current menu disappear if it is currently displayed, uncovering any part of the form display that the menu had covered. The menu will redisplay if the SHOW_MENU built-in is invoked or the operator presses [Menu].
Parameters: none

3.2.8.6 ITEM_ENABLED

Syntax: ITEM_ENABLED(mnunam,itmnam);
Built-in Type: unrestricted function
Returns: BOOLEAN
Enter Query Mode: yes
Description: Returns the Boolean value TRUE when the menu item is enabled. Returns the Boolean value FALSE when the menu item is disabled.
Note: ITEM_ENABLED is equivalent to GET_MENU_ITEM_PROPERTY (MENU_ITEM, ENABLED).
Parameters: see the manual or on-line help

3.2.8.7 MAIN_MENU

Syntax: MAIN_MENU;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: MAIN_MENU navigates to the main menu of the current application.
Parameters: none

3.2.8.8 MENU_CLEAR_FIELD

Syntax: MENU_CLEAR_FIELD;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: MENU_CLEAR_FIELD clears the current field's value from the current cursor position to the end of the field. If the current cursor position is to the right of the last nonblank character, MENU_CLEAR_FIELD clears the entire field, making its value NULL.
Parameters: none

3.2.8.9 MENU_NEXT_FIELD

Syntax: MENU_NEXT_FIELD;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: MENU_NEXT_FIELD navigates to the next field in an Enter Parameter Values dialog.
Parameters: none

3.2.8.10 MENU_PARAMETER

Syntax: MENU_PARAMETER;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: MENU_PARAMETER displays all the parameters associated with the current menu, and their current values, in the Enter Parameter Values dialog.
Parameters: none
Restrictions: You must be in an Enter Parameter Values dialog box.

3.2.8.11 MENU_PREVIOUS_FIELD

Syntax: MENU_PREVIOUS_FIELD;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: MENU_PREVIOUS_FIELD returns to the previous field in an Enter Parameter Values dialog.
Parameters: none

3.2.8.12 MENU_REDISPLAY

Syntax: MENU_REDISPLAY;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: This procedure redraws the screen in a menu.
Parameters: none
Restrictions: You must be on a character mode or block mode platform.

3.2.8.13 MENU_SHOW_KEYS

Syntax: MENU_SHOW_KEYS;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: MENU_SHOW_KEYS displays the Show Keys screen for the menu module at runtime.
Parameters: none
Restrictions: MENU_SHOW_KEYS is available in any context.

3.2.8.14 NEXT_MENU_ITEM

Syntax: NEXT_MENU_ITEM;
Built-in Type: restricted procedure
Description: Navigates to the next menu item in the current menu.
Parameters: none
Restrictions: NEXT_MENU_ITEM is available only in a custom menu running in the full-screen menu display style.

3.2.8.15 PREVIOUS_MENU

Syntax: PREVIOUS_MENU;
Built-in Type: restricted procedure
Description: PREVIOUS_MENU navigates to the previously active item in the previous menu.
Parameters: none
Restrictions: PREVIOUS_MENU applies only in full-screen and bar menus.

3.2.8.16 PREVIOUS_MENU_ITEM

Syntax: PREVIOUS_MENU_ITEM;
Built-in Type: restricted procedure
Description: PREVIOUS_MENU_ITEM navigates to the previous menu item in the current menu.
Parameters: none
Restrictions: PREVIOUS_MENU_ITEM applies only in full-screen menus.

3.2.8.17 QUERY_PARAMETER

Syntax: QUERY_PARAMETER(parameter_string);
Built-in Type: unrestricted procedure
Description: Displays the Query Parameter dialog showing the current values of the specified substitution parameters. Operators can set the value of any parameter you include in the list.
The Query Parameter dialog is modal, and control does not return to the calling trigger or procedure until the operator either accepts or cancels the dialog. This means that any PL/SQL statements that follow the call to QUERY_PARAMETER are not executed until the Query Parameter dialog is dismissed.
Parameters: see the manual or on-line help

3.2.8.18 SET_INPUT_FOCUS

Syntax: SET_INPUT_FOCUS(MENU);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the input focus on the menu of the current form. Once trigger processing is completed, Oracle Forms activates the menu.
Parameters: MENU
Restrictions: Only for use in character mode and block mode environments.

3.2.8.19 SET_MENU_ITEM_PROPERTY

Syntax: SET_MENU_ITEM_PROPERTY(menuitem_id, property, value);
SET_MENU_ITEM_PROPERTY(menu_name.menuitem_name, property, value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Modifies the given properties of a menu item.
Parameters: see the manual or on-line help

3.2.8.20 SHOW_BACKGROUND_MENU

Syntax: SHOW_BACKGROUND_MENU;
Built-in Type: unrestricted procedure

Enter Query Mode: yes
Description: Displays the background menu.
Parameters: none
Restrictions: If you issue a call to this built-in when no background menu is defined for the current application, Oracle Forms issues error message FRM-10207: No background menu present.

3.2.8.21 TERMINATE

Syntax: TERMINATE;
Built-in Type: restricted function
Description: TERMINATE terminates input in a form or dialog box. This function is equivalent to the operator pressing [ACCEPT].
Parameters: none
Restrictions: Terminate applies only in the Enter Parameter Values dialog.

3.2.8.22 WHERE_DISPLAY

Syntax: WHERE_DISPLAY;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Toggles the Where menu navigation option on and off. In a full-screen menu, the Where option displays information about the operator's current location in the menu hierarchy.
Parameters: none
Restrictions: WHERE_DISPLAY is valid only in a full-screen menu.

3.2.9 Messages Built-ins

3.2.9.1 CLEAR_MESSAGE

Syntax: CLEAR_MESSAGE;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Removes the current message from the screen message area.

3.2.9.2 DBMS_ERROR_CODE

Syntax: DBMS_ERROR_CODE;
Built-in Type: unrestricted function
Enter Query Mode: yes
Description: Returns the error number of the last database error that was detected.
Parameters: none
Usage Notes: For recursive errors, this built-in returns the code of the first message in the stack, so the error text must be parsed for numbers of subsequent messages.

3.2.9.3 DBMS_ERROR_TEXT

Syntax: DBMS_ERROR_TEXT;
Built-in Type: unrestricted function
Enter Query Mode: yes
Description: Returns the message number (such as ORA-01438) and message text of the database error.
Parameters: none
Usage Notes: You can use this function to test database error messages during exception handling routines. DBMS_ERROR_TEXT returns the entire sequence of recursive errors.

3.2.9.4 DISPLAY_ERROR

Syntax: DISPLAY_ERROR;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays the Display Error screen if there is a logged error. When the operator presses a function key while viewing the Display Error screen, Oracle Forms redisplays the form. If there is no error to display when you call this built-in, Oracle Forms ignores the call and does not display the DISPLAY ERROR screen.
Parameters: none

3.2.9.5 ERROR_CODE

Syntax: ERROR_CODE;

Built-in Type: unrestricted function
Enter Query Mode: yes
Description: Returns the error number of the Oracle Forms error.
Parameters: none

3.2.9.6 ERROR_TEXT

Syntax: ERROR_TEXT;
Built-in Type: unrestricted function
Enter Query Mode: yes
Description: Returns the message text of the Oracle Forms error.
Parameters: none
Usage Notes: You can use this function to test error messages during exception handling subprograms.

3.2.9.7 ERROR_TYPE

Syntax: ERROR_TYPE;
Built-in Type: unrestricted function
Returns: ERROR_TYPE returns one of the following values for the error message type:

FRM Indicates an Oracle Forms error.
ORA Indicates an ORACLE error.

Enter Query Mode: yes
Description: Returns the error message type for the action most recently performed during the current Runform session.
Parameters: none
Usage Notes: You can use this function to do one of the following:

- test the outcome of a user action, such as pressing a key, to determine processing within an On-Error trigger
- test the outcome of a built-in to determine further processing within any trigger

To get the correct results in either type of test, you must perform the test immediately after the action executes, before any other action occurs.

3.2.9.8 GET_MESSAGE

Syntax: GET_MESSAGE;
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the current message, regardless of type.
Parameters: none
Restrictions: GET_MESSAGE is only instantiated when a message is directed to the display device, either by Oracle Forms or by a call to the MESSAGE built-in. If you redirect messages using the On-Message trigger, a call to GET_MESSAGE does not return a value. Refer to the On-Message trigger description in Chapter 2, "Triggers" for more information.

3.2.9.9 MESSAGE

Syntax: MESSAGE(message_string, user_response);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays specified text on the message line.
Parameters: see the manual or on-line help
Restrictions: Message_string can be up to 200 characters long. Note, however, that several factors affect the maximum number of characters that can be displayed, including the current font and the limitations of the runtime window manager.

3.2.9.10 MESSAGE_CODE

Syntax: MESSAGE_CODE;
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Returns a message number for the message that Oracle Forms most recently generated during the current Runform session. MESSAGE_CODE returns zero at the beginning of a session, before Oracle Forms generates any messages.

Use MESSAGE_CODE to test the outcome of a user action (e.g., pressing a key) to determine processing within an On-Message trigger.
Refer to the Messages appendix for a list of messages and message numbers.
Parameters: none

3.2.9.11 MESSAGE_TEXT

Syntax: MESSAGE_TEXT;
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns message text for the message that Oracle Forms most recently generated during the current Runform session. MESSAGE_TEXT returns NULL at the beginning of a session, before Oracle Forms generates any messages.
Use MESSAGE_TEXT to test the outcome of a user action (e.g., pressing a key) to determine processing within an On-Message trigger.
Note: If your applications must be supported in more than one language, use the MESSAGE_CODE built-in instead of the MESSAGE_TEXT built-in. Referencing message codes rather than message text is particularly useful in applications that provide national language support.
Parameters: none

3.2.9.12 MESSAGE_TYPE

Syntax: MESSAGE_TYPE;
Built-in Type: unrestricted function
Returns: CHAR
MESSAGE_TYPE returns one of three values for the message type:

FRM	Indicates that an Oracle Forms message was generated.
ORA	Indicates that an ORACLE message was generated.
NULL	Indicates that Oracle Forms has not yet issued any messages during the session.

Enter Query Mode: yes
Description: Returns a message type for the message that Oracle Forms most recently generated during the current Runform session.
Use MESSAGE_TYPE to test the outcome of a user action (e.g., pressing a key) to determine processing within an On-Message trigger.
Parameters: none

3.2.10 Miscellaneous Built-ins

3.2.10.1 CREATE_TIMER

See the manual or on-line help.

3.2.10.2 DELETE_TIMER

See the manual or on-line help.

3.2.10.3 FIND_EDITOR

See the manual or on-line help.

3.2.10.4 FIND_LOV

Syntax: FIND_LOV(LOV_name);
Built-in Type: unrestricted function
Returns: LOV
Enter Query Mode: yes
Description: Searches the list of LOVs and returns an LOV ID when it finds a valid LOV with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of LOV.
Parameters:

LOV_name	Specifies the valid CHAR LOV name.

3.2.10.5 FIND_TIMER

See the manual or on-line help.

3.2.10.6 GET_LOV_PROPERTY

3.2.10.7 ID_NULL

See Alert built-ins.

3.2.10.8 LIST_VALUES

See the manual or on-line help.

3.2.10.9 SET_LOV_COLUMN_PROPERTY

See the manual or on-line help.

3.2.10.10 SET_LOV_PROPERTY

See the manual or on-line help.

3.2.10.11 SET_TIMER

See the manual or on-line help.

3.2.10.12 SHOW_EDITOR

See the manual or on-line help.

3.2.10.13 SHOW_LOV

Syntax: SHOW_LOV(lov_id);
SHOW_LOV(lov_id, x, y);
SHOW_LOV(lov_name);
SHOW_LOV(lov_name, x, y);
Built-in Type: restricted function
Returns: BOOLEAN
Enter Query Mode: yes
Description: Displays a list of values (LOV) window at the given coordinates, and returns TRUE if the operator selects a value from the list, and FALSE if the operator Cancels and dismisses the list.
Parameters: see the manual or on-line help
Usage Notes: Because SHOW_LOV is a restricted built-in, when you use it to display an LOV, Oracle Forms ignores the LOV's Auto-Skip property.
If you want to move the cursor to the next navigable item, use the LIST_VALUES built-in. LIST_VALUES is an unrestricted built-in.
Restrictions: If the lov_name argument is not supplied and there is no LOV associated with the current item, Oracle Forms issues an error.
If the record group underlying the LOV contains 0 records, the BOOLEAN return value for SHOW_LOV will be FALSE.

3.2.10.14 VALIDATE

Syntax: VALIDATE(validation_scope);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: VALIDATE forces Oracle Forms to immediately execute validation processing for the indicated validation scope.
Parameters: see the manual or on-line help

3.2.11 Multiple-form Application Built-ins

3.2.11.1 CLOSE_FORM

Syntax: CLOSE_FORM(form_name);
CLOSE_FORM(form_id);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: In a multiple-form application, closes the indicated form. When the indicated form is the current form, CLOSE_FORM is equivalent to EXIT_FORM.
Parameters: see the manual or on-line help

Restrictions: You cannot close a form that is currently disabled as a result of having issued CALL_FORM to invoke a modal called form.

3.2.11.2 GO_FORM

Syntax: GO_FORM(form_name);
GO_FORM(form_id);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: In a multiple-form application, navigates from the current form to the indicated target form. When navigating with GO_FORM, no validation occurs and no triggers fire except WHEN-WINDOW-DEACTIVATED, which fires for the form that initiates navigation, and WHEN-WINDOW-ACTIVATED, which fires for the target window in the target form.
Attempting to navigate to a form that has not yet been opened raises an error.
Parameters: see the manual or on-line help
Restrictions: The target form cannot be a form that is currently disabled as a result of having invoked another form with CALL_FORM.

3.2.11.3 NEXT_FORM

Syntax: NEXT_FORM;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: In a multiple-form application, navigates to the independent form with the next highest sequence number. (Forms are sequenced in the order they were invoked at runtime.) If there is no form with a higher sequence number, NEXT_FORM navigates to the form with the lowest sequence number. If there is no such form, the current form remains current.
When navigating with NEXT_FORM, no validation occurs and no triggers fire except WHEN-WINDOW-DEACTIVATED, which fires for the form that initiates navigation, and WHEN-WINDOW-ACTIVATED, which fires for the target form.
Restrictions: The target form cannot be a form that is currently disabled as a result of having invoked another form with CALL_FORM.

3.2.11.4 OPEN_FORM

Syntax OPEN_FORM(form_name);
OPEN_FORM(form_name,activate_mode);
OPEN_FORM(form_name,activate_mode,session_mode);
OPEN_FORM(form_name,activate_mode,session_mode, paramlist_name);
OPEN_FORM(form_name,activate_mode,session_mode, paramlist_id);
Built-in Type: restricted procedure (cannot be called in Enter Query mode)
Enter Query Mode: no
Description: Opens the indicated form. Call OPEN_FORM to create multiple-form applications, that is, applications that open more than one form at the same time.
Parameters: see the manual or on-line help
Restrictions and usage notes: see the manual and on-line help

3.2.11.5 PREVIOUS_FORM

Syntax: PREVIOUS_FORM;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: In a multiple-form application, navigates to the form with the next lowest sequence number. (Forms are sequenced in the order they were invoked at runtime.) If there is no form with a lower sequence number, PREVIOUS_FORM navigates to the form with the highest sequence number. If there is no such form, the current form remains current.
When navigating with PREVIOUS_FORM, no validation occurs and no triggers fire except WHEN-WINDOW-DEACTIVATED, which fires for the form that initiates navigation, and WHEN-WINDOW-ACTIVATED, which fires for the target form.
Restrictions: The target form cannot be a form that is currently disabled as a result of having invoked another form with CALL_FORM

3.2.12 OLE Built-ins

See the manual or on-line help.

3.2.13 Parameter List Built-ins

See the manual or on-line help.

3.2.14 Query Built-ins

3.2.14.1 ABORT_QUERY

Syntax: ABORT_QUERY;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Closes a query that is open in the current block.
A query is open between the time the SELECT statement is issued and the time when all the rows have been fetched from the database. In particular, a query is not open when the form is in Enter Query mode, because the SELECT statement has not yet been issued.
Parameters: none
Usage Notes:
ABORT_QUERY is not the equivalent of the Query, Cancel runtime default menu command. It does not prevent the initial fetch from the database, but rather interrupts fetch processing, thus preventing subsequent fetches.
Restrictions: see the manual or on-line help

3.2.14.2 COUNT_QUERY

Syntax: COUNT_QUERY;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: In an On-Count trigger, performs the default Oracle Forms processing for identifying the number of rows that a query will retrieve for the current block, and clears the current block. If there are changes to commit in the block, Oracle Forms prompts the operator to commit them during COUNT_QUERY processing. Oracle Forms returns the following message as a result of a valid call to COUNT_QUERY:
FRM-40355: Query will retrieve <number> records.
This built-in is included primarily for applications that will run against a non-ORACLE data source.
Parameters: none
Restrictions: Valid only in triggers that allow restricted built-ins.

3.2.14.3 ENTER_QUERY

Syntax: ENTER_QUERY;
ENTER_QUERY(keyword_one);
ENTER_QUERY(keyword_two);
ENTER_QUERY(keyword_one, keyword_two);
ENTER_QUERY(keyword_one, keyword_two, locking);
Built-in Type: restricted procedure
Enter Query Mode: yes (to redisplay the example record from the last query executed in the block)
Description: The behavior of ENTER_QUERY varies depending on any parameters you supply.
Parameters and restrictions: see the manual or on-line help

3.2.14.4 EXECUTE_QUERY

Syntax: EXECUTE_QUERY;
EXECUTE_QUERY(keyword_one);
EXECUTE_QUERY(keyword_two);
EXECUTE_QUERY(keyword_one, keyword_two);
EXECUTE_QUERY(keyword_one, keyword_two, locking);
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Flushes the current block, opens a query, and fetches a number of selected records. If there are changes to commit, Oracle Forms prompts the operator to commit them before continuing EXECUTE_QUERY processing.
Parameters and restrictions: see the manual or on-line help

3.2.15 Record Built-ins

3.2.15.1 CHECK_RECORD_UNIQUENESS

Syntax: CHECK_RECORD_UNIQUENESS;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: When called from an On-Check-Unique trigger, initiates the default Oracle Forms processing for checking the primary key uniqueness of a record.
This built-in is included primarily for applications that will run against a non-ORACLE data source.
Parameters: none
Restrictions: Valid only in an On-Check-Unique trigger.

3.2.15.2 CLEAR_RECORD

Syntax: CLEAR_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: yes
Description: Causes Oracle Forms to remove, or flush, the current record from the block, without performing validation. If a query is open in the block, Oracle Forms fetches the next record to refill the block, if the record space is no longer filled after removing the current record.
A database record that has been cleared is not processed as a delete by the next Post and Commit Transactions process.
In a default master-detail block relation, clearing the master record causes all corresponding detail records to be cleared without validation.

3.2.15.3 CREATE_QUERIED_RECORD

Syntax: CREATE_QUERIED_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: When called from an On-Fetch trigger, creates a record on the block's waiting list. The waiting list is an intermediary record buffer that contains records that have been fetched from the data source but have not yet been placed on the block's list of active records. This built-in is included primarily for applications using transactional triggers to run against a non-ORACLE data source.
Note that there is no way to remove a record from the waiting list. Consequently, the application must ensure that there is data available to be used for populating the record programmatically.
Parameters: none
Restrictions:

- Valid only in the On-Fetch trigger.
- In blocks with a large number of records, this procedure can have side effects on disk I/O, memory allocation, or both.

3.2.15.4 CREATE_RECORD

Syntax: CREATE_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Creates a new record in the current block after the current record. Oracle Forms then navigates to the new record.
Parameters: none

3.2.15.5 DELETE_RECORD

Syntax: DELETE_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: When used outside an On-Delete trigger, removes the current record from the block and marks the record as a delete. Records removed with this built-in are not removed one at a time, but are added to a list of records that are deleted during the next available commit process.
If the record corresponds to a row in the database, Oracle Forms locks the record before removing it and marking it as a delete.
If a query is open in the block, Oracle Forms fetches a record to refill the block if necessary. See also the description for the CLEAR_RECORD built-in subprogram.

In an On-Delete trigger, DELETE_RECORD initiates the default Oracle Forms processing for deleting a record during the Post and Commit Transaction process, as shown in Example 2 below.
Parameters: none

3.2.15.6 DOWN

Syntax: DOWN;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the instance of the current item in the record with the next higher sequence number. If necessary, Oracle Forms fetches a record. If Oracle Forms has to create a record, DOWN navigates to the first navigable item in the record.
Parameters: none

3.2.15.7 DUPLICATE_RECORD

Syntax: DUPLICATE_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Copies the value of each item in the record with the next lower sequence number to the corresponding items in the current record. The current record must not correspond to a row in the database. If it does, an error occurs. The duplicate record inherits the record status (NEW, CHANGED, or QUERY) of the source record.
Parameters: none
Restrictions: A previous record must exist in your current session.

3.2.15.8 FIRST_RECORD

Syntax: FIRST_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the first record in the block's list of records.
Parameters: none

3.2.15.9 GENERATE_SEQUENCE_NUMBER

Syntax: GENERATE_SEQUENCE_NUMBER;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Initiates the default Oracle Forms processing for generating a unique sequence number when a record is created. When a sequence object is defined in the database, you can reference it as a default value for an item by setting the Default property to SEQUENCE.my_seq.NEXTVAL. By default, Oracle Forms gets the next value from the sequence whenever a record is created. When you are connecting to a non-ORACLE data source, you can include a call to this built-in in the On-Sequence-Number trigger. Refer to Chapter 2, "Triggers" for more information.
Parameters: none
Restrictions: Valid only in an On-Sequence-Number trigger.

3.2.15.10 GET_RECORD_PROPERTY

Syntax: GET_RECORD_PROPERTY(record_number, block_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the value for the given property for the given record number in the given block. The three parameters are required. If you do not pass the proper constants, Oracle Forms issues an error. For example, you must pass a valid record number as the argument to the record_number parameter.
Parameters and usage notes: see the manual

3.2.15.11 GO_RECORD

Syntax: GO_RECORD(record_number);
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the record with the specified record number.
Parameters:

record_number Specifies any integer value that PL/SQL can evaluate to a number. This includes values derived from calls to system variables, such as TO_NUMBER(:SYSTEM.TRIGGER_RECORD) + 8.
You can use the system variables SYSTEM.CURSOR_RECORD or SYSTEM.TRIGGER_RECORD to determine a record's sequence number.
Restrictions:

- The specified record number must evaluate to a positive integer.
- If the query is open and the specified record number is greater than the number of records already fetched, Oracle Forms fetches additional records to satisfy the call to this built-in.

3.2.15.12 INSERT_RECORD

Syntax: INSERT_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: When called from an On-Insert trigger, inserts the current record into the database during Post and Commit Transactions processing. This built-in is included primarily for applications that will run against a non-ORACLE datasource.
Parameters: none
Restrictions: Valid only in an On-Insert trigger.

3.2.15.13 LAST_RECORD

Syntax: LAST_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the last record in the block's list of records. If a query is open in the block, Oracle Forms fetches the remaining selected records into the block's list of records, and closes the query.
Parameters: none

3.2.15.14 LOCK_RECORD

Syntax: LOCK_RECORD;
Built-in Type: unrestricted procedure
Enter Query Mode: no
Description: Attempts to lock the row in the database that corresponds to the current record. LOCK_RECORD locks the record immediately, regardless of whether the Locking Mode block property is set to Immediate (the default) or Delayed.
When executed from within an On-Lock trigger, LOCK_RECORD initiates default database locking. The following example illustrates this technique (see the manual).
Parameters: none

3.2.15.15 NEXT_RECORD

Syntax: NEXT_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the first enabled and navigable item in the record with the next higher sequence number than the current record. If there is no such record, Oracle Forms will fetch or create a record. If the current record is a new record, NEXT_RECORD fails.
Parameters: none
Restrictions: Not allowed in Enter Query mode.

3.2.15.16 NEXT_SET

Syntax: NEXT_SET;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Fetches another set of records from the database and navigates to the first record that the fetch retrieves. NEXT_SET succeeds only if a query is open in the current block.
Parameters: none

3.2.15.17 PREVIOUS_RECORD

Syntax: PREVIOUS_RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no

Description: Navigates to the first enabled and navigable item in the record with the next lower sequence number than the current record.
Parameters: none

3.2.15.18 SCROLL_DOWN

Syntax: SCROLL_DOWN;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Scrolls the current block's list of records so that previously hidden records with higher sequence numbers are displayed. If there are available records and a query is open in the block, Oracle Forms fetches records during SCROLL_DOWN processing. In a single-line block, SCROLL_DOWN displays the next record in the block's list of records. SCROLL_DOWN puts the input focus in the instance of the current item in the displayed record with the lowest sequence number.
Parameters: none

3.2.15.19 SCROLL_UP

Syntax: SCROLL_UP;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Scrolls the current block's list of records so that previously hidden records with lower sequence numbers are displayed. This action displays records that were "above" the block's display.
SCROLL_UP puts the input focus in the instance of the current item in the displayed record that has the highest sequence number.
Parameters: none

3.2.15.20 SELECT_RECORDS

Syntax: SELECT_RECORDS;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: When called from an On-Select trigger, initiates default Oracle Forms SELECT processing. This built-in is included primarily for applications that run against a non-ORACLE data source, and use transactional triggers to replace default Oracle Forms transaction processing.
Parameters: none
Restrictions: Valid only within an On-Select trigger.

3.2.15.21 SET_RECORD_PROPERTY

Syntax: SET_RECORD_PROPERTY(record_number, block_name, property, value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the specified record property to the specified value.
Parameters and restrictions: see the manual

3.2.15.22 UP

Syntax: UP;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: Navigates to the instance of the current item in the record with the next lowest sequence number.
Parameters: none

3.2.15.23 UPDATE_RECORD

Syntax: UPDATE RECORD;
Built-in Type: restricted procedure
Enter Query Mode: no
Description: When called from an On-Update trigger, initiates the default Oracle Forms processing for updating a record in the database during the Post and Commit Transaction process.
This built-in is included primarily for applications that run against a non-ORACLE data source.
Parameters: none
Restrictions: Valid only in an On-Update trigger.

3.2.16 Record Group Built-ins

3.2.16.1 ADD_GROUP_COLUMN

Syntax: ADD_GROUP_COLUMN(recordgroup_id, groupcolumn_name, column_type);
ADD_GROUP_COLUMN(recordgroup_name, groupcolumn_name, column_type);
ADD_GROUP_COLUMN(recordgroup_id, groupcolumn_name, column_type, column_width);
ADD_GROUP_COLUMN(recordgroup_name, groupcolumn_name, column_type, column_width);
Built-in Type: unrestricted function
Enter Query Mode: yes
Returns: GroupColumn
Description: Adds a column of the specified type to the given record group.
Parameters and restrictions: see the manual or on-line help

3.2.16.2 ADD_GROUP_ROW

Syntax: ADD_GROUP_ROW(recordgroup_id, row_number);
ADD_GROUP_ROW(recordgroup_name, row_number);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Adds a row to the given record group.
Parameters and restrictions: see the manual or on-line help

3.2.16.3 CREATE_GROUP

Syntax: CREATE_GROUP(recordgroup_name);
Built-in Type: unrestricted function
Returns: RecordGroup
Enter Query Mode: yes
Description: Creates a non-query record group with the given name. The new record group has no columns and no rows until you explicitly add them using the ADD_GROUP_COLUMN, the ADD_GROUP_ROW, and the POPULATE_GROUP_WITH_QUERY built-ins.
Parameters: see the manual or on-line help
Restrictions: A record group created with CREATE_GROUP does not have an associated query. For this reason, you cannot populate such a record group with the POPULATE_GROUP built-in. Use POPULATE_GROUP_WITH_QUERY instead.

3.2.16.4 CREATE_GROUP_FROM_QUERY

Syntax: CREATE_GROUP_FROM_QUERY(recordgroup_name, query);
Built-in Type: unrestricted function
Returns: RecordGroup
Enter Query Mode: yes
Description: Creates a record group with the given name. The record group has columns representing each column you include in the select list of the query. Add rows to the record group with the POPULATE_GROUP built-in.
Note: If you do not pass a formal column name or alias for a column in the SELECT statement, Oracle Forms creates ICRGGQ with a dummy counter <NUM>. This happens whenever the column name would have been invalid. The first dummy name-counter always takes the number one. For example, the query SELECT 1 + 1 FROM DUAL would result in a column named ICRGGQ_1.
Parameters: see the manual or on-line help
Restrictions: You can use the POPULATE_GROUP_WITH_QUERY built-in to populate groups created with the built-in CREATE_GROUP_FROM_QUERY, but the columns you specify in the SELECT statement must match those in the record group, as specified in the CREATE_GROUP_FROM_QUERY.

3.2.16.5 DELETE_GROUP

Syntax: DELETE_GROUP(recordgroup_id);
DELETE_GROUP(recordgroup_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Deletes a programmatically created record group.
Parameters: see the manual or on-line help
Restrictions: This built-in cannot be used to delete a record group that was created at design time.

3.2.16.6 DELETE_GROUP_ROW

Syntax: DELETE_GROUP_ROW(recordgroup_id, row_number);
DELETE_GROUP_ROW(recordgroup_name, row_number);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Deletes the indicated row or all rows of the given record group. Oracle Forms automatically decrements the row numbers of all rows that follow a deleted row. When rows are deleted, the appropriate memory is freed and available to Oracle Forms.
If you choose to delete all rows of the group by supplying the ALL_ROWS constant, Oracle Forms deletes the rows, but the group still exists until you perform the DELETE_GROUP subprogram.
When a single row is deleted, subsequent rows are renumbered so that row numbers remain contiguous.
Parameters: see the manual or on-line help
Restrictions: This built-in cannot be used to delete rows from a static record group.

3.2.16.7 FIND_COLUMN

Syntax: FIND_COLUMN(recordgroup.groupcolumn_name);
Built-in Type: unrestricted function
Returns: GroupColumn
Enter Query Mode: yes
Description: Searches the list of record group columns and returns a groupcolumn ID when it finds a valid column with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of GroupColumn.
Parameters: see the manual or on-line help

3.2.16.8 FIND_GROUP

Syntax: FIND_GROUP(recordgroup_name);
Built-in Type: unrestricted function
Returns: RecordGroup
Enter Query Mode: yes
Description: Searches the list of record groups and returns a record group ID when it finds a valid group with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of RecordGroup.
Parameters: see the manual or on-line help
Restrictions: Performance of this function depends upon the number of record groups.

3.2.16.9 GET_GROUP_CHAR_CELL

Syntax: GET_GROUP_CHAR_CELL(groupcolumn_id, row_number);
GET_GROUP_CHAR_CELL(groupcolumn_name, row_number);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the CHAR or LONG value for a record group cell identified by the given row and column. A cell is an intersection of a row and column.
Parameters: see the manual or on-line help
Restrictions: The row_number specified must be within the bounds implied by the number of rows in the record group. A non-existent row_number results in an index out of bounds error.

3.2.16.10 GET_GROUP_DATE_CELL

Syntax: GET_GROUP_DATE_CELL(groupcolumn_id, row_number);
GET_GROUP_DATE_CELL(groupcolumn_name, row_number);
Built-in Type: unrestricted function
Returns: DATE
Enter Query Mode: yes
Description: Returns the DATE value for a record group cell identified by the given row and column. A cell is an intersection of a row and column.
Parameters: see the manual or on-line help
Restrictions: The row_number specified must be within the bounds implied by the number of rows in the record group. A non-existent row_number results in an index out of bounds error.

3.2.16.11 GET_GROUP_NUMBER_CELL

Syntax: GET_GROUP_NUMBER_CELL(groupcolumn_id, row_number);
GET_GROUP_NUMBER_CELL(groupcolumn_name, row_number);
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Returns the NUMBER value for a record group cell identified by the given row and column. A cell is an intersection of a row and column.
Parameters: see the manual or on-line group
Restrictions: The row_number specified must be within the bounds implied by the number of rows in the record group. A non-existent row_number results in an index out of bounds error.

3.2.16.12 GET_GROUP_RECORD_NUMBER

Syntax: GET_GROUP_RECORD_NUMBER(groupcolumn_id, cell_value);
GET_GROUP_RECORD_NUMBER(groupcolumn_name, cell_value);
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Returns the record number of the first record in the record group with a column value equal to the cell_value parameter. If there is no match, 0 (zero) is returned.
Parameters: see the manual or on-line help
Restrictions: The dataype of the cell_value parameter must match the datatype of the record group column. The comparison is case-sensitive for CHAR comparisons.

3.2.16.13 GET_GROUP_ROW_COUNT

Syntax: GET_GROUP_ROW_COUNT(recordgroup_id);
GET_GROUP_ROW_COUNT(recordgroup_name);
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Returns the number of rows in the record group.
Parameters: see the manual or on-line help

3.2.16.14 GET_GROUP_SELECTION

Syntax:
GET_GROUP_SELECTION(recordgroup_id, selection_number);
GET_GROUP_SELECTION(recordgroup_name, selection_number);
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Retrieves the sequence number of the selected row for the given group.
Parameters: see the manual or on-line help

3.2.16.15 GET_GROUP_SELECTION_COUNT

Syntax: GET_GROUP_SELECTION_COUNT(recordgroup_id);
GET_GROUP_SELECTION_COUNT(recordgroup_name);
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Returns the number of rows in the indicated record group that have been programmatically marked as selected by a call to SET_GROUP_SELECTION.
Parameters: see the manual or on-line help

3.2.16.16 ID_NULL

See Alert built-ins

3.2.16.17 POPULATE_GROUP

Syntax: POPULATE_GROUP(recordgroup_id);
POPULATE_GROUP(recordgroup_name);
Built-in Type: unrestricted function

Returns: NUMBER
Enter Query Mode: yes
Description: Executes the query associated with the given record group and returns a number indicating success or failure of the query. Upon a successful query, POPULATE_GROUP returns a 0 (zero). An unsuccessful query generates an ORACLE error number that corresponds to the particular SELECT statement failure. The rows that are retrieved as a result of a successful query replace any rows that exist in the group.
Note: Be aware that the POPULATE_GROUP array fetches 100 records at a time. To improve network performance, you may want to restrict queries, thus limiting network traffic.
Parameters and restrictions: see the manual or on-line help

3.2.16.18 POPULATE_GROUP_WITH_QUERY

Syntax: POPULATE_GROUP_WITH_QUERY(recordgroup_id, query);
POPULATE_GROUP_WITH_QUERY(recordgroup_name, query);
Built-in Type: unrestricted function
Returns: NUMBER
Enter Query Mode: yes
Description: Populates a record group with the given query. The record group is cleared and rows that are fetched replace any existing rows in the record group.
If the SELECT statement fails, Oracle Forms returns an ORACLE error number. If the query is successful, this built-in returns 0 (zero).
You can use this built-in to populate record groups that were created by a call to either:

- the CREATE_GROUP built-in or
- the CREATE_GROUP_FROM_QUERY built-in

When you use this built-in, the indicated query becomes the default query for the group, and will be executed whenever the POPULATE_GROUP built-in is subsequently called.
Note: Be aware that the POPULATE_GROUP_WITH_QUERY array fetches 100 records at a time. To improve network performance, you may want to restrict queries, thus limiting network traffic.
Parameters and restrictions: see the manual or on-line help

3.2.16.19 RESET_GROUP_SELECTION

Syntax: RESET_GROUP_SELECTION(recordgroup_id);
RESET_GROUP_SELECTION(recordgroup_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Deselects any selected rows in the given group. Use this built-in to deselect all record group rows that have been programmatically marked as selected by executing SET_GROUP_SELECTION on individual rows.
Parameters: see the manual or on-line help

3.2.16.20 SET_GROUP_CHAR_CELL

Syntax: SET_GROUP_CHAR_CELL(groupcolumn_id, row_number, cell_value);
SET_GROUP_CHAR_CELL(groupcolumn_name, row_number, cell_value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the value for the record group cell identified by the given row and column.
Parameters and restrictions: see the manual or on-line help

3.2.16.21 SET_GROUP_DATE_CELL

Syntax: SET_GROUP_DATE_CELL(groupcolumn_id, row_number, cell_value);
SET_GROUP_DATE_CELL(groupcolumn_name, row_number, cell_value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the value for the record group cell identified by the given row and column.
Parameters and restrictions: see the manual or on-line help

3.2.16.22 SET_GROUP_NUMBER_CELL

Syntax: SET_GROUP_NUMBER_CELL(groupcolumn_id, row_number, cell_value);
SET_GROUP_NUMBER_CELL(groupcolumn_name, row_number, cell_value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes

Description: Sets the value for the record group cell identified by the given row and column.
Parameters and restrictions: see the manual or on-line help

3.2.16.23 SET_GROUP_SELECTION

Syntax: SET_GROUP_SELECTION(recordgroup_id, row_number);
SET_GROUP_SELECTION(recordgroup_name, row_number);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Marks the specified row in the given record group for subsequent programmatic row operations. Rows are numbered sequentially starting at 1. If you select rows 3, 8, and 12, for example, those rows are considered by Oracle Forms to be selections 1, 2, and 3. You can undo any row selections for the entire group by calling the RESET_GROUP_SELECTION built-in.
Parameters: see the manual or on-line help

3.2.16.24 UNSET_GROUP_SELECTION

Syntax: UNSET_GROUP_SELECTION(recordgroup_id, row_number);
UNSET_GROUP_SELECTION(recordgroup_name, row_number);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Unmarks the specified row in the indicated record group. Use the procedure to unmark rows that have been programmatically selected by a previous call to SET_GROUP_SELECTION.
Rows are numbered sequentially starting at 1. If you select rows 3, 8, and 12, for example, those rows are considered by Oracle Forms to be selections 1, 2, and 3. You can undo any row selections for the entire group by calling the RESET_GROUP_SELECTION built-in.
Parameters: see the manual or on-line help

3.2.17 Relation Built-ins

3.2.17.1 FIND_RELATION

Syntax: FIND_RELATION(relation_name);
Built-in Type: unrestricted function
Returns: Relation
Enter Query Mode: yes
Description: Searches the list of relations and returns a relation ID when it finds a valid relation with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of Relation.
Parameters: relation_name Specifies a valid CHAR relation name.

3.2.17.2 GET_RELATION_PROPERTY

Syntax: GET_RELATION_PROPERTY(relation_id, property);
GET_RELATION_PROPERTY(relation_name, property);
Built-in Type: unrestricted function
Returns: CHAR
Enter Query Mode: yes
Description: Returns the state of the given relation property.
Parameters: see the manual or on-line help

3.2.17.3 ID_NULL

See Alert built-ins.

3.2.17.4 SET_RELATION_PROPERTY

Syntax: SET_RELATION_PROPERTY(relation_id, property, value);
SET_RELATION_PROPERTY(relation_name, property, value);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Sets the given relation property in a master-detail relationship.
Parameters: see the manual or on-line help
Restrictions: You can only set one property per call to this built-in.

3.2.18 Transactional Built-ins

3.2.18.1 CHECK_RECORD_UNIQUENESS

See Record built-ins.

3.2.18.2 DELETE_RECORD

See Record built-ins.

3.2.18.3 ENFORCE_COLUMN_SECURITY

Syntax: ENFORCE_COLUMN_SECURITY
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Executes default processing for checking column security on a database column. This built-in is included primarily for applications that run against a non-ORACLE data source, and use transactional triggers to replace default Oracle Forms transaction processing.
Default Check Column Security processing refers to the sequence of events that occurs when Oracle Forms enforces column-level security for each block that has the Column Security block property set True. To enforce column security, Oracle Forms queries the database to determine the base table columns to which the current form operator has update privileges. For columns to which the operator does not have update privileges, Oracle Forms makes the corresponding base table items in the form non-updateable by setting the Update Allowed item property to False dynamically. By default, Oracle Forms performs this operation at form startup, processing each block in sequence.
For more information, refer to Oracle Forms Advanced Techniques, Chapter 4, "Connecting to Non-Oracle Data Sources."
Restrictions: Valid only in an On-Column-Security trigger.
Usage Notes: You can include this built-in subprogram in the On-Column-Security trigger if you intend to augment the behavior of that trigger rather than completely replace the behavior. For more information, refer to Chapter 2, "Triggers," in the manual.

3.2.18.4 FETCH_RECORDS

Syntax: FETCH_RECORDS;
Built-in Type: unrestricted procedure
Enter Query Mode: no
Description: When called from an On-Fetch trigger, initiates the default Oracle Forms processing for fetching records that have been identified by SELECT processing.
This built-in is included primarily for applications that will run against a non-ORACLE data source.
Parameters: none

3.2.18.5 FORMS_DDL

Syntax: FORMS_DDL(statement);
Built-in Type: unrestricted function
Enter Query Mode: yes
Description: Issues dynamic SQL statements at runtime, including server-side PL/SQL and DDL.
Note: All DDL operations issue an implicit COMMIT and will end the current transaction without allowing Oracle Forms to process any pending changes.
Parameters: see the manual or on-line help

3.2.18.6 GENERATE_SEQUENCE_NUMBER

See Record built-ins.

3.2.18.7 INSERT_RECORD

See Record built-ins.

3.2.18.8 ISSUE_ROLLBACK

Syntax: ISSUE_ROLLBACK(savepoint_name);
Built-in Type: unrestricted procedure
Enter Query Mode: no
Description: When called from an On-Rollback trigger, initiates the default Oracle Forms processing for rolling back to the indicated savepoint.
This built-in is included primarily for applications that will run against a non-ORACLE data source.

Parameters and restrictions: see the manual or on-line help

3.2.18.9 ISSUE_SAVEPOINT

Syntax: ISSUE_SAVEPOINT(savepoint_name);
Built-in Type: unrestricted procedure
Enter Query Mode: no
Description: When called from an On-Savepoint trigger, ISSUE_SAVEPOINT initiates the default processing for issuing a savepoint. You can use GET_APPLICATION_PROPERTY (SAVEPOINT_NAME) to determine the name of the savepoint that Oracle Forms would be issuing by default, if no On-Savepoint trigger were present.
This built-in is included primarily for applications that will run against a non-ORACLE datasource.
Parameters and restrictions: see the manual or on-line help

3.2.18.10 LOGON

Syntax: LOGON(username, password);
LOGON(username, password, logon_screen_on_error);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Performs the default Oracle Forms logon processing with an indicated username and password. Call this procedure from an On-Logon trigger when you want to augment default logon processing.
Parameters and restrictions: see the manual or on-line help

3.2.18.11 LOGON_SCREEN

Syntax: LOGON_SCREEN;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Displays the default Oracle Forms logon screen and requests a valid username and password. Most commonly, you will include this built-in subprogram in an On-Logon trigger to connect to a non-ORACLE data source.
Parameters: none
Restrictions:

- When first entering a form, if the On-Logon trigger fails with an unhandled exception, no other triggers are executed, and the current Runform session is aborted. Otherwise, to change connections in the middle of a session, you can log off, then log on again.
- You must issue a call to the LOGON built-in to create the connection to your data source.

3.2.18.12 LOGOUT

Syntax: LOGOUT;
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Disconnects the application from the ORACLE RDBMS. All open cursors are automatically closed when you issue a call to the LOGOUT built-in. You can programmatically log back on with LOGON. If you LOGOUT of a multiple-form application with multiple connections, Oracle Forms tries to re-establish all of those connections when you subsequently execute LOGON.
Parameters: none

3.2.18.13 SELECT_RECORDS

See Record built-ins.

3.2.18.14 UPDATE_RECORD

See Record built-ins.

3.2.19 VBX Control Built-ins

See the manual or on-line help

3.2.20 Window Built-ins

3.2.20.1 FIND_WINDOW

Syntax: FIND_WINDOW(window_name);

Built-in Type: unrestricted function
Returns: Window
Enter Query Mode: yes
Description: Searches the list of windows and returns a window ID when it finds a valid window with the given name. You must define an appropriately typed variable to accept the return value. Define the variable with a type of Window.
Parameters: window_name Specifies the valid CHAR window name.
Syntax: HIDE_WINDOW(window_id);
HIDE_WINDOW(window_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Hides the given window. HIDE_WINDOW is equivalent to setting VISIBLE to False by calling SET_WINDOW_PROPERTY.
Parameters:
window_id Specifies the unique ID that Oracle Forms assigns the window at the time it creates it. Use the FIND_WINDOW built-in to return the ID to an appropriately typed variable. The data type of the ID is Window.
window_name Specifies the name that you gave the window when creating it.

3.2.20.2 ID_NULL

See Alert built-ins.

3.2.20.3 MOVE_WINDOW

Syntax: MOVE_WINDOW(window_id, x, y);
MOVE_WINDOW(window_name, x, y);
Built-in Type: unrestricted function
Enter Query Mode: yes
Description: Moves the given window to the location specified by the given coordinates.
If you have specified the form property Coordinate System as Character, then your x, y coordinates are specified in characters. If the Coordinate System is specified as Real, then your x, y coordinates are specified in the real units you have selected--pixels, inches, centimeters, or points.
Parameters: see the manual or online help

3.2.20.4 REPLACE_CONTENT_VIEW

Syntax: REPLACE_CONTENT_VIEW(window_id, view_id);
REPLACE_CONTENT_VIEW(window_name, view_id);
REPLACE_CONTENT_VIEW(window_id, view_name);
REPLACE_CONTENT_VIEW(window_name, view_name);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Replaces the content canvas-view currently displayed in the indicated window with a different content canvas-view.
Parameters and restrictions: see the manual or online help

3.2.20.5 RESIZE_WINDOW

Syntax: RESIZE_WINDOW(window_id, width, height);
RESIZE_WINDOW(window_name, width, height);
Built-in Type: unrestricted procedure
Enter Query Mode: yes
Description: Changes the size of the given window to the given width and height. A call to RESIZE_WINDOW sets the width and height of the window, even if the window is not currently displayed. RESIZE_WINDOW does not change the position of the window, as specified by the x and y coordinates of the window's upper left corner on the screen.
On Microsoft Windows, you can resize the MDI application window by specifying the constant FORMS_MDI_WINDOW as the window name.
You can also resize a window with SET_WINDOW_PROPERTY.
Parameters: see the manual or online help

3.2.20.6 SHOW_WINDOW

Syntax: SHOW_WINDOW(window_id);

SHOW_WINDOW(window_id, x, y);
SHOW_WINDOW(window_name);
SHOW_WINDOW(window_name, x, y);

Built-in Type: unrestricted procedure

Enter Query Mode: yes

Description: Displays the indicated window at either the optionally included X,Y coordinates, or at the window's current X,Y coordinates. If the indicated window is a modal window, SHOW_WINDOW is executed as a GO_ITEM call to the first navigable item in the modal window.

Parameters:

window_id Specifies the unique ID that Oracle Forms assigns the window when created. Use the FIND_WINDOW built-in to return the ID to an appropriately typed variable. The data type of the ID is Window.

window_name Specifies the name that you gave the window when defining it. The data type of the name is CHAR

x Specifies the x coordinate of the window. Supply a whole number for this argument.

y Specifies the y coordinate of the window. Specify this value as a whole NUMBER.