Oracle® Forms Developer

Form Builder Reference, Volume 2

Release 6*i*

January, 2000

Part No: A73074-01

ORACLE

Oracle Forms Developer: Form Builder Reference, Release 6*i*

Volume 2

Part No: A73074-01

Contributors: Fred Bethke, Joan Carter, Ken Chu, Kate Dumont, Tom Haunert, Colleen McCann, Leanne Soylemez, Poh Lee Tan, Tony Wolfram

Table of Contents

Send Us Your Comments

Oracle Forms Developer: Form Builder Reference, Release 6*i*

Volume 2

Part No: A73074-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the part number, chapter, section, and page number (if available). You can send comments to us by electronic mail to oddoc@us.oracle.com.

If you have any problems with the software, please contact your local Oracle World Wide Support Center.

Preface

This book is Volume 2 of the *Oracle Forms Develop:r Form Builder Reference*. For more information about the book, please see the preface in Volume 1.

Properties (continued)

Format Mask property

Description

Specifies the display format and input accepted for data in text items.

Applies to text item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Required/Optional optional

Usage Notes

Valid format masks for character strings, numbers and dates are described in the following tables.

Character Strings

The following table describes valid format masks for character strings.

Element	*Example*	*Description*
FM	FMXX99	Fill mode: accept string as typed, do not right justify. Allows end user input string to be shorter than the format mask.
X	XXXX	Any alphabetic, numeric, or special character. End user input string must be exact length specified by format mask.
9	9999	Numeric characters only. End user input string must be exact length specified by format mask.
A	AAAA	Alphabetic characters only. End user input string must be exact length specified by format mask.

Character String Examples

Format Mask	*Description*

XXAA	Will accept: --ab, abcd, 11ab; will *not* accept: --11, ab11, or ab-- (must use XX to accept hyphens and other special characters).
XXXX	Will accept any combination of alphabetic, numeric, or special characters: --ab, abcd, 11ab, --11, ab11, or ab--. Will accept 1234 or abcd; will *not* accept 123 or abc. (To accept input string shorter than mask, use FMXXXX.)
FMXX99	Will accept ab12, ab1, ab followed by two spaces; will *not* accept 12ab or abcd. (To produce the Form Builder Version 3.0 Alpha datatype, use FMAAAAAA.)

- To embed additional characters such as a hyphen (-) or a comma (,), surround the character with double-quotes (").
- Embedded characters are separate from text item values and are not collated along with text item values, even when the end user enters them.

NUMBERS

The following table describes valid format masks for numbers.

Element	***Example***	***Description***
9	9999	Number of nines determines display width. Any leading zeros will be displayed as blanks.
0	0999	Display leading zeros.
0	9990	Display zero value as zero, not blank.
$	$9999	Prefix value with dollar sign.
B	B9999	Display zero value as blank, not "0".
MI	9999MI	Display "-" after a negative value.
PR	9999PR	Display a negative value in <angle brackets>.
comma	9,999	Display a comma in this position. For correct behavior in multilingual applications, substitute G to return the appropriate group (thousands) separator.
period	99.99	Display a decimal point in this position. For correct behavior in multilingual applications, substitute D to return the appropriate decimal separator.
E	9.999EEEE	Display in scientific notation (format must contain exactly four "E"s).
FM	FM999	Fill mode: accept string as typed, do not right justify.

- When you mask a number with nines (9), Form Builder adds a space in front of the number to

accommodate the plus (+) or minus (-) sign. However, since the plus sign is not displayed, it appears as if Form Builder adds a space in front of the number. (The minus sign is displayed.)

- To embed additional characters such as a hyphen (-) or a comma (,), surround the character with double-quotes (").
- Embedded characters are separate from text item values and are not collated along with text item values, even when the end user enters them.

NUMBER Examples

Format Mask	*Description*
FM099"-"99"-"9999	Displays the social security number as formatted, including hyphens, even if end user enters only nine digits.To create a Social Security column, create an 11-character column, set to fixed length, with a format mask of 099"-"99"-"9999. This mask will accommodate Social Security numbers that begin with zero, accepting 012-34-5678 or 012345678 (both stored as 012345678).
99999PR	Accepts -123; reformats as <123>.
999MI	Accepts -678; reformats as 678-.
9.999EEEE	Displays as 1.00E+20.

How Forms handles length mismatches

If a runtime user enters a numeric string that exceeds the format mask specification, the value will be rejected. For example:

Format Mask	*User enters*	*Result*
99.9	321.0	Invalid
99.9	21.01	Invalid
99.9	21.1	21.1
99.9	01.1	1.1

In contrast, if a numeric value fetched from the database exceeds the format mask specification for its display field, the value is displayed, but truncated, with rounding, to fit the mask. (The item itself within the Forms application retains its full value.) For example, if the database held the value 2.0666, and the format mask was 99.9, the value displayed to the user would be 2.1. However, the value of the item within the form would be the full 2.0666.

Dates

The following table describes valid format masks for dates.

Element	*Description*

YYYY or SYYYY	4-digit year; "S" prefixes "BC" date with "-".
YYY or YY or Y	Last 3, 2, or 1 digits of year.
Y,YYY	Year with comma in this position.
BC or AD	BC/AD indicator.
B.C. or A.D.	BD/AD indicator with periods.
RR	Defaults to correct century. Deduces the century from a date entered by comparing the 2 digit year entered with the year and century to which the computer's internal clock is set. Years 00-49 will be given the 21st century (the year 2000), and years from 50-99 will be given the 20th century (the year 1900).
MM	Month (01-12; JAN = 01).
MONTH	Name of month, padded with blanks to length of 9 characters.
MON	Name of month, 3-letter abbreviation.
DDD	Day of year (1-366).
DD	Day of month (1-31).
D	Day of week (1-7; Sunday=1).
DAY	Name of day, padded with blanks to length of 9 characters.
DY	Name of day, 3-letter abbreviation.
J	Julian day; the number of days since January 1, 4712 BC.
AM or PM	Meridian indicator.
A.M. or P.M.	Meridian indicator with periods.
HH or HH12	Hour of day (1-12).
HH24	Hour of day (0-23).
MI	Minute (0-59).
SS	Second (0-59).
SSSSS	Seconds past midnight (0-86399).
/. , .	Punctuation is reproduced in the result.
"..."	Quoted string is reproduced in the result.
FM	Fill mode: assumes implied characters such as O or space; displays significant characters left justified. Allows end user input to be shorter than the format mask. (Use in conjunction with FX to require specific delimiters.)
FX	All date literals must match the format mask exactly,

including delimiters.

- When you prefix a date mask with FX, the end user must enter the date exactly as you define the mask, including the specified delimiters:

Date Examples

Format Mask	*Description*
FXDD-MON-YY	Will accept 12-JAN-94, but will *not* accept 12.JAN.94 or 12/JAN/94 because the delimiters do not match the mask. Will not accept 12JAN94 because there are no delimiters. Will accept 01-JAN-94 but will not accept 1-JAN-94.
FMDD-MON-YY	Will accept 01-JAN-94. Will also accept the entry of other delimiters, for example 01/JAN/94 and 01 JAN 94. However, will not accept 01JAN94. Will accept 1-JAN-94, converting it to 01-JAN-94.
DD-MON-YY	Will accept 12.JAN.94, 12/JAN/94 or 12-JAN-94. Note: Any delimiter characters will be accepted, but if delimiters are omitted by the end user, this mask will interpret date characters as a delimiters. Will accept 12-JAN94, (but will erroneously interpret as 12-JAN-04); but will *not* accept 12JAN94, because "AN" is not a valid month name.

- Use of a format mask only affects how the data looks. Form Builder stores full precision, regardless of how the data is presented.
- Embedded characters are separate from text item values and are not collated along with text item values, even when the end user enters them.
- To embed additional characters such as a hyphen (-) or a comma (,), surround the character with double-quotes ("). Note, however, that double-quotes themselves cannot be used as a character. In other words, trying to achieve output of DD"MM by specifying a mask of DD""""MM would not work.

Format Mask	*Description*
FMMONTH" "DD", "YYYY	Displays the text item data in the specified date format: JANUARY 12, 1994, including the appropriate blank spaces and comma.
FMDD-MONTH-YYYY	Displays as 12-JANUARY-1994.
DY-DDD-YYYY	Displays as WED-012-1994. Note: for input validation including day of the week, a mask that allows specific determination of the day is required, such as this example or DY-DD-MM-YY.

- When you use day of the week formats, be sure that the data includes day of the week information. To avoid illogical masks, display also either the day of the year (1-366) or the month in some format.

Format Mask	*Description*
DD-MONTH-YYYY	Displays as 12-JANUARY-1994.
DY-DDD-YYYY	Displays as WED-012-1994.
DY-DD-MON-YY	Displays as WED-12-JAN-94. Be sure to include month. Avoid masks such as DY-DD-YY, which could generate an error.

NLS Format Masks

The following table describes valid National Language Support (NLS) format masks.

Element	*Example*	*Description*
C	C999	Returns the international currency symbol.
L	L9999	Returns the local currency symbol.
D	99D99	Returns the decimal separator.
G	9G999	Returns the group (thousands) separator.
comma	9,999	Displays a comma in this position.
period	9.999	Displays a decimal point in this position. Displays a decimal point in this position.

NLS Format Mask Examples

Format Mask	*Description*
L99G999D99	Displays the local currency symbol, group, and decimal separators: if NLS_LANG=American, this item displays as $1,600.00; if NLS_LANG=Norwegian, this item displays as Kr.1.600,00.
C99G999D99	Displays the appropriate international currency symbol: if NLS_LANG=American, this item displays as USD1,600.00; if NLS_LANG=French, this item displays as FRF1.600,00.

Format Mask restrictions

- When setting the Maximum Length property for a text item, include space for any embedded characters inserted by the format mask you specify.
- Format masks can contain a maximum of 30 characters.
- Form Builder supports only ORACLE format masks that are used for both input and output. Output-only format masks, such as WW, are not supported.

Form_Name property

Description

Specifies the name of the form.

Applies to form

Set not settable

Refer to Built-in

GET_FORM_PROPERTY

Usage Notes

Form_Name at the form level corresponds to Current_Form_Name at the application level. Current_Form_Name is gettable with GET_APPLICATION_PROPERTY.

Formula property

Description

Specifies a single PL/SQL expression that determines the value for a formula calculated item. The expression can reference built-in or user-written subprograms.

Applies to item

Set Form Builder

Refer to Built-in

RECALCULATE

Usage Notes

You *cannot* enter an entire PL/SQL statement as your formula; accordingly, do not terminate your calculation expression with a semicolon. Form Builder adds the actual assignment code to the formula internally do not code it yourself. For example, instead of coding an entire assignment statement, code just the expression

```
:emp.sal + :emp.comm
```

Form Builder will internally convert this into a complete statement, e.g.,

```
:emp.gross_comp := (:emp.sal + :emp_comm);
```

Required/Optional required if Calculation Mode property is set to Formula

Frame Alignment property

Description

Specifies how objects should be aligned within the width of the frame, either Start, End, Center, Fill, or Column. This property is valid when the Layout Style property is set to Form.

Applies to frame

Set Form Builder

Default

Fill

Required/Optional required

Frame Title property

Description

Specifies the frame's title.

Applies to frame

Set Form Builder

Default

blank

Required/Optional optional

Frame Title Alignment property

Description

Specifies the title alignment for a frame, either Start, End, or Center.

Note: Title alignment is relative to the Direction of the canvas on which the canvas appears.

Applies to frame

Set Form Builder

Default

Start

Required/Optional required

Frame Title Background Color property

Description

Specifies the color to apply to the frame title background.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font color (usually white).

Required/Optional required

Frame Title Font Name property

Description

Specifies the name of the font (typeface) to apply to the frame title.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font

Required/Optional required

Frame Title Font Size property

Description

Specifies the size of the font (typeface) to apply to the frame title.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font size

Required/Optional required

Frame Title Font Spacing property

Description

Specifies the spacing to apply to the frame title text.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font spacing

Required/Optional required

Frame Title Font Style property

Description

Specifies the typographic style (for example, *Italic*) to apply to the frame title text.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font style

Required/Optional required

Frame Title Font Weight property

Description

Specifies the typographic weight (for example, **Bold**) to apply to the frame title text.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font weight.

Required/Optional required

Frame Title Foreground Color property

Description

Specifies the color to apply to the frame title text.

Applies to frame

Set Form Builder

Default

Defaults to the standard operating system font color (usually black).

Required/Optional required

Frame Title Offset property

Description

Specifies the distance between the frame and its title.

Applies to frame

Set Form Builder

Default

2 char cells (or the equivalent depending on the form coordinate system)

Required/Optional required

Frame Title Reading Order property

Description

Specifies the reading order for frame titles, either Default, Left-to-Right, or Right-to-Left.

Applies to frame

Set Form Builder

Default

Default

Required/Optional required

Frame Title Spacing property

Description

Specifies the amount of space reserved on either side of the frame's title.

Applies to frame

Set Form Builder

Default

1 char cell (or the equivalent depending on the form coordinate system)

Required/Optional required

Frame Title Visual Attribute Group property

Description

Specifies how the frame title's individual attribute settings (Font Name, Background Color, Fill Pattern, etc.) are derived. The following settings are valid for this property:

Default	Specifies that the object should be displayed with default color, pattern, and font settings. When Visual Attribute Group is set to Default, the individual attribute settings reflect the current system defaults. The actual settings are determined by a combination of factors, including the type of object, the resource file in use, and the platform.
Named visual attribute	Specifies a named visual attribute that should be applied to the object. Named visual attributes are separate objects that you create in the Object Navigator and then apply to interface objects, much like styles in a word processing program. When Visual Attribute Group is set to a named visual attribute, the individual attribute settings reflect the attribute settings defined for the named visual attribute object. When the current form does not contain any named visual attributes, the poplist for this property will show Default.

Applies to frame title

Set Form Builder

Default

Default

Usage Notes

- Default and named visual attributes can include the following individual attributes, listed in the order they appear in the Property Palette:

  Font Name The font family, or typeface, that should be used for text in the object. The list of fonts available is system-dependent.

  Font Size The size of the font, specified in points.

  Font Style The style of the font.

  Font Spacing The width of the font, that is, the amount of space between characters (kerning).

  Font Weight The weight of the font.

Foreground Color The color of the object's foreground region. For items, the Foreground Color attribute defines the color of text displayed in the item.

Background Color The color of the object's background region.

Fill Pattern The pattern to be used for the object's fill region. Patterns are rendered in the two colors specified by Background Color and Foreground Color.

Character Mode Logical Attribute Specifies the name of a character mode logical attribute defined in an Oracle Terminal resource file that is to be used as the basis of device attributes for a character mode version of your application.

White on Black Specifies that the object is to appear on a monochrome bitmap display device as white text on a black background.

Not all attributes are valid for each object type. For example, setting font attributes for a window object has no effect. (The font used in a window's title bar is derived from the system.)

A new object in a new form has Default visual attributes. The default settings are defined internally. Override the default font for new items and boilerplate by setting the optional FORMS60_DEFAULTFONT environment variable. For example, On Microsoft Windows, set this variable in the ORACLE.INI file, as follows:
`FORMS60_DEFAULTFONT="COURIER.10".` The default font specified determines the font used for new boilerplate text generated by the New Block window, and for any items that have Visual Attribute Group set to Default.

When you create an item in the Layout Editor, its initial visual attribute settings are determined by the current Layout Editor settings for fonts, colors, and patterns, as indicated by the Font dialog and Color and Pattern palettes.

On Microsoft Windows, the colors of buttons, window title bars, and window borders are controlled by the Windows Control Panel color settings specified for these elements. You cannot override these colors in Form Builder.

When the Use 3D Controls form property is set to Yes on Microsoft Windows (the default), items are rendered with shading that provides a sculpted, three-dimensional look. A side effect of setting this property is that any canvases that have Visual Attribute Group set to Default derive their color setting from the Windows Control Panel (gray for most color schemes). You can override this setting by explicitly applying named visual attributes to the canvas.

An item that has Visual Attribute Group set to Default, or that has individual attribute settings left unspecified, inherits those settings from the canvas to which it is assigned. Similarly, a canvas that has Visual Attribute Group set to Default, or that has individual attribute settings left unspecified, inherits those settings from the window in which it is displayed. For example, if you set a window's Background Color to CYAN, and then leave Background Color unspecified for the canvas assigned to the window, at runtime, that canvas will inherit the CYAN background from its window. Visual attribute settings derived through window canvas or canvas item inheritance are apparent only at runtime, not at design time.

You can apply property classes to objects to specify visual attribute settings. A property class can contain either the Visual Attribute Group property, or one or more of the individual attribute properties. (If a property class contains both Visual Attribute Group and individual attributes, the Visual Attribute Group property takes precedence.)

If you apply both a named visual attribute and a property class that contains visual attribute settings to the same object, the named visual attribute settings take precedence, and the property class visual attribute settings are ignored.

Logical attribute definitions defined in the resource file take precedence over visual attributes specified in the Form Builder, local environment variable definitions, and default Form Builder attributes. To edit the resource file, use the Oracle Terminal utility.

Graphics Type property

Description

A read-only property that specifies the type of the graphic object. Possible values include: Arc, Chart, Group, Image, Line, Polygon, Rectangle, Rounded Rectangle, Symbol, and Text. (Same possible values as Graphics Builder property Object Type.)

Applies to graphics general

Set Form Builder

Default

the type

Required/Optional required

Group_Name property

Description

Specifies the name of the record group on which an LOV is based.

Applies to LOV

Set programmatically

Refer to Built-in

- GET_LOV_PROPERTY
- SET_LOV_PROPERTY

Default

Name of the underlying record group.

Usage Notes

Set Group_Name to replace the LOV's current record group with another record group at runtime. The column names and types in the new record group must match the column names and types in the record group you are replacing.

Help property

Description

On character mode platform specifies help text for the menu item. Help text is displayed in a window when the end user presses [Help] while the menu item is selected.

Applies to menu item

Set Form Builder

Required/Optional optional

Help restrictions

Applies to character mode applications only.

Hide on Exit property

Description

For a modeless window, determines whether Form Builder hides the window automatically when the end user navigates to an item in another window.

Applies to window

Set Form Builder, programmatically

Refer to Built-in

- GET_WINDOW_PROPERTY
- SET_WINDOW_PROPERTY

Default

No

Hide on Exit restrictions

- Cannot be set for a root window: a root window always remains visible when the end user navigates to an item in another window.

Highest Allowed Value/Lowest Allowed Value property

Description

Determines the maximum value or minimum value, inclusive, that Form Builder allows in the text item.

Applies to text item

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Required/Optional optional

Usage Notes

- The following values are valid for range settings:
  any valid constant
  form item (:block_name.item_name)
  global variable (:GLOBAL.my_global)
  form parameter (:PARAMETER.my_param)
- Form Builder evaluates the values in items by data type, as follows:

ALPHA alphabetical according to your system's collating sequence

CHAR alphabetical according to your system's collating sequence

DATE chronological

DATETIME chronological

INT numerical ascending

NUMBER numerical ascending

- For all items, you can enter dates in either:
  the default format for your NLS_LANG setting *or*
  the format you specified as a format mask

For compatibility with prior releases, a reference to a form item or to a sequence may be specified with a leading ampersand (&) instead of a leading colon (:).

To specify a raw value that begins with a leading ampersand (‘&’) or a leading colon (‘:’), specify two of them (that is, ‘&&’ or ‘::’). (This is a change in Forms behavior, beginning with Release 6.5.)

Hint (Item) property

Description

Specifies item-specific help text that can be displayed on the message line of the root window at runtime. Hint text is available when the input focus is in the item.

Applies to all items except chart items, display items, and custom items

Set Form Builder

Refer to Built-in

- GET_ITEM_PROPERTY (HINT_TEXT)

Default

"Enter value for: <item name>"	For an item that was created by using the Data Block Wizard
NULL	For all other items

Required/Optional optional

Usage Notes

Leave the Hint property NULL if you do not want the item to have hint text.

Hint (Menu Item) property

Description

For a character mode application, specifies hint text for a menu item. In pull-down and bar menu display styles, hint text is displayed on the message line when the input focus is in the menu item.

In full-screen display style, hint text, if specified, is displayed as the item descriptor, and the menu item name is ignored. (When no hint text is specified, Form Builder displays the item name as the descriptor.)

Applies to menu item

Set Form Builder

Required/Optional optional

Hint (Menu Item) restrictions

- Applies to character mode applications only.

Hint (Menu Substitution Parameter) property

Description

Specifies a description or instruction to appear on the message line when the end user enters a value for the menu substitution parameter.

Applies to menu substitution parameter

Set Form Builder

Required/Optional optional

Horizontal Justification property

Description

Specifies the horizontal justification of the text object as either Left, Right, Center, Start, or End.

Applies to graphic text

Set Form Builder

Default

Start

Required/Optional required

Horizontal Margin property

Description

Specifies the distance between the frame's borders (left and right) and the objects within the frame.

Applies to frame

Set Form Builder

Default

1 char cell (or the equivalent depending on the form coordinate system)

Required/Optional required

Horizontal Object Offset property

Description

Specifies the horizontal distance between the objects within a frame.

Applies to frame

Set Form Builder

Default

2 char cells (or the equivalent depending on the form coordinate system)

Required/Optional required

Horizontal Origin property

Description

Specifies the horizontal position of the text object relative to its origin point as either Left, Right, or Center.

Applies to graphic text

Set Form Builder

Default

Left

Required/Optional required

Horizontal Toolbar Canvas property

Description

Specifies the canvas that should be displayed as a horizontal toolbar on the window. The canvas specified must be a horizontal toolbar canvas (Canvas Type property set to Horizontal Toolbar) and must be assigned to the current window by setting the Window property.

Applies to window

Set Form Builder

Default

Null

Required/Optional required if you are creating a horizontal toolbar

Usage Notes

- In the Properties window, the poplist for this property shows only canvases that have the Canvas Type property set to Horizontal Toolbar.
- At runtime, Form Builder attempts to display the specified horizontal toolbar on the window. However, if more than one toolbar of the same type has been assigned to the same window (by setting the canvas Window property to point to the specified window), Form Builder may display a different toolbar in response to navigation events or programmatic control.
- On Microsoft Windows, the specified horizontal toolbar canvas will not be displayed on the window if you have specified that it should be displayed on the MDI application window by setting the Form Horizontal Toolbar Canvas form property.

Icon Filename property

Description

Specifies the name of the icon resource that you want to represent the iconic button, menu item, or window.

Applies to button, menu item, window

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_MENU_ITEM_PROPERTY
- SET_MENU_ITEM_PROPERTY
- GET_WINDOW_PROPERTY
- SET_WINDOW_PROPERTY

Default

NULL

Required/Optional optional

Usage Notes

```
When defining the Icon Filename property, do not include the
icon file extension (.ico, .xpm, etc.).  For example, enter
my_icon, not my_icon.ico.
```

The icon filename should not end in any of the following five letters: A, L, M, S, and X. (Neither upper-case nor lower-case are allowed.) These are reserved letters used internally for icon sizing. Unexpected icon placement results may occur if your icon filename ends in any of these letters.

Use the platform-specific environment variable to indicate the directory where icon resources are located. For example, the Microsoft Windows name for this variable is UI60_ICON. (For more information on this variable name, refer to the Form Builder documentation for your operating system.)

Icon Filename restrictions

- For a window, it is only valid when Minimize Allowed property set to Yes.
- Icon resources must exist in the runtime operating system, and are not incorporated in the form definition. For this reason, icon resource files are not portable across platforms.

Icon in Menu property

Description

Specifies whether an icon should be displayed in the menu beside the menu item. If Yes, the Icon Filename property specifies the icon that will be displayed.

Applies to menu item

Set Form Builder

Default

No

Required/Optional optional

Iconic property

Description

Specifies that a button is to be an iconic button.

Applies to button

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Default

No

Required/Optional optional

Usage Notes

When Iconic is Yes, the button's Icon Filename property specifies the icon resource that Form Builder should display for the button.

Iconic restrictions

A valid icon resource file name must be supplied.

Image Depth property

Description

Specifies the image depth setting Form Builder applies to an image being read from or written to a file in the filesystem. Valid values are:

- Original
- Monochrome
- Gray
- LUT (Lookup Table)
- RGB (Red, Green, Blue)

Applies to image item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- WRITE_IMAGE_FILE

Default

Original

Required/Optional required

Image Format property

Description

Specifies the format in which an image item will be stored in the database. Valid values are:

- BMP
- CALS
- GIF
- JFIF
- PICT
- RAS
- TIFF
- TPIC

Applies to image item

Set Form Builder

Refer to Built-in

- GET_ITEM_PROPERTY
- WRITE_IMAGE_FILE

Default

TIFF

Required/Optional required

Usage Notes

- The default Oracle image storage format no longer is valid.
- The value you set for this property will override the original format of an image when the record containing the image item is stored in the database. For example, if an image item's Image Format property is set to GIF, and a TIFF image is pasted into the image item at runtime, the pasted image will be stored in GIF format when the record is saved to the database.

Implementation Class property

Description

Identifies the class name of a container of a JavaBean, or of a custom implementation for a control item type when you want to supply an alternate to the standard Form Builder control.

Applies to

The following control item types:

Bean Area
Check Box
List Item
Push Button
Radio Group
Text Item

Set

Form Builder

Default

None.

Required/Optional

Always required for Bean Area. This property identifies the class name of the container of the JavaBean you are adding to the application. (If this property is not supplied, the form's end user will see an empty square.)

Also required for any other control item type listed above if the form is to use a customized, user-supplied implementation of that control. This identifies the class name of the alternate control you are supplying.

Set at Runtime

No.

Usage Notes

- The Implementation Class property is only available for those control item types listed above, not for all control item types.

Include REF Item property

Description

Creates a hidden item called REF for this block. This item is used internally to coordinate master-detail relationships built on a REF link. This item also can be used programmatically to access the object Id (OID) of a row in an object table.

Applies to

Blocks based on object tables; master-detail REF links, in particular.

Set Form Builder

Default

Default is No. However, when creating a relationship based on a REF pointer, Form Builder sets this property to Yes.

Required/Optional

Required for a master block in a master-detail relationship based on a REF pointer.

Usage Notes

This REF item is used to obtain the object-ids (OIDs) of the rows in an object table.

Each row in an object table is identified by a unique object id (OID). The OID is a unique identifier for that row. These OIDs form an implicit column in an object table.

In a REF pointer relationship between two tables, the REF column in the pointing table holds copies of the OID values (addresses) of the pointed-to table. This forms the link between the two tables.

The Data Block wizard sets this property to Yes and creates this REF item when you build a master-detail relationship based on a REF pointer. The item is named REF, and is in the master block. It is not visible to the end user. In addition, the wizard sets the Copy_Value_From_Item property in the detail block to access this new REF. This is how Form Builder coordinates the master-detail relationship at runtime.

Inherit Menu property

Description

Specifies whether the window should display the current form menu on window managers that support this feature.

Applies to window

Set Form Builder

Default

Yes

Required/Optional optional

Inherit Menu restrictions

- Not valid on Microsoft Windows.

Initial Keyboard State property

Description

Note: This property is specific to bidirectional National Language Support (NLS) applications.

Initial Keyboard State sets the keyboard to generate either Local or Roman characters when the item receives input focus, so the end user can begin to type immediately, without switching the keyboard state.

Value	*Description*
Default	Initial keyboard state is based on the value of the Reading Order property.
Local	Initial keyboard state is Local (Right To Left language).
Roman	Initial keyboard state is Roman (Left To Right language).

Applies to display item, text item

Set Form Builder

Usage Notes

- Most of the time, you will use this property only for text items.
- The end user can override the setting for Initial Keyboard State by pressing the keyboard state toggle key.

Initial Menu property

Description

Specifies the name of the individual menu in the menu module that Form Builder should use as the main, or top-level, menu for this invocation. End users cannot navigate above the menu specified as the starting menu.

By default, the starting menu is the menu named in the menu module property, Main Menu. The Initial Menu property allows you to override the Main Menu property.

Applies to form module

Set Form Builder

Default

blank (Form Builder uses the default main menu as the starting menu)

Required/Optional optional

Initial Menu restrictions

- The menu specified must exist in the menu module.

Initial Value (Item) property

Description

Specifies the default value that Form Builder should assign to the item whenever a record is created. The default value can be one of the following:

- raw value (216, 'TOKYO')
- form item (:block_name.item_name)
- global variable (:GLOBAL.my_global)
- form parameter (:PARAMETER.my_param)
- a sequence (:SEQUENCE.my_seq.NEXTVAL)

Applies to check boxes, display items, list items, radio groups, text items, and user areas

Set Form Builder

Default

Null

Required/Optional Optional for all items except radio groups, check boxes, and list items.

For a radio group, a valid Initial Value is required unless

a) the radio group specifies Mapping of Other Values or,

b) the value associated with one of the radio buttons in the group is NULL.

For a list item, a valid Initial Value is required unless

a) the list item specifies Mapping of Other Values or,

b) the value associated with one of the list elements is NULL.

For a check box, a valid Initial Value is required unless

a) the check box specifies Mapping of Other Values or,

b) the value associated with Checked or Unchecked is NULL.

Usage Notes

- When using the default value to initialize the state of items such as check boxes, radio groups, or list items, keep in mind that the default value does not get assigned until Form Builder creates a record in the block.

Subordinate mirror items are initialized from the master mirror item's Initial Value property. The ON-SEQUENCE-NUMBER trigger is also taken from the master item. If the subordinate mirror item specifies Initial Value and ON-SEQUENCE-NUMBER, Form Builder ignores them and issues a warning.

At runtime, the initial value set by this property will be ignored if all of the following are true for the item (or an item that mirrors it):

the item is a poplist, T-list, radio group, or check box

there is no element corresponding to the initial value

the item does not allow other values

For compatibility with prior releases, a reference to a form item or to a sequence may be specified with a leading ampersand (&) instead of a leading colon (:).

To specify a raw value that begins with a leading ampersand ('&') or a leading colon (':'), specify two of them (that is, '&&' or '::'). (This is a change in Forms behavior, beginning with Release 6.5.)

Initial Value (Item) property restrictions

- For a text item, the value cannot be outside the range defined by the Lowest Allowed Value and Highest Allowed Value properties.
- For a radio group, the default value must be either the name (not the label) of one of the radio buttons, or the value associated with one of the radio buttons. Form Builder checks first for a radio button name.
- For a list item, the default value must be either the name of one of the list elements, or the value associated with one of the list elements. Form Builder checks first for a list element name.

Insert Allowed (Block) property

Description

Specifies whether records can be inserted in the block.

Applies to block

Set Form Builder , programatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

Yes

Insert Allowed (Item) property

Description

Determines whether an end user can modify the value of an item in a new record (i.e., when the Record_Status is NEW or INSERT).

If you set Insert Allowed to No for an item, the user will not be able to manipulate the item in a new record. For example, the user will not be able to type into a text item, check a check box, or select a radio button.

Applies to text item, check box, list item, radio button, image item, custom items

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_INSTANCE_PROPERTY
- GET_ITEM_PROPERTY
- SET_ITEM_INSTANCE_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Usage Notes

Set Insert Allowed to No when you want the user to be able to inspect an item without being able to modify it. For example, for a system-generated key field, you might set Insert Allowed to No to prevent modification of the key while still displaying it normally (not grayed out).

Set the *Enabled* property to No if you want to prevent an item from responding to mouse events. Disabled items are grayed out to emphasize the fact that they are not currently applicable, while enabled items with Insert Allowed set to No allow the user to browse an item's value with the mouse or keyboard, but not to modify the item's value.

Insert Allowed resembles Update Allowed, which applies to records with a Record_Status of QUERY or CHANGED. For items in database blocks, Insert Allowed, in combination with Update Allowed, lets you control whether the end user can enter or change the value displayed by an item. For items in non-database blocks, setting Insert Allowed to No lets you create a display-only item without disabling it.

If Enabled or Visible is set to No (or PROPERTY_FALSE for runtime), then the items' or item instance's Insert Allowed property is effectively false.

- Setting INSERT_ALLOWED to Yes (or PROPERTY_TRUE for runtime) has no effect at the item instance level unless it is set consistently at the block and item levels. For example, your user cannot type data into an item instance if INSERT_ALLOWED is true at the instance level, but not at the item or block levels.

Insert Allowed (Item) restrictions

- If you are using SET_ITEM_PROPERTY to set Insert Allowed to true, then you must set item properties as follows:

  Enabled to Yes (PROPERTY_TRUE for runtime)

  Visible to Yes (PROPERTY_TRUE for runtime)

When Insert Allowed is specified at multiple levels (item instance, item, and block), the values are ANDed together. This means that setting INSERT_ALLOWED to Yes (PROPERTY_TRUE for runtime) has no effect at the item instance level unless it is set consistently at the block and item levels. For example, your user cannot type data into an item instance if INSERT_ALLOWED is true at the instance level, but not at the item or block levels.

Insert Procedure Arguments property

Description

Specifies the names, datatypes, and values of the arguments to pass to the procedure for inserting data into the data block. The Insert Procedure Arguments property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Insert Procedure Name property

Description

Specifies the name of the procedure to be used for inserting data into the data block. The Insert Procedure Name property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Insert Procedure Result Set Columns property

Description

Specifies the names and datatypes of the result set columns associated with the procedure for inserting data into the data block. The Insert Procedure Result Set Columns property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Interaction Mode property

Description

Specifies the interaction mode for the form module. Interaction mode dictates how a user can interact with a form during a query. If Interaction Mode is set to Blocking, then users are prevented from resizing or otherwise interacting with the form until the records for a query are fetched from the database. If set to Non-Blocking, then end users can interact with the form while records are being fetched.

Non-blocking interaction mode is useful if you expect the query will be time-consuming and you want the user to be able to interrupt or cancel the query. In this mode, the Forms runtime will display a dialog that allows the user to cancel the query.

You cannot set the interaction mode programmatically, however, you can obtain the interaction mode programmatically using the GET_FORM_PROPERTY built-in.

Applies to form module

Set Form Builder

Refer to Built-in

- GET_FORM_PROPERTY

Default Blocking

Required/Optional required

Isolation Mode property

Description

Specifies whether or not transactions in a session will be serializable. If Isolation Mode has the value Serializable, the end user sees a consistent view of the database for the entire length of the transaction, regardless of updates committed by other users from other sessions. If the end user queries and changes a row, and a second user updates and commits the same row from another session, the first user sees Oracle error (ORA-08177: Cannot serialize access.).

Applies to form module

Set Form Builder

Usage Notes

Serializable mode is best suited for an implementation where few users are performing a limited number of transactions against a large database; in other words, an implementation where there is a low chance that two concurrent transactions will modify the same row, and where long-running transactions are queries. For transaction-intensive implementations, leave Isolation Mode set to Read Committed (the default). Serializable mode is best used in conjunction with the block-level property Locking Mode set to Delayed.

Default

Read Committed

Required/Optional required

Item Roles property

Description

Specifies which menu roles have access to a menu item.

Applies to menu item

Set Form Builder

Required/Optional optional

Usage Notes

You can only grant access to members of those roles displayed in the roles list. To add a role to this list, set the menu module property Module Roles.

Item Roles restrictions

Valid only when the name of at least one role has been specified in the menu module roles list.

Item Type property

Description

Specifies the type of item. An item can be one of the following types:

- ActiveX Control (32-bit Windows platforms)
- Bean Area
- Chart Item
- Check Box
- Display Item
- Hierarchical Tree
- Image
- List Item
- OLE Container
- Push Button
- Radio Group
- Sound
- Text Item
- User Area
- VBX Control (Microsoft Windows 3.1 only)

Applies to: items

Set: Form Builder

Default: Text Item

Required/Optional required

Item_Is_Valid property

Description

Specifies whether an item is marked internally as valid.

Applies to item

Set programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

item in a new record: No; item in a queried record: Yes

Usage Notes

- Use Item_Is_Valid to check whether the current status of a text item is valid.
- Set Item_Is_Valid to Yes to instruct Form Builder to treat any current data in an item as *valid* and skip any subsequent validation. Set Item_Is_Valid to No to instruct Form Builder to treat any current data in a text item as *invalid* and subject it to subsequent validation.

Item_Tab_Page property

Description

Specifies the tab page on which an item is placed.

Applies to item

Refer to Built-in

- GET_ITEM_PROPERTY

Default

None

Join Condition property

Description

Defines the relationship that links a record in the detail block with a record in the master block.

Applies to relation

Set Form Builder

Required/Optional required for a relation object

Usage Notes

You can specify a join condition with the following entries:

- an item name that exists in both the master block and the detail block (block_2.item_3)
- an equating condition of two item names, where one item exists in the master block and the other item exists in the detail block
- a combination of item names and equating conditions

Join Condition restrictions

- Maximum length for a join condition is 255 characters.

Join Condition examples

Examples:

To link a detail block to its master block through the ORDID text item that is common to both blocks, define the following join condition:

ORDID

To link the detail block to its master block through a number of text items, define the join condition as follows:

```
block1.item1 = block2.item1 AND block1.item2 = block2.item2
```

Keep in mind that the join condition specifies the relationship between the items in each block, not between the columns in the tables on which those blocks are based. Thus, the items specified must actually exist in the form for the join condition to be valid.

Join Style property

Description

Specifies the join style of the graphic object as either Mitre, Bevel, or Round.

Applies to graphic physical

Set Form Builder

Default

Mitre

Required/Optional optional

Justification property

Description

Specifies the text justification within the item. The allowable values for this property are as follows:

Value	*Description*
Left	Left-justified, regardless of Reading Order property.
Center	Centered, regardless of Reading Order property.
Right	Right-justified, regardless of Reading Order property.
Start	Item text is aligned with the starting edge of the item bounding box. The starting edge depends on the value of the item's Reading Order property. Start is evaluated as Right alignment when the reading order is Right To Left, and as Left alignment when the reading order is Left to Right.
End	Item text is aligned with the ending edge of the item bounding box. The ending edge depends on the value of the item's Reading Order property. End is evaluated as Left alignment when the reading order is Right To Left, and as Right alignment when the reading order is Left to Right.

Applies to display item, text item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Start

Usage Notes

- In unidirectional applications (reading order Left to Right), accept the default, Start, in most cases. For unidirectional applications, Start gives exactly the same results as Left and End gives the same results as Right.
- In bidirectional applications:
- If your data must be aligned with the item's Reading Order, choose Start (the default).

- If your data must be aligned opposite to the item's Reading Order, choose End.
- Unsupported by some window managers.

Keep Cursor Position property

Description

Specifies that the cursor position be the same upon re-entering the text item as when last exited.

Applies to text item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Usage Notes

Use this property if you want to give the end user the flexibility to move the cursor to an item, then back to the partially filled item, and have the cursor reposition itself to the end of the partial text.

Keep Cursor Position restrictions

Unsupported on some window managers.

Key Mode property

Description

Specifies how Form Builder uniquely identifies rows in the database. This property is included for applications that will run against non-ORACLE data sources. For applications that will run against ORACLE, use the default setting.

By default, the ORACLE database uses unique ROWID values to identify each row. Non-ORACLE databases do not include the ROWID construct, but instead rely solely on unique primary key values to identify unique rows. If you are creating a form to run against a non-ORACLE data source, you must use primary keys, and set the Key Mode block property accordingly.

Value	*Description*
Automatic (default)	Specifies that Form Builder should use ROWID constructs to identify unique rows in the datasource but only if the datasource supports ROWID.
Non-Updateable	Specifies that Form Builder should not include primary key columns in any UPDATE statements. Use this setting if your database does not allow primary key values to be updated.
Unique	Instructs Form Builder to use ROWID constructs to identify unique rows in an ORACLE database.
Updateable	Specifies that Form Builder should issue UPDATE statements that include primary key values. Use this setting if your database allows primary key columns to be updated and you intend for the application to update primary key values.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

Unique

Usage Notes

When the Key Mode property is set to one of the primary key modes, you must identify the primary key items in your form by setting the Enforce Primary Key block property to Yes for the block, and the Primary Key item property to Yes for at least one item in the block.

Keyboard Accelerator property

Description

Specifies a logical function key to be associated with a menu item. Accelerator keys are named ACCELERATOR1, ACCELERATOR2, and so on, through ACCELERATOR5. End users can select the menu item by pressing the key or key combination that is mapped to the logical accelerator key.

Applies to menu item

Set Form Builder

Required/Optional optional

Usage Notes

The mappings of logical accelerator keys to physical device keys is defined in the runtime resource file. You must edit the resource file in Oracle Terminal to change the key mappings. You can also create additional accelerator keys in Oracle Terminal (ACCELERATOR6, ACCELERATOR7, and so on), which you can then associate with menu items in a menu module.

Keyboard Accelerator restrictions

- Not valid for separator menu items.
- Key mappings must not interfere with standard Form Builder key mappings.
- When running with bar-style menus, accelerator keys can be used only for items on the menu that is currently displayed.

Keyboard Help Description property

Description

Specifies the key trigger description that is displayed in the runtime Keys help screen if the Display in Keyboard Help property is set to Yes. An entry in the Keys screen includes a text description for the key name and the physical keystroke associated with it, for example, Ctrl-S.

Applies to trigger

Set Form Builder

Default

blank

Usage Notes

- If you do not want the name or the description to appear in the Keys window, set the Display Keyboard Help property to No. This is the default setting.
- If you want the name of the key that corresponds to the trigger and its default description to be displayed in the Keys window, set the Display Keyboard Help property to Yes and leave the Keyboard Help Description blank.
- If you want to replace the default key description, set the Display Keyboard Help property to Yes, then enter the desired description in the Keyboard Help Description field.

Keyboard Help Description restrictions

Valid only for key triggers.

Keyboard Navigable property

Description

Determines whether the end user or the application can place the input focus in the item during default navigation. When set to Yes for an item, the item is navigable. When set to No, Form Builder skips over the item and enters the next navigable item in the default navigation sequence. The default navigation sequence for items is defined by the order of items in the Object Navigator.

Applies to all items except chart items and display items

Set Form Builder, programmatically [NAVIGABLE]

Refer to Built-in

- GET_ITEM_INSTANCE_PROPERTY
- GET_ITEM_PROPERTY
- SET_ITEM_INSTANCE_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Usage Notes

If Enabled or Visible is set to No (PROPERTY_FALSE for runtime), then the items' or item instance's Keyboard navigable property is effectively false. At runtime, when the Enabled property is set to PROPERTY_FALSE, the Keyboard_Navigable property is also set to PROPERTY_FALSE. However, if the Enabled property is subsequently set back to PROPERTY_TRUE, the keyboard Navigable property is NOT set to PROPERTY_TRUE, and must be changed explicitly.

- When Keyboard Navigable is specified at multiple levels (item instance, item, and block), the values are ANDed together. This means that setting Keyboard Navigable to Yes (or NAVIGABLE to PROPERTY_TRUE for runtime) has no effect at the item instance level unless it is set consistently at the item level. For example, your user cannot navigate to an item instance if Keyboard Navigable is true at the instance level, but not at the item level.
- You can use the GO_ITEM built-in procedure to navigate to an item that has its Keyboard Navigable property set to No (PROPERTY_FALSE) for runtime.

Keyboard Navigable restrictions

- If you are using SET_ITEM_PROPERTY to set NAVIGABLE to true, then you must set item properties as follows:
  Enabled to Yes (PROPERTY_TRUE for runtime)
  Visible to Yes (PROPERTY_TRUE for runtime)

Keyboard State property

Description

Specifies supported international keyboard states as Any, Roman Only, or Local Only.

Applies to item international

Set Form Builder

Default

Any

Required/Optional required

Label (Item) property

Description

Specifies the text label that displays for a button, check box, or radio button in a radio group.

Applies to button, check box, radio group button

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_ITEM_INSTANCE_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Default

blank

Required/Optional optional

Label (Menu Item) property

Description

Specifies the text label for each menu item.

Applies to menu item

Set Form Builder, programmatically

Refer to Built-in

- GET_MENU_ITEM_PROPERTY
- SET_MENU_ITEM_PROPERTY

Required/Optional optional

Usage Notes

Each menu item has both a name and a label. The *label*, used only in the runtime GUI, may differ from the *name*, which can be used programmatically.

Unlike the name, which must follow PL/SQL naming conventions, the label can include multiple words and punctuation. For example,*More Info...* is an acceptable label, while the corresponding name would be *more_info*.

When you create a new menu item in the Menu editor, Form Builder gives it a default *name*, like ITEM2, and a default *label*, <New Item>. When you edit the item *label* in the Menu editor, making it, for instance, "Show Keys," the menu item *name* remains ITEM2 until you change it in either the Object Navigator or the Properties Palette.

Label (Menu Substitution Parameter) property

Description

Specifies the label that will prompt the end user to supply a value for the substitution parameter.

Applies to menu substitution parameter

Set Form Builder

Required/Optional optional

Label (Menu Substitution Parameter) restrictions

none

Label (Tab Page) property

Description

The label identifying the tab page. End users click the labelled tab to display the tab pages of a tab canvas.

Applies to tab page

Set Form Builder, programmatically

Refer to Built-in

- GET_TAB_PAGE_PROPERTY
- SET_TAB_PAGE_PROPERTY

Required/Optional optional

Label (Tab Page) restrictions

none

Last_Block property

Description

Specifies the name of the block with the highest sequence number in the form, as indicated by the sequence of blocks listed in the Object Navigator.

Applies to form module

Set not settable

Refer to Built-in

GET_FORM_PROPERTY

Last_Item property

Description

Specifies the name of the item with the highest sequence number in the indicated block, as indicated by the sequence of items listed in the Object Navigator.

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Last_Query property

Description

Specifies the SQL statement for the last query of the specified block.

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Layout Data Block property

Description

Specifies the name of the data block which the frame is associated with; the items within this block are arranged within the frame. A block can only be associated with one frame. You cannot arrange a block's items within multiple frames.

Applies to frame

Set Form Builder

Default

NULL

Required/Optional required

Layout Style property

Description

Specifies the layout style for the items within the frame.

Form	The default frame style. When Frame Style is set is to Form, Form Builder arranges the items in a two-column format, with graphic text prompts positioned to the left of each item.
Tabular	When Frame Style is set to Tabular, Form Builder arranges the items next to each other across a single row, with graphic text prompts above each item.

Applies to frame

Set Form Builder

Default

Form

Required/Optional required

Length (Record Group) property

Description

See Column Specifications.

Line Spacing property

Description

Specifies the line spacing of the text objext as Single, One-and-a-half, Double, Custom. If Custom is selected, the Custom Spacing property determines the line spacing.

Applies to graphic text

Set Form Builder

Default

Single

Required/Optional required

Line Width property

Description

Specifies the width of the object's edge in points (1/72 inch). (Same as Graphics Builder property Edge Width.)

Applies to graphic physical

Set Form Builder

Required/Optional optional

List Item Value property

Description

Specifies the value associated with a specific radio.

Applies to radio button

Set Form Builder

Default

NULL

Required/Optional required

Usage Notes

When you leave the List Item Value field blank, the value associated with the radio button is NULL.

List Item Value restrictions

- Must be unique among values associated with radio button.

List of Values property

Description

Specifies the list of values (LOV) to attach to the text item. When an LOV is attached to a text item, end users can navigate to the item and press [List of Values] to invoke the LOV. To alert end users that an LOV is available, Form Builder displays the LOV list lamp on the message line when the input focus is in a text item that has an attached LOV.

Applies to text item

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Required/Optional optional

List of Values restrictions

The LOV must exist in the active form module.

List Style property

Description

Specifies the display style for the list item, either poplist, combo box, or Tlist. The poplist and combo box styles take up less space than a Tlist, but end users must open the poplist or combo box to see list elements. A Tlist remains "open," and end users can see more than one value at a time if the list is large enough to display multiple values.

Applies to list item

Set Form Builder

Default

Poplist

Usage Notes

The display style you select for the list item has no effect on the data structure of the list item.

List Type property

Description

Specifies how you intend to reference the record group object on which the LOV will be based. Every LOV has an associated record group from which it derives its values at runtime.

Applies to:

List of Values (LOV)

Set Form Builder

Default

Query

Required/Optional required

Usage Notes

The following settings are valid for this property:

Record Group	Indicates that you intend to base the LOV on an existing record group. When you select this option, you must choose the record group in the Record Group property drop-down list. The record group you specify can be either a *static record group* or a *query record group*, and must already exist in the active form.
Old	This option is included for compatibility with previous versions of Form Builder. It cannot be used in new applications.

List Type restrictions

none

List X Position property

Description

Specifies the horizontal (X) coordinate of the upper left corner of the LOV relative to the screen. When you attach an LOV to a text item by setting the List of Values property, you can also set the List X Position and List Y Position properties to override the default display coordinates specified for the LOV.

Applies to text item

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Default

0; indicating that Form Builder should use the default LOV display horizontal (X) coordinate, as specified by the List X Position property.

Required/Optional required

Usage Notes

- If you leave the List X Position property set to 0 and the List Y Position property set to 0, Form Builder displays the LOV at the display coordinates you specified when you created the LOV. If you specify position coordinates, the coordinates you specify override the LOV's default position coordinates.
- The List of Values property must be specified.

List Y Position property

Description

Specifies the vertical (Y) coordinate of the upper left corner of the LOV relative to the screen. When you attach an LOV to a text item by setting the List of Values property, you can also set the List Y Position and List X Position properties to override the default display coordinates specified for the LOV.

Applies to text item

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Default

0; indicating that Form Builder should use the default LOV display vertical (Y) coordinate, as specified by the List Y Position property.

Required/Optional required

Usage Notes

- If you leave the List X Position property set to 0 and the List Y Position property set to 0, Form Builder displays the LOV at the display coordinates you specified when you created the LOV. If you specify position coordinates, the coordinates you specify override the LOV's default position coordinates.
- The List of Values property must be specified.

Listed in Data Block Menu/Data Block Description

Specifies whether the block should be listed in the block menu and, if so, the description that should be used for the block.

Form Builder has a built-in block menu that allows end users to invoke a list of blocks in the current form by pressing [Block Menu]. When the end user selects a block from the list, Form Builder navigates to the first enterable item in the block.

Applies to block

Set Form Builder

Default

Yes. Block Description: For a new block, NULL; For an existing block, the block name at the time the block was created.

Required/Optional optional

Listed in Block Menu/Block Description restrictions

The block does not appear in the Block Menu if you set the Listed in Block Menu property to Yes but leave the Block Description property blank.

Lock Procedure Arguments property

Description

Specifies the names, datatypes, and values of the arguments that are to be passed to the procedure for locking data. The Lock Procedure Arguments property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Lock Procedure Name property

Description

Specifies the name of the procedure to be used for locking data. The Lock Procedure Name property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Lock Procedure Result Set Columns property

Description

Specifies the names and datatypes of the result set columns associated with the procedure for locking data. The Lock Procedure Result Set Columns property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Lock Record property

Description

Specifies that Form Builder should attempt to lock the row in the database that corresponds to the current record in the block whenever the text item's value is modified, either by the end user or programmatically.

Applies to text item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Usage Notes

- Set this property to Yes when the text item is a control item (an item not associated with a base table column), but you still want Form Builder to lock the row in the database that corresponds to the current record in the block.
- Useful for lookup text items where locking underlying record is required.
- To set the Lock Record property with SET_ITEM_PROPERTY, use the constant LOCK_RECORD_ON_CHANGE.

Lock Record restrictions

Valid only when the item is a control item (Base Table Item property set to No) in a data block.

Locking Mode property

Description

Specifies when Form Builder tries to obtain database locks on rows that correspond to queried records in the form. The following table describes the allowed settings for the Locking Mode property:

Value	***Description***
Automatic (default)	Identical to *Immediate* if the datasource is an Oracle database. For other datasources, Form Builder determines the available locking facilities and behaves as much like *Immediate* as possible.
Immediate	Form Builder locks the corresponding row as soon as the end user presses a key to enter or edit the value in a text item.
Delayed	Form Builder locks the row only while it posts the transaction to the database, not while the end user is editing the record. Form Builder prevents the commit action from processing if values of the fields in the block have changed when the user causes a commit action.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

Automatic

Usage Notes

For most applications use the default setting of Automatic.

The Immediate setting remains for compatibility with existing applications, but there is no reason to use it in new applications. Use Automatic instead.

The Delayed setting is useful for applications that must minimize the number of locks or the amount of time that rows remain locked. Use delayed locking if the form's Isolation Mode property has the value *Serializable*.

The main drawbacks of delayed locking are

- The changes an end user makes to a block may need to be redone at commit time.
- Another user's lock can force the first end user to choose between waiting indefinitely or abandoning the changes.

Magic Item property

Description

Specifies one of the the following predefined menu items for custom menus: Cut, Copy, Paste, Clear, Undo, About, Help, Quit, or Window. Magic menu items are automatically displayed in the native style for the platform on which the form is being executed, with the appropriate accelerator key assigned.

Cut, Copy, Paste, Clear, Window, and Quit have built-in functionality supplied by Form Builder, while the other magic menu items can have commands associated with them.

Applies to menu item

Set Form Builder

Default

Cut

Required/Optional optional

Usage Notes

The following settings are valid for this property:

Setting	*Description*
Cut, Copy, Paste, Clear	These items perform the usual text-manipulation operations. Form Builder supplies their functionality, so the designer may not enter a command for these items.
Undo, About	These items have no native functionality, so the designer must enter a command for these items. Any type of command can be used for these items, except Menu.
Help	The command for the Help menu item must be Menu. The designer provides the functionality of items on this submenu.
Quit	Quit also has built-in functionality, so the designer may not assign a command to this item.
Window	The Window magic menu item presents a submenu of all open windows, allowing the user to activate any of them. If the Window magic menu item has a command that invokes a submenu, that submenu will contain both the list of open widows and the user-defined submenu items, in an order determined by Form Builder. The command type for a magic Window item is Null or Menu.

Magic Item restrictions

- Any given magic menu item may appear only once in the entire menu hierarchy for a given menu module. For example, a menu containing the magic menu item Cut cannot be a submenu of two different options in the menu module.
- Leave the magic menu item's Icon, Keyboard Accelerator, and Hint properties blank.

Main Menu property

Description

Specifies the name of the individual menu in the document that is to be the main or starting menu at runtime.

If you are creating a pulldown menu, you will not need to change this property: it is automatically set to the name of the first menu you create, and updated thereafter if you change the name of that menu.

The Main Menu property is used mostly with full-screen menus, to limit the menus to which end users have access. End users cannot navigate to any menu that is above the main menu in the menu module hierarchy.

Applies to menu module

Set Form Builder

Default

blank

Required/Optional required

Usage Notes

When you attach the menu module to a form by setting the appropriate properties in the Form Module property sheet, you can specify a different menu in the document to be the main menu by setting the Initial Menu property.

Mapping of Other Values property

Description

Specifies how any fetched or assigned value that is not one of the pre-defined values associated with a specific list element or radio button should be interpreted.

Applies to list item, radio group

Set Form Builder

Default

blank

Required/Optional optional

Usage Notes

- Leave this property blank to indicate that other values are not allowed for this item or radio group. Any queried record that contains a value other than the user-defined element value is silently rejected. Any attempt to assign an other value is disallowed.
- Any value you specify must evaluate to one of the following references:
- the value associated with one of the list elements or radio groups
- the name (not the label) of one of the list elements

Maximize Allowed property

Description

Specifies that end users can resize the window by using the zooming capabilities provided by the runtime window manager.

Applies to window

Set Form Builder

Default

Yes

Maximize Allowed restrictions

- Only valid when Resize Allowed is set to No

Maximum Length property

Description

Specifies the maximum length of the data value that can be stored in the item.

Applies to all items except buttons, image items, and chart items

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Default

- For a database item, the length of the corresponding column in the database table. **Note:** If the item's data type is NUMBER, the maximum length will be set to the defined column length plus two, to allow for the possible use of a minus sign (for negative values) and a decimal separator.
- For a control item, 30.
- For a LONG item, 240 bytes.

Required/Optional required

Usage Notes

- At runtime, Forms will increase the value of the Maximum Length property if the item’s format mask requires a longer length. (The format mask may be either an explicit mask specified by the form’s designer for this item, or one of the implicit masks used by Forms in its internal conversion operations.)
- For CHAR values, the Maximum Length is 2000 characters.

Note: Bear these considerations in mind if you are writing applications for a multi-byte character set:

- Form Builder allows end users to enter the full number of single-byte characters, up to the Maximum Length specified.
- If the end user enters a combination of single-byte and multi-byte characters that produce a string whose total length in bytes exceeds the item's Maximum Length, the string will be truncated to the nearest whole character and a warning will be displayed. To avoid this situation, consider raising the Maximum Length for the item. (If Maximum Length exceeds the display width of the item, Form Builder will automatically allow the end user to scroll the contents of the item.)

Maximum Length (Form Parameter) property

Description

Specifies the maximum length, in characters, of a form parameter of type CHAR.

Applies to form parameter

Set Form Builder

Default

For a parameter of type CHAR, 30

Required/Optional required

Maximum Length (Form Parameter) restrictions

- Maximum length of a CHAR parameter is 2000 bytes.

Maximum Length (Menu Substitution Parameter) property

Description

Specifies the maximum length, in characters, of a menu substitution parameter.

Applies to menu substitution parameter

Set Form Builder

Default

30

Required/Optional required

Maximum Objects Per Line property

Description

Specifies the maximum number of objects that can appear on each line within a frame.

When the Maximum Number of Frame Objects is set to 0 (the default), there is no maximum--Form Builder arranges the maximum number of objects per line within a frame.

This property is valid when the Frame Style property is set to Form and the Vertical Fill property is set to No.

Applies to frame

Set Form Builder

Default

0

Required/Optional required

Maximum Query Time property

Description

Provides the option to abort a query when the elapsed time of the query exceeds the value of this property.

Applies to form, block

Set Form Builder, Programmatically

Refer to Built-in

- GET_FORM_PROPERTY
- GET_BLOCK_PROPERTY

Required/Optional optional

Usage Notes

This property is only useful when the Query All Records property is set to Yes.

Maximum Records Fetched property

Description

Specifies the number of records fetched when running a query before the query is aborted.

Applies to form, block

Set Form Builder, Programmatically

Refer to Built-in

- GET_FORM_PROPERTY
- GET_BLOCK_PROPERTY

Required/Optional optional

Usage Notes

Maximum Records Fetched is only useful when the properties Query Allowed and Query All Records are set to Yes. Set the Maximum Records Fetched property to limit the records returned by a user's query.

Menu Description property

Description

For applications running on character mode in the pull-down and bar menu display styles, this property specifies the string that displays on the message line when the end user navigates to the menu. In full-screen display style, this property specifies the string that identifies the menu module.

Applies to menu module

Set Form Builder

Default

The default document name

Required/Optional optional

Menu Description restrictions

- Applicable for character mode applications only.

Menu Directory property

Description

Specifies the directory in which Form Builder should look for the .MMX runtime menu file. This property is applicable only when you want Form Builder to locate the menu .MMX runfile through database lookup, rather than direct reference.

When using database lookup, the menu module must be stored in the database. At runtime, Form Builder queries the menu module definition stored in the database to find out the directory and filename of the menu .MMX runfile. The Menu Directory and Menu Filename menu module properties specify the path where Form Builder should look for the .MMX menu file.

Applies to menu module

Set Form Builder

Default

blank

Required/Optional optional

Usage Notes

If you leave the directory path unspecified, Form Builder first searches the default directory for the file, then searches any predefined search paths. For more information on search paths, refer to the Form Builder documentation for your platform.

Menu Directory restrictions

Not valid when using direct reference to specify the location of the menu .MMX runfile. You use direct reference when you attach a menu to a form by setting the Menu Source form module property to Yes.

Menu Filename property

Description

Specifies the name of the .MMX runtime menu file that Form Builder should look for at form startup. This property is applicable only when you want Form Builder to locate the menu runfile through database lookup, rather than direct reference.

To use database lookup, the menu module must be stored in the database. At runtime, Form Builder queries the menu module definition stored in the database to find out the directory and filename of the menu .MMX runfile. The Menu Directory and Menu Filename menu module properties specify the path where Form Builder should look for the .MMX menu file.

Applies to menu module

Set Form Builder

Default

Module Name property

Required/Optional required

Usage Notes

If you leave the directory unspecified, Form Builder first searches the default directory for the file, then searches any predefined search paths. For more information on search paths, refer to the Form Builder documentation for your platform.

Menu Filename restrictions

- The .MMX file extension is not required.

Menu Item Code property

Description

Contains the PL/SQL commands for a menu item.

Applies to menu items

Set Form Builder

Required/Optional required

Usage Notes

Clicking the More… button opens the PL/SQL Editor for the menu item, allowing you to edit the PL/SQL commands.

Menu Item Radio Group property

Description

Specifies the name of the radio group to which the current radio menu item belongs.

Applies to menu item

Set Form Builder

Required/Optional required

Usage Notes

Specify the same Radio Group for all radio items that belong to the same logical set.

Menu Item Radio Group restrictions

- Radio items must be adjacent to each other on the same menu.
- Only one radio group per individual menu is allowed.

Menu Item Type property

Description

Specifies the type of menu item: Plain, Check, Magic, Radio, or Separator. The type determines how the item is displayed and whether the item can have an associated command.

Applies to:

menu items

Set Form Builder

Default

Plain

Usage Notes

The following menu item types are available:

- PlaiDefault. Standard text menu item.

Check	Indicates a Boolean menu item that is either Yes or No, checked or unchecked. Whenever the end user selects a Check menu item Form Builder toggles the state of that item and executes the command associated with that menu item, if there is one.
Magic	Indicates one of the the following predefined menu items: Cut, Copy, Paste, Clear, Undo, About, Help, Quit, and Window. Magic menu items are automatically displayed in the native style of the platform on which the form is executed, in the position determined by the platform's conventions, with the appropriate accelerator key assigned. Cut, Copy, Paste, Clear, Windows, and Quit have built-in functionality supplied by Form Builder, while the other magic menu items require that commands be associated with them.
Radio	Indicates a BOOLEAN menu item that is part of a radio group. Enter a radio group name in the Radio Group property field. One and only one Radio menu item in a group is selected at any given time. When the end user selects a Radio menu item, Form Builder toggles the selection state of the item and executes its command, if there is one.
Separator	A line or other cosmetic item. You specify a Separator menu item for the purpose of separating other menu items on the menu. A Separator menu item cannot be selected and therefore it cannot have a command associated with it.

- You can use GET_MENU_ITEM_PROPERTY and SET_MENU_ITEM_PROPERTY to get and set the state of check and radio menu items.
- Magic menu items Cut, Copy, Clear, and Paste are automatically enabled and disabled by Form

Builder. You can also use GET_MENU_ITEM_PROPERTY and SET_MENU_ITEM_PROPERTY to get and set the state of magic menu items programmatically, but the result of disabling magic menu items will vary, depending on the behavior of the native platform.

Menu Item Type restrictions

The top-level menu should only have plain or magic menu items.

Menu Module property

Description

Specifies the name of the menu to use with this form. When this property is set to Default, Form Builder runs the form with the default menu that is built in to every form. When left NULL, Form Builder runs the form without any menu.

When any value other than Default or Null is specified, the Menu Source property determines how the Menu Module property setting is interpreted:

- When the Menu Source property is Yes, the Menu Module property specifies the name of the menu .MMX runfile that Form Builder should use with this form.
- When the Menu Source property is No, it specifies the name of the menu module in the database that Form Builder should query at form startup to find out the name of the menu .MMX file to use with this form.

Applies to form module

Set Form Builder

Default

Default, indicating that Form Builder should run the form with the default form menu.

Required/Optional optional

Menu Role property

Description

Specifies the security role that Form Builder should use to run the menu. When the Menu Role property is specified, Form Builder runs the indicated menu as if the current end user were a member of the security role specified.

Applies to form module

Set Form Builder

Required/Optional optional

Usage Notes

The Menu Role property is included for backward compatibility only. Its use is not recommended in current applications.

In previous versions of Form Builder, the Menu Role property allowed designers to test a menu by creating a master role that had access to all of the items in the menu, and then running the menu under that role. You can obtain the same result by setting the menu module property Use Security to No. When Use Security is No, all end users have access to all menu items, and you do not have to be a member of any particular role to test your application.

Menu Source property

Description

Menu Source allows you to specify the location of the .MMX runfile when you attach a custom menu to a form module. Form Builder loads the .MMX file at form startup.

Applies to form module

Set Form Builder

Default

Yes

Required/Optional optional

Usage Notes

Setting the Menu Source property allows you to specify the location of the menu .MMX runfile through either direct reference or through database lookup. In most cases, you will want to use direct reference to the file system. Database lookup is included for backward compatibility.

Direct Reference	To refer directly to the .MMX file, set the Menu Source property to Yes, then enter the path/filename of the .MMX file in the Menu Module field.
Database Lookup	To refer to the menu by way of database lookup, set the Menu Source property to No, then enter the name of the menu module stored in the database. At form startup, Form Builder queries the menu module definition to look up the name of the .MMX runfile it needs. (The Menu Module Menu Filename and Menu Directory define the path to the .MMX file in the file system.) When the form is loaded at runtime, Form Builder locates the .MMX file by querying the database to look up the pointer to the .MMX file defined by the menu module Menu Filename and Menu Directory properties.

The following table compares the property settings and database conditions required when attaching a menu to a form through direct reference to those required for database lookup.

Condition or Property	***Direct Reference***	***Database Lookup***
Form Module Property: "Menu Source"	Yes	No
Form Module Property: "Menu Module"	Name of .MMX runfile	Name of .MMB menu design document in database
Menu Module Property:	n/a	Path/filename of .MMX file in

"Menu Directory/Menu Filename"		file system
Database Connection	Not required	Required at form startup
Location of Menu .MMB at Load Time	n/a	Must be stored in database

The following diagrams compare using direct reference and database lookup when attaching a custom menu to a form.

Menu Style property

Description

Specifies the menu display style Form Builder should use to run the custom menu specified by the Menu Module property. Display style options are pull-down or bar.

Applies to form module

Set Form Builder

Default

Pull-down

Required/Optional optional

Menu Style restrictions

Not valid when the Menu Module property is set to DEFAULT. (The default menu runs only in pull-down display style.)

Message property

Description

Specifies the message that is to be displayed in an alert.

Applies to alert

Set Form Builder, programmatically

Refer to Built-in

SET_ALERT_PROPERTY

Required/Optional optional

Message restrictions

Maximum of 200 characters. Note, however, that several factors affect the maximum number of characters displayed, including the font chosen and the limitations of the runtime window manager.

Minimize Allowed property

Description

Specifies that a window can be iconified on window managers that support this feature.

Applies to window

Set Form Builder

Default

Yes

Required/Optional optional

Minimize Allowed restrictions

Cannot be set for a root window: a root window is always iconifiable.

Minimized Title property

Description

Specifies the text string that should appear below an iconified window.

Applies to window

Set Form Builder

Default

No

Required/Optional optional

Minimized Title restrictions

Only applicable when the Minimize Allowed property is set to Yes.

Modal property

Description

Specifies whether a window is to be modal. Modal windows require an end user to dismiss the window before any other user interaction can continue.

Applies to window

Set Form Builder

Default

No

Modal restrictions

- When Modal is set to Yes, the following window properties are ignored:

- Close Allowed
- Icon Filename
- Minimize Allowed
- Move Allowed
- Show Vertical/Horizontal Scroll Bar
- Resize Allowed
- Minimized Title
- Inherit Menu
- Maximize Allowed

Module_NLS_Lang property

Description

Specifies the complete current value of the NLS_LANG environment variable defined for the form, for national language support. MODULE_NLS_LANG is the equivalent of concatenating the following properties:

- MODULE_NLS_LANGUAGE (language only)
- MODULE_NLS_TERRITORY (territory only)
- MODULE_NLS_CHARACTER_SET (character set only)

Applies to form

Set Not settable from within Form Builder. Set at your operating system level.

Refer to Built-in

GET_FORM_PROPERTY

Default

Default is usually "America_American.WE8ISO8859P1," but all the defaults can be port-specific.

Module Roles property

Description

Specifies which database roles are available for items in this menu module.

Applies to menu module

Set Form Builder

Required/Optional optional

Usage Notes

Use Menu Module Roles to construct the entire list of roles with access to this menu module, then use Menu Item Roles to specify which of these roles should have access to a specific menu item.

Mouse Navigate property

Description

Specifies whether Form Builder should perform navigation to the item when the end user activates the item with a mouse.

Applies to button, check box, list item, radio group

Set Form Builder

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Usage Notes

When Mouse Navigate is No, Form Builder does not perform navigation to the item when the end user activates it with the mouse. For example, a mouse click in a button or check box is *not* treated as a navigational event. Form Builder fires any triggers defined for the button or check box (such as When-Button-Pressed), but the input focus remains in the current item.

When Mouse Navigate is Yes, Form Builder navigates to the item, firing any appropriate navigation and validation triggers on the way.

Mouse Navigate restrictions

Applies only in mouse-driven environments.

Mouse Navigation Limit property

Description

Determines how far outside the current item an end user can navigate with the mouse. Mouse Navigation Limit can be set to the following settings:

Form	(The default.) Allows end users to navigate to any item in the current form.
Block	Allows end users to navigate only to items that are within the current block.
Record	Allows end users to navigate only to items that are within the current record.
Item	Prevents end users from navigating out of the current item. This setting prevents end users from navigating with the mouse at all.

Applies to form

Set Form Builder

Default

Form

Move Allowed property

Description

Specifies whether or not the window can be moved .

Windows can be moved from one location to another on the screen by the end user or programmatically by way of the appropriate built-in subprogram.

Applies to window

Set Form Builder

Default

Yes

Required/Optional optional

Usage Notes

In general, it is recommended that windows always be movable.

Move Allowed restrictions

Cannot be set to NO for a window with the name of ROOT_WINDOW. Such a window is always movable.

Multi-Line property

Description

Determines whether the text item is a single-line or multi-line editing region.

Applies to text item

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Default

No

Usage Notes

Setting the Multi-line property Yes allows a text item to *store* multiple lines of text, but it does not automatically make the item large enough to *display* multiple lines. It is up to you to set the Width, Height, Font Size, and Maximum Length properties to ensure that the desired number of lines and characters are displayed.

Single-line	Pressing the carriage return key while the input focus is in a single-line text item initiates a [Next Item] function.
Multi-line	Pressing the carriage return key while the input focus is in a multi-line text item starts a new line in the item.

Multi-Line restrictions

Valid only for text items with data type CHAR, ALPHA, or LONG.

Multi-Selection property

Description

Indicates whether multiple nodes may be selected at one time. If set to FALSE, attempting to select a second node will deselect the first node, leaving only the second node selected.

Applies to hierarchical tree

Set Form Builder

Default

False

Required/Optional required

Name property

Description

Specifies the internal name of the object. Every object must have a valid name that conforms to Oracle naming conventions.

Applies to all objects

Set Form Builder

Default

OBJECT_CLASS_N, where *OBJECT_CLASS* is the type of object, and *N* is the next available number in the document; for example, BLOCK5 or EDITOR3.

Required/Optional required

Usage Notes

- For menu items and radio buttons, the Name property has unique characteristics:
- The Name property specifies an internal handle that does not display at runtime.
- The Name property is used to refer to the menu item or radio button in PL/SQL code.
- The Label property specifies the text label that displays for the menu item or current radio button.

For menu substitution parameters, the following restrictions apply:

- Restricted to a two-character identifier for the substitution parameter.
- Must be alphanumeric.
- Must start with an alphabetic character.
- When referencing the parameter in a menu command line, the parameter must be preceded by an ampersand (&RN)
- In a PL/SQL reference, the parameter must be preceded by a colon (:SS).

Name restrictions

- Can be up to 30 characters long
- Must begin with a letter
- Can contain letters, numbers, and the special characters $, #, @ and _ (underscore)
- Are not case sensitive
- Must uniquely identify the object:
- Item names must be unique among item names in the same block
- Relation names must be unique among relations that have the same master block
- Cannot be set for the root window

Name examples

Example

ENAME, ADDRESS1, PHONE_NO1

Navigation Style property

Description

Determines how a Next Item or Previous Item operation is processed when the input focus is in the last navigable item or first navigable item in the block, respectively.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

Same Record

Usage Notes

The following settings are valid for this property:

Same Record	The default navigation style. A Next Item operation from the block's last navigable item moves the input focus to the first navigable item in the block, *in that same record*.
Change Record	A Next Item operation from the block's last navigable 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, Form Builder creates a new record. If there is an open query in the block (the block contains queried records), Form Builder retrieves additional records as needed.
Change Block	A Next Item operation from the block's last navigable item moves the input focus to the first navigable item in the first record of the next block. Similarly, a Previous Item operation from the first navigable item in the block moves the input focus to the last item in the current record of the previous block. The Next Navigation Block and Previous Navigation Block properties can be set to redefine a block's "next" or "previous" navigation block.

Next Navigation Block property

Description

Specifies the name of the block that is defined as the"next navigation block" with respect to this block. By default, this is the block with the next higher sequence number in the form, as indicated by the order of blocks listed in the Object Navigator. However, you can set this property to redefine a block's "next" block for navigation purposes.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

The name of the block with the next higher sequence number, as indicated by the order of blocks listed in the Object Navigator.

Required/Optional optional

Usage Notes

Setting this property does not change the value of the NextBlock property.

Next Navigation Item property

Description

Specifies the name of the item that is defined as the "next navigation item" with respect to this current item. By default, the next navigation item is the item with the next higher sequence as indicated by the order of items in the Object Navigator. However, you can set this property to redefine the "next item" for navigation purposes.

Applies to item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

NULL. NULL indicates the default sequence, which is the name of the item with the next higher sequence number.

Next Navigation Item restrictions

The item specified as Next Navigation Item must be in the same block as the current item.

NextBlock property

Description

Specifies the name of the block with the next higher sequence number in the form, as indicated by the order of blocks listed in the Object Navigator.

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Usage Notes

- You can programmatically visit all of the blocks in a form by using GET_BLOCK_PROPERTY to determine the First_Block and NextBlock values.
- The value of NextBlock is NULL when there is no block with a higher sequence number than the current block.
- Setting the Next Navigation Block property has no effect on the value of NextBlock.

NextItem property

Description

Specifies the name of the item with the next higher sequence number in the block, as indicated by the order of items listed in the Object Navigator.

Applies to item

Set not settable

Refer to Built-in

GET_ITEM_PROPERTY

Next_Detail_Relation property

Description

Returns the name of the relation that has the same detail block as the specified relation. If no such relation exists, returns NULL.

Applies to relation

Set not settable

Refer to Built-in

GET_RELATION_PROPERTY

Usage Notes

Use this property with the FIRST_DETAIL_RELATION property to traverse a list of relations for a given master block.

Next_Master_Relation property

Description

Returns the name of the next relation that has the same master block as the specified relation. If no such relation exists, returns NULL.

Applies to relation

Set not settable

Refer to Built-in

GET_RELATION_PROPERTY

Usage Notes

Use this property with the FIRST_MASTER_RELATION property to traverse a list of relations for a given master block.

Number of Items Displayed property

Description

Specifies the number of item instances displayed for the item when the item is in a multi-record block.

Setting Number of Items Displayed > 0 overrides the Number of Records Displayed block property.

Applies to item

Set Form Builder

Default

Zero. Zero indicates that the item should display the number of instances specified by the Number of Records Displayed block property.

Required/Optional optional

Usage:

Use Number of Items Displayed to create a single button, chart, OLE item, image, VBX control (in 16-bit Microsoft Windows), or ActiveX control (in 32-bit Windows) as part of a multi-record block. For instance, if Number of Records Displayed is set to 5 to create a multi-record block and you create a button, by default you will get 5 buttons, one per record. To get only one button, set Number of Items Displayed to 1.

Number of Items Displayed restrictions

Number of Items Displayed must be <= Number of Records Displayed block property setting.

Number of Records Buffered property

Description

Specifies the minimum number of records buffered in memory during a query in the block.

Applies to block

Set Form Builder

Default

NULL; which indicates the minimum setting allowed (the value set for the Number of Records Displayed property plus a constant of 3).

Required/Optional optional

Usage Notes

- Form Builder buffers any additional records beyond the maximum to a temporary file on disk.
- Improve processing speed by increasing the number of records buffered.
- Save memory by decreasing the number of records buffered. This can, however, result in slower disk I/O.
- If you anticipate that the block may contain a large number of records either as the result of a query or of heavy data entry, consider raising the Number of Records Buffered property to increase performance.
- Consider lowering the Number of Records Buffered property if you anticipate retrieving large items, such as image items, because of the amount of memory each item buffered may require.

Number of Records Buffered restrictions

- If you specify a number lower than the minimum, Form Builder returns an error when you attempt to accept the value.

Number of Records Displayed property

Description

Specifies the maximum number of records that the block can display at one time. The default is 1 record. Setting Number of Records Displayed greater than 1 creates a multi-record block.

Applies to block

Set Form Builder

Refer to Built-in

GET_BLOCK_PROPERTY

Default

1

Required/Optional required

OLE Activation Style property

Description

Specifies the event that will activate the OLE containing item.

Applies to OLE Container

Set Form Builder

Default

Double Click

Usage Notes

The following settings are valid for this property:

Double Click	The default OLE activation style. An OLE object becomes active by double-clicking anywhere on the OLE object.
• Focus-i	Navigating to the OLE object causes the OLE object to become active.
Manual	An OLE object becomes active by selecting Edit or Open from the Object submenu of the OLE popup menu. The Show OLE Popup Menu property must be set to YES and the Object menu item must be set to displayed and enabled. The OLE popup menu is accessible when the mouse cursor is on the OLE object and the right mouse button is pressed.

If the Show OLE Popup Menu property is YES and the Object menu item is displayed and enabled, it is also possible to manually activate the OLE object through the OLE popup menu when the OLE Activation Style is Double Click or Focus-in.

OLE Activation Style restrictions

Valid only on Microsoft Windows and Macintosh.

OLE Class property

Description

Determines what class of OLE objects can reside in an OLE container. The following settings are valid for this property:

NULL	The default OLE class. You can insert any kind of OLE object class specified in the registration database in an OLE container.
other than NULL	Only OLE objects from the specified class can be inserted in an OLE container at runtime. The OLE object classes that are available for selection depend on information contained in the registration database. The content of the registration database is determined by the OLE server applications installed on your computer.

Applies to OLE Container

Set Form Builder

Default

NULL

Usage Notes

You select a specific class if you want to create an application that allows end users to change the current OLE object in the OLE container, but want to restrict the end users to creating OLE objects from a particular class.

OLE Class restrictions

Valid only on Microsoft Windows and Macintosh.

OLE In-place Activation property

Description

Specifies if OLE in-place activation is used for editing embedded OLE objects. The following settings are valid for this property:

YES	Turns on OLE in-place activation. OLE in-place activation is used for editing embedded OLE objects; linked objects are activated with external activation.
NO	Turns off OLE in-place activation and turns on external activation. External Activation is used for editing embedded or linked OLE objects.

Applies to OLE Container

Set Form Builder

Default

NO

OLE In-place Activation restrictions

- Valid only on Microsoft Windows and Macintosh.

OLE Inside-Out Support property

Description

Specifies if the OLE server of the embedded object allows inside-out object support during in-place activation. Inside-out activation allows for more than one embedded object to have an active editing window within an OLE container. The following settings are valid for this property:

YES	Turns on inside-out object support for embedded objects that have the OLE In-place Activation property set to Yes.
NO	Turns off inside-out object support for embedded objects that have the OLE in-place Activation property set to Yes.

Applies to OLE Container

Set Form Builder

Default

YES

OLE Inside-Out Support restrictions

- Valid only on Microsoft Windows and Macintosh.

OLE Popup Menu Items property

Description

Determines which OLE popup menu commands are displayed and enabled when the mouse cursor is on the OLE object and and the right mouse button is pressed. The OLE popup menu commands manipulate OLE objects. OLE popup menu commands and their actions include:

OLE Popup Menu Command	***Action***
CUT	Cuts an OLE object and places the content on the clipboard.
COPY	Copies an OLE object and places the content on the clipboard.
PASTE	Pastes the content from the clipboard to an OLE container.
PASTE SPECIAL	Pastes an OLE object from the clipboard to an OLE container in a format other than the original format.
INSERT OBJECT	Inserts an OLE object in an OLE container.
DELETE OBJECT	Deletes an OLE object from an OLE container.
LINKS	Invokes a dialog that has settings to determine how links are updated, edit linked source files, and change links from one source file to another source file.
OBJECT	Depending on the OLE server, it is possible to perform various operations on an OLE object. Some examples include opening an OLE object, editing an OLE object, and converting an OLE object from one format to another.

Applies to OLE Container

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Display On and Enable On for all menu commands

Required/Optional required

Usage Notes

- In the Form Builder, you can set each OLE popup menu command to exhibit the following characteristics by selecting the appropriate check box:

Display	Specifies whether the selected menu command is displayed.
Enable	Specifies whether a menu command that has Display On is enabled or disabled. A disabled item appears dimmed or grayed.

- In addition to setting OLE popup menu command properties in the Form Builder, you can set and get OLE popup menu command properties programmatically. To set or get the OLE popup menu commands programmatically, use a programmatic property name that corresponds to a menu command. The following list includes each of the OLE popup menu commands and a corresponding programmatic property name:

Menu Command	***Programmatic Property Name***
Cut	POPUPMENU_CUT_ITEM
Copy	POPUPMENU_COPY_ITEM
Paste	POPUPMENU_PASTE_ITEM
Paste Special	POPUPMENU_PASTESPEC_ITEM
Insert Object	POPUPMENU_INSOBJ_ITEM
Delete Object	POPUPMENU DELOBJ ITEM
Links	POPUPMENU_LINKS_ITEM
Object	POPUPMENU_OBJECT_ITEM

- You can programmatically set the OLE popup menu command properties to any of the following values:

DISPLAYED	Specifies that an OLE popup menu command is displayed and enabled.
ENABLED	Specifies that an OLE popup menu command is displayed and disabled. A disabled item appears dimmed or grayed.

- HIDDESpecifies that an OLE popup menu command is not displayed on the OLE popup menu. A command that is not displayed is not enabled.

In addition to the values that you can set programmatically, you can programmatically get the following values from each of the OLE popup menu commands:

DISPLAYED	Return value when an OLE popup menu command is displayed and enabled.
ENABLED	Return value when an OLE popup menu command is displayed and disabled. A disabled item appears dimmed or grayed.

- HIDDEReturn value when an OLE popup menu command is not displayed

	on the OLE popup menu. A command that is not displayed is not enabled.
UNSUPPORTED	Return value when the OLE popup menu is not supported. This is the return value for every platform except Microsoft Windows.

OLE Popup Menu Items restrictions

Valid only on Microsoft Windows.

OLE Resize Style property

Description

Determines how an OLE object is displayed in an OLE container. The following settings are valid for this property:

CLIP	The default OLE resize style. An OLE object is cropped to fit into an OLE container.
SCALE	An OLE object is scaled to fit into an OLE container.
INITIAL	An OLE container is resized to fit an OLE object at creation time only.
DYNAMIC	An OLE container is resized to fit an OLE object whenever the OLE object size changes.

Applies to OLE Container

Set Form Builder

Required/Optional required

Default

CLIP

OLE Resize Style restrictions

Valid only on Microsoft Windows and Macintosh.

OLE Tenant Aspect property

Description

Determines how an OLE object appears in an OLE container.

Applies to OLE Container

Set Form Builder

Default

CONTENT

Usage Notes

The following settings are valid for this property:

CONTENT	The default OLE tenant aspect. The content of an OLE object is displayed in an OLE container. The content of the OLE object depends on the value of the OLE Resize Style property and can either be clipped, scaled, or full size.
ICO	An icon of an OLE object is displayed in an OLE container. The default icon is one that represents the OLE server application that created the OLE object. You can choose which icon to use from the insert object dialog.
THUMBNAIL	A reduced view of the OLE object is displayed in an OLE container.

An OLE object type is saved to the database in a LONG RAW column. When the OLE object is queried from the database, make sure that it has the same OLE Tenant Aspect property setting as that of the OLE object saved to the database. If the OLE Tenant Aspect property of the saved OLE object is different from that of the queried OLE object, the record containing the object is automatically LOCKED.

OLE Tenant Aspect restrictions

Valid only on Microsoft Windows.

OLE Tenant Types property

Description

Specifies the type of OLE objects that can be tenants of the OLE container. The following settings are valid for this property:

ANY	The default OLE tenant type. Any OLE object can be a tenant of the OLE container.
NONE	No object can reside in the OLE container.
STATIC	Only static OLE objects can be a tenant of the OLE container. A static OLE object is a snapshot image of a linked OLE object that has a broken link to its source. A static OLE object cannot be modified.
EMBEDDED	Only an embedded OLE object can be a tenant of the OLE container.
LINKED	Only a linked OLE object can be a tenant of the OLE container.

Applies to OLE Container

Set Form Builder

Default

ANY

OLE Tenant Types restrictions

Valid only on Microsoft Windows and Macintosh.

Operating_System property

Description

Specifies the name of the current operating system, such as Microsoft WINDOWS, WIN32COMMON, UNIX, Sun OS, MACINTOSH, VMS, and HP-UX.

Applies to application

Set not settable

Refer to Built-in

GET_APPLICATION_PROPERTY

Usage Notes

Because the value returned by this property is platform-specific, refer to the Form Builder documentation for your operating system if the platform you are using is not listed above.

Optimizer Hint property

Description

Specifies a hint string that Form Builder passes on to the RDBMS optimizer when constructing queries. Using the optimizer can improve the performance of database transactions.

Applies to block

Set Designer, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Restrictions:

Valid only for applications running against the Oracle7 Server or Oracle8 Server.

Usage Notes

Consider a form that contains a block named *DeptBlock* based on the DEPT table. If the end user enters a criteria of " > 25 " for the DEPTNO column and executes the query, the default SELECT statement that Form Builder generates to query the appropriate rows from the database is as follows:

```
SELECT DEPTNO,DNAME,LOC,ROWID
FROM DEPT
WHERE (DEPTNO > 25)
```

The designer can use SET_BLOCK_PROPERTY to set the Optimizer Hint property to request that the Oracle7 Server attempt to optimize the SQL statement for best response time:

```
Set_Block_Property('DeptBlock',OPTIMIZER_HINT,'FIRST_ROWS');
SELECT /*+ FIRST_ROWS */ DEPTNO,DNAME,LOC,ROWID
FROM DEPT
WHERE (DEPTNO > 25)
```

For more information on how to use this feature with Oracle7, refer to the following sources:

- *Oracle7 Server Application Developer's Guide*, Chapter 5, "Tuning SQL Statements"
- *Oracle7 Server Concepts Manual*, Chapter 13, "The Optimizer"

Order By property

Description

See WHERE CLAUSE/ORDER BY CLAUSE.

Other Reports Parameters property

Description

A <keyword>=<value> list of parameters to include in the running of the report. For a list of valid parameters, see the keyword list in the Report Builder online help.

Applies to Report Builder reports

Set Form Builder

Default

blank

Required/Optional optional

Usage Notes:

When passing multi-word parameter values in the where-clause, the entire where-clause should be enclosed in single quotes. When a name appears in such a multi-word parameter, then two single quotes should also be used to begin and to end that name. For example, in order to pass the parameter value where ename = 'MILLER' it is necessary to code this as:

```
'where ename = ''MILLER'''
```

Output_Date/Datetime_Format property

Description

Holds the current output date or datetime format mask established by the environment variable FORMSnn_OUTPUT_DATE_FORMAT or FORMSnn_OUTPUT_DATETIME_FORMAT. Forms uses these format masks as defaults in its runtime output processing.

There are two separate properties: Output_Date_Format and Output_Datetime_Format.

Applies to application

Set Not settable from within Form Builder.

Refer to Built-in

GET_APPLICATION_PROPERTY

Parameter Data Type property

Description

Specifies what kinds of values Form Builder allows as input and how Form Builder displays those values.

Applies to check box, display item, list item, radio group, text item, custom item, form parameter

Note: All data types do not apply to each item type.

Set Form Builder

Usage Notes

- It is recommended that you use only the standard data types CHAR, DATE, LONG, and NUMBER. These data types are based on native ORACLE data types, and offer better performance and application portability. The other data types are valid only for text items, and are included primarily for compatibility with previous versions. You can achieve the same formatting characteristics by using a standard data type with an appropriate format mask.
- The data type of a base table item must be compatible with the data type of the corresponding database column. Use the CHAR data type for items that correspond to ORACLE VARCHAR2 database columns.
- Do not create items that correspond to database CHAR columns if those items will be used in queries or as the join condition for a master-detail relation; use VARCHAR2 database columns instead.
- Form Builder will perform the following actions on items, as appropriate:
- remove any trailing blanks
- change the item to NULL if it consists of all blanks
- remove leading zeros if the data type is NUMBER, INT, MONEY, RINT, RMONEY, or RNUMBER (unless the item's format mask permits leading zeros)
- The form parameter Parameter Data Type property supports the data types CHAR, DATE, and NUMBER.

ALPHA

Contains any combination of letters (upper and/or lower case).

Default	Blanks
Example	"Employee", "SMITH"

CHAR

Supports VARCHAR2 up to 2000 characters. Contains any combination of the following characters:

- Letters (upper and/or lower case)
- Digits

- Blank spaces
- Special characters ($, #, @, and _)

Default	Blanks
Example	"100 Main Street", "CHAR_EXAMPLE_2"

DATE

Contains a valid date. You can display a DATE item in any other valid format by changing the item's format mask.

Default	DD-MON-YY
Restrictions	Refers to a DATE column in the database and is processed as a true date, not a character string.
The DATE data type contains a ZERO time component	.
Example	01-JAN-92

DATETIME

Contains a valid date and time.

Default	DD-MON-YY HH24:MI[:SS]
Restrictions	Refers to a DATE column in the database and is processed as a true date, not a character string.
	The DATETIME data type contains a four digit year. If the year input to a DATETIME data type is two digits, the year is interpreted as 00YY.
Example	31-DEC-88 23:59:59

EDATE

Contains a valid European date.

Default	DD/MM/YY
Restrictions	V3 data type.
	Must refer to a NUMBER column in the database.
	Included for backward compatibility. Instead, follow these recommendations:
	Use the DATE data type.
	Apply a format mask to produce the European date format.
	Reference a DATE column in the database, rather than a NUMBER column.
Example	23/10/92 (October 23, 1992)
	01/06/93 (June 1, 1993)

INT

Contains any integer (signed or unsigned whole number).

Default	0
Example	1, 100, -1000

JDATE

Contains a valid Julian date.

Default	MM/DD/YY
Restrictions	V3 data type.
	Must refer to a NUMBER column in the database.
	Included for backward compatibility. Instead, follow these recommendations:
	Use the DATE data type.
	Apply a format mask to produce the Julian date format.
	Reference a DATE column in the database, rather than a NUMBER column.
Example	10/23/92 (October 23, 1992)
	06/01/93 (June 1, 1993)

LONG

Contains any combination of up to 65,534 characters. Stored in ORACLE as variable-length character strings.

Default	Blanks
Restrictions	Not allowed as a reference in the WHERE or ORDER BY clauses of any SELECT statement.
	LONG items are not queryable in Enter Query mode.

MONEY

Contains a signed or unsigned number to represent a sum of money.

Restrictions	V3 data type
	Included for backward compatibility. Instead, use a format mask with a number to produce the same result.
Example	10.95, 0.99, -15.47

NUMBER

Contains fixed or floating point numbers, in the range of 1.0x10-129 to 9.99x10124, with one or more of the following characteristics:

- signed
- unsigned
- containing a decimal point
- in regular notation

- in scientific notation
- up to 38 digits of precision

NUMBER items refer to NUMBER columns in the database and Form Builder processes their values as true numbers (not character strings).

Default	0
Restrictions	Commas cannot be entered into a number item (e.g., 99,999). Use a format mask instead.
Example	-1, 1, 1.01, 10.001, 1.85E3

RINT

Displays integer values as right-justified.

Restrictions	V3 data type
	Included for backward compatibility. Instead, follow these recommendations:
	Use the NUMBER data type.
	Apply a format mask such as 999 to produce a right-justified number.

RMONEY

Displays MONEY values as right-justified.

Restrictions	V3 data type
	Included for backward compatibility. Instead, follow these recommendations:
	Use the NUMBER data type
	Apply a format mask such as $999.99 to produce a right-justified number.

RNUMBER

Displays NUMBER values as right-justified.

Restrictions	V3 data type
	Included for backward compatibility. Instead, follow these recommendations:
	Use the NUMBER data type.
	Apply a format mask such as 999.999 to produce a right-justified number.

TIME

Contains numbers and colons that refer to NUMBER columns in the database.

Default	HH24:MI[:SS]
Restrictions	V3 data type
	Included for backward compatibility. Instead, follow these recommendations:
	Use the DATETIME data type.
	Apply a format mask to produce only the time.

	Not allowed as a reference to DATE columns in the database.
Example	:10:23:05
	21:07:13

Parameter Initial Value (Form Parameter) property

Description

Specifies the value that Form Builder assigns the parameter at form startup.

Applies to Form Parameter

Set Form Builder

Default

NULL

Required/Optional optional

Usage Notes

Any valid constant is a valid value for this property.

Menu Parameter Initial Value (Menu Substitution Parameter) property

Description

Specifies the value that Form Builder assigns the parameter at form startup.

Set Form Builder

Required/Optional required

Password property

Description

Specifies the password of the current end user.

Applies to application

Set not settable

Refer to Built-in

GET_APPLICATION_PROPERTY

Usage Notes

The Password property returns only the password. If you want a connect string as well, examine the Connect_String property.

PLSQL_Date_Format property

Description

This property establishes the format mask used in converting date values when executing PL/SQL (for a trigger or called function or procedure) within Forms, in the following cases:

- evaluating TO_DATE (char_value) or TO_DATE (date_value) with no explicit format mask
- assigning a CHAR value to a date variable, or vice versa.

Applies to entire Forms application (global value)

Set programmatically

Refer to Built-in

- GET_APPLICATION_PROPERTY built-in
- SET_APPLICATION_PROPERTY built-in

Required/Optional optional. However, it is STRONGLY RECOMMENDED that, for a new application, you set this property to a format mask containing full century and time information. It is also recommended that this format mask be the same as the one specified in the Builtin_Date_Format property.

Default

As noted above, it is strongly recommended that you explicitly set this value for a new application. If you do not, the default value will be DD-MON-YY. (This value is used for compatibility with Release 4.5 and earlier.)

Compatibility with other Oracle products

In Oracle products other than Form Builder, PL/SQL version 2 does not necessarily use a default date format mask of DD-MON-YY. Instead, it typically uses a format mask derived from the current NLS environment. If for some reason you want your Forms application to exhibit the same behavior, you can use the USER_NLS_DATE_FORMAT application property to get the current NLS date format mask, and then assign it to the application's PLSQL_DATE_FORMAT property.

PL/SQL Library Location property

Description

Shows the location of the attached PL/SQL library.

This property is set when you attach a PL/SQL library to a Forms module. If you requested that the directory path be retained, this property will be the full pathname of the library. If you requested that the directory path be removed, this property will be just the library name.

This property is displayed only for your information. You cannot set or change it through the property palette.

Applies to PL/SQL libraries

Set Form Builder

Required/Optional display only.

Default none

PL/SQL Library Source property

Description

This property is set when you attach a PL/SQL library to a Forms module. It shows the source of this PL/SQL library – either File or Database.

This property is displayed only for your information. You cannot set or change it through the property palette.

Applies to PL/SQL libraries

Set Form Builder

Required/Optional display only

Default File

Popup Menu property

Description

Specifies the popup menu to display for the canvas or item.

Applies to canvases and items

Set not settable

Required/Optional Optional

Default

NULL

Refer to Built-in

GET_MENU_ITEM_PROPERTY

- SET__MENU_ITEM_PROPERTY

Note: ENABLED, DISABLED, and LABEL are the only properties valid for popup menus.

Popup Menu restrictions

The popup menu must be defined within the current form module.

- You cannot attach a popup menu to individual radio buttons, but you can assign a popup menu to a radio group.

Precompute Summaries property

Description

Specifies that the value of any summarized item in a data block is computed before the normal query is issued on the block. Form Builder issues a special query that selects all records (in the database) of the summarized item and performs the summary operation (sum, count, etc.) over all the records.

Applies to block

Set Form Builder

Required/Optional Required if the block contains summarized items and the block's Query All Records property is set to No.

Default

No

Usage Notes

When an end user executes a query in a block with Precompute Summaries set to Yes, Form Builder fires the Pre-Query trigger (if any) once before it executes the special query. Form Builder fires the Pre-Select trigger (if any) twice: once just before executing the special query, and again just before executing the normal query.

Precompute Summaries restrictions

You cannot set Precompute Summaries to Yes if any of the following are true: (1) the block contains a summarized control item, (2) a minimize or maximize operation is performed on a summarized item in the block, (3) the block's Query Data Source is a stored procedure or transactional triggers (must be a table or sub-query), or (4) the block contains a checkbox item, list item, or radio group with an empty Other Values property..

- Read consistency cannot be guaranteed unless (1) the form is running against an Oracle7.3 database, and (2) the form-level Isolation Mode property is set to Serializable.

Prevent Masterless Operations property

Description

Specifies whether end users should be allowed to query or insert records in a block that is a detail block in a master-detail relation. When set to Yes, Form Builder does not allow records to be inserted in the detail block when there is no master record in the master block, and does not allow querying in the detail block when there is no master record that came from the database.

When Prevent Masterless Operation is Yes, Form Builder displays an appropriate message when end users attempt to insert or query a record:

```
FRM-41105: Cannot create records without a parent record.
FRM-41106: Cannot query records without a parent record.
```

Applies to relation

Set Form Builder, programmatically

Refer to Built-in

- GET_RELATION_PROPERTY
- SET_RELATION_PROPERTY

Default

No

Previous Navigation Block property

Description

Specifies the name of the block that is defined as the "previous navigation block" with respect to this block. By default, this is the block with the next lower sequence in the form, as indicated by the order of blocks in the Object Navigator. However, you can set this property to redefine a block's "previous" block for navigation purposes.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

The name of the block with the next lower sequence in the form.

Required/Optional optional

Usage Notes

Setting this property has no effect on the value of the PreviousBlock property.

Previous Navigation Item property

Description

Specifies the name of the item that is defined as the"previous navigation item" with respect to the current item. By default, this is the item with the next lower sequence in the form, as indicated by the order of items in the Object Navigator. However, you can set this property to redefine the "previous item" for navigation purposes.

Applies to item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

NULL. NULL indicates the default, which is the name of the item with the next lower sequence in the form.

Required/Optional optional

Previous Navigation Item restrictions

The item specified as Previous Navigation Item must be in the same block as the current item.

PreviousBlock property

Description

Specifies the name of the block with the next lower sequence in the form, as indicated by the order of blocks in the Object Navigator.

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Required/Optional optional

Usage Notes

- You may use this property with the First_Block or Last_Block form properties to traverse a list of blocks.
- The value of PreviousBlock is NULL when there is no block with a lower sequence number than the current block.
- Setting the Previous Navigation Block property has no effect on the value of PreviousBlock.

PreviousItem property

Description

Specifies the name of the item with the next lower sequence number in the block, as indicated by the order of items in the Object Navigator.

Applies to item

Set not settable

Refer to Built-in

GET_ITEM_PROPERTY

Required/Optional optional

Primary Canvas property

Description

Specifies the canvas that is to be the window's *primary content view.* At runtime, Form Builder always attempts to display the primary view in the window. For example, when you display a window for the first time during a session by executing the SHOW_WINDOW built-in procedure, Form Builder displays the window with its primary content view.

If, however, Form Builder needs to display a different content view because of navigation to an item on that view, the primary content view is superseded by the target view.

Applies to window

Set Form Builder

Default

NULL

Required/Optional Required only for a window that will be shown programmatically, rather than in response to navigation to an item on a canvas assigned to the window.

Primary Canvas restrictions

The specified view must be a content view (Canvas Type property set to Content), and must be assigned to the indicated window (by setting the Window canvas property).

Primary Key (Item) property

Description

Indicates that the item is a base table item in a data block and that it corresponds to a primary key column in the base table. Form Builder requires values in primary key items to be unique.

Applies to all items except buttons, chart items, and image items

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Required/Optional optional

Primary Key (Item) restrictions

The Enforce Primary Key block property must be set to Yes for the item's owning block.

Program Unit Text property

Description

Specifies the PL/SQL code that a program unit contains. When you click on More… in the Property Palette the Program Unit Editor is invoked.

Applies to program unit

Set Form Builder

Required/Optional required

Prompt property

Description

Specifies the text label that displays for an item.

Applies to item prompt

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

blank

Required/Optional optional

Prompt Alignment property

Description

Specifies how the prompt is aligned along the item's edge, either Start, End, or Center.

Applies to item prompt

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Alignment.)

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Start

Required/Optional required

Prompt Alignment Offset property

Description

Specifies the prompt's alignment offset.

Applies to item prompt

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Alignment_Offset.)

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

blank

Required/Optional optional

Prompt Attachment Edge property

Description

Specifies which edge the prompt should be attached to, either Start, End, Top, or Bottom.

Applies to item prompt

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Attachment_Edge.)

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Start

Required/Optional required

Prompt Attachment Offset property

Description

Specifies the distance between the item and its prompt.

Applies to item prompt

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Attachment_Offset.)

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

blank

Required/Optional optional

Prompt Background Color property

Description

The color of the object's or background region.

Applies to item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Background_Color.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Display Style property

Description

Specifies the prompt's display style.

First Record	Form Builder displays a prompt beside the first record.

- HiddeForm Builder does not display a prompt.

All Records	Form Builder displays a prompt beside each record.

Applies to item prompt

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Display_Style.)

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

First Record

Required/Optional required

Prompt Fill Pattern property

Description

Specifies the pattern to be used for the object’s fill region. Patterns are rendered in the two colors specified by Background_Color and Foreground_Color.

Applies to item, item prompt's, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Fill_Pattern.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Font Name property

Description

Specifies the font family, or typeface, to be used for text in the object. The list of fonts available is system-dependent.

Applies to item item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Font_Name.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Font Size property

Description

The size of the font, specified in points.

Applies to item, item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Font_Size.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Font Spacing property

Description

Specifies the width of the font (i.e., the amount of space between characters, or kerning).

Applies to item, item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Font_Spacing.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Font Style property

Description

Specifies the style of the font.

Applies to item, item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Font_Style.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Font Weight property

Description

Specifies the weight of the font.

Applies to item, item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Font_Weight.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Foreground Color property

Description

Specifies the color of the object's foreground region. For items, defines the color of the text displayed in the item.

Applies to item, item prompt, radio button

Set Form Builder, programmatically (Note: When you use this property in a PL/SQL program, replace the spaces with underscores: Prompt_Foreground_Color.)

Default

Unspecified

Required/Optional optional

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Prompt Justification property

Description

Specifies justification of the prompt as either Left, Right, Center, Start, or End.

Applies to item prompt

Set Form Builder

Default

Start

Required/Optional required

Prompt Reading Order property

Description

Specifies the prompt's reading order, either Default, Left to Right, or Right to Left.

Applies to item prompt

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Default

Required/Optional required

Prompt Visual Attribute Group property

Description

Specifies the named visual attribute that should be applied to the prompt at runtime.

Applies to item prompt

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Default

Required/Optional required

Prompt_White_On_Black property

Description

Specifies that the object is to appear on a monochrome bitmap display device as white text on a black background.

Applies to item, radio button

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Property Class property

Description

Specifies the name of the property class from which the object can inherit property settings.

Applies to all objects

Set Form Builder

Default

Null

Required/Optional optional

Query All Records property

Description

Specifies whether all the records matching the query criteria should be fetched into the data block when a query is executed.

Yes - Fetches all records from query; equivalent to executing the EXECUTE_QUERY (ALL_RECORDS) built-in.

No - Fetches the number of records specified by the Query Array Size block property.

Applies to block

Set Form Builder

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

No

Required/Optional Required if a data block contains summarized items, and the block's Precompute Summaries property is set to No.

Query Allowed (Block) property

Description

Specifies whether Form Builder should allow the end user or the application to execute a query in the block. When Query Allowed is No, Form Builder displays the following message if the end user attempts to query the block:

```
FRM-40360: Cannot query records here.
```

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

Yes

Restrictions:

When the Query Allowed block property is Yes, the Query Allowed item property must be set to Yes for at least one item in the block.

Query Allowed (Item) property

Description

Determines if the item can be included in a query against the base table of the owning block.

Applies to all items except buttons, chart items, and image items

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes; however if the item is part of the foreign key in the detail block of a master-detail block relation, Form Builder sets this property to No.

Usage Notes

To set the Query Allowed (Item) property programmatically, use the constant QUERYABLE.

Query Allowed (Item) restrictions

- The Visible property must also be set to Yes.
- Items with the data type LONG cannot be directly queried.

Query Array Size property

Description

Specifies the maximum number of records that Form Builder should fetch from the database at one time.

Applies to block

Set Form Builder

Refer to Built-in

GET_BLOCK_PROPERTY

Default

The number of records the block can display, as indicated by the Number of Records Displayed block property.

Required/Optional required

Usage Notes

A size of 1 provides the fastest perceived response time, because Form Builder fetches and displays only 1 record at a time. By contrast, a size of 10 fetches up to 10 records before displaying any of them, however, the larger size reduces overall processing time by making fewer calls to the database for records.

Query Array Size restrictions

- There is no maximum.

Query Data Source Arguments property

Description

Specifies the names, datatypes, and values of the arguments that are to be passed to the procedure for querying data. The Query Procedure Arguments property is valid only when the Query Data Source Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Query Data Source Columns property

Description

Specifies the names and datatypes of the columns associated with the block's query data source. The Query Data Source Columns property is valid only when the Query Data Source Type property is set to Table, Sub-query, or Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Query Data Source Name property

Description

Specifies the name of the block's query data source.

The Query Data Source Name property is valid only when the Query Data Source Type property is set to Table, Sub-Query, or Procedure.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_ITEM_PROPERTY

Default

NULL

Required/Optional optional

Query Data Source Name restrictions

Prior to setting the Query Data Source Name property you must perform a COMMIT_FORM or a CLEAR_FORM.

Query Data Source Type property

Description

Specifies the query data source type for the block. A query data source type can be a Table, Procedure, Transactional trigger, or FROM clause query.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

GET_BLOCK_PROPERTY

Default

Table

Required/Optional required

Query Length property

Description

Specifies the number of characters an end user is allowed to enter in the text item when the form is Enter Query mode.

Applies to text item

Set Form Builder

Default

The value of the item's Maximum Length property.

Usage Notes

You can make the query length greater than the Maximum Length when you want to allow the end user to enter complex query conditions. For example, a query length of 5 allows an end user to enter the query condition !=500 in a text item with a Maximum Length of 3.

Query Length restrictions

- The maximum query length is 255 characters.

Query Name property

Description

Specifies the name of the query in the report with which to associate the forms block.

Applies to report integration

Set Form Builder

Default

blank

Required/Optional optional

Query Only property

Description

Specifies that an item can be queried but that it should not be included in any INSERT or UPDATE statement that Form Builder issues for the block at runtime.

Applies to check box, radio group, list item, image item, text item, custom item (OLE)

Set programmatically

Default

No

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Query_Hits property

Description

Specifies the NUMBER value that indicates the number of records identified by the COUNT_QUERY operation. If this value is examined while records are being retrieved from a query, QUERY_HITS specifies the number of records that have been retrieved.

This property is included primarily for applications that will run against non-ORACLE data sources.

Applies to block

Set programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Usage Notes

This property can be used in several ways:

- In an application that runs against a non-ORACLE data source, use SET_BLOCK_PROPERTY(QUERY_HITS) in an On-Count trigger to inform Form Builder of the number of records that a query will return. This allows you to implement count query processing equivalent to Form Builder default Count Query processing.
- Use GET_BLOCK_PROPERTY(QUERY_HITS) during Count Query processing to examine the number of records a query will potentially retrieve.
- Use GET_BLOCK_PROPERTY(QUERY_HITS) during fetch processing to examine the number of records that have been retrieved by the query so far and placed on the block's list of records.

Query_Hits restrictions

Set this property greater than or equal to 0.

Query_Options property

Description

Specifies the type of query operation Form Builder would be doing by default if you had not circumvented default processing. This property is included for applications that will run against non-ORACLE data sources.

Values for this property include:

- VIEW
- FOR_UPDATE
- COUNT_QUERY
- NULL

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Radio Button Value Property

Description

Specifies the value associated with a radio button item in a radio group.

Applies to radio button

Set Form Builder

Default

blank

Raise on Entry property

Description

For a canvas that is displayed in the same window with one or more other canvases, Raise on Entry specifies how Form Builder should display the canvas when the end user or the application navigates to an item on the canvas.

- When Raise on Entry is No, Form Builder raises the view in front of all other views in the window *only* if the target item is behind another view.
- When Raise on Entry is Yes, Form Builder *always* raises the view to the front of the window when the end user or the application navigates to *any* item on the view.

Applies to canvas

Set Form Builder

Default

No

Raise on Entry restrictions

Applicable only when more than one canvas is assigned to the same window.

Reading Order property

Description

Note: This property is specific to bidirectional National Language Support (NLS) applications.

Specifies the reading order for groups of words (segments) in the same language within a single text item.

Reading Order allows you to control the display of bilingual text items, text items that include segments in both Roman and Local languages. (The Reading Order property has no effect on text items composed of a single language.)

The allowable values for this property are:

Value	*Description*
Default	Text item inherits the reading order specified by its canvas Language Direction property setting.
Right-To-Left	Item reading order is right-to-left.
Left-To-Right	Item reading order is left-to-right.

Applies to display item, text item

Set Form Builder

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Default

Usage Notes

- In most cases, you will not need to explicitly set the Reading Order property (the Default setting will provide the functionality you need). Use the Reading Order property only when you need to override the default reading order for an item.
- To get or set the Reading Order property programmatically, use the Language Direction property.
- To display a Local segment in Right-To-Left mode and a Roman segment in Left-To-Right, use the Default value.
- If your item text is mostly Local, choose the Right-To-Left value.
- If your item text is mostly Roman, choose the Left-To-Right value.

Real Unit property

Description

When the Coordinate System property is set to Real, the Real Unit property specifies the real units to be used for specifying size and position coordinates in the form. Real units can be centimeters, inches, pixels, points, or decipoints. (A point is 1/72nd of an inch.)

Form Builder interprets all size and position coordinates specified in the form in the real units you specify here. When you convert from one real unit to another, some loss of precision may occur for existing object size and position values.

Applies to form module

Set Form Builder

Default

Centimeter

Required/Optional optional

Real Unit restrictions

Valid only when the coordinate system property is set to Real.

Record Group property

Description

Specifies the name of the record group from which the LOV or hierarchical tree derives its values.

Applies to:

LOV, hierarchical tree

Set Form Builder, programmatically

Refer to Built-in

GET_LOV_PROPERTY (GROUP_NAME)

SET_LOV_PROPERTY (GROUP_NAME)

POPULATE_TREE

POPULATE_GROUP_FROM_TREE

Default

Null

Required/Optional Required for LOV, Optional for hierarchical tree

Usage Notes

An LOV displays the records stored in its underlying record group. Each LOV must be based on a record group. A record group can be populated by a query (query record group) or by fixed values (static record group).

Record Group Fetch Size property

Description

Specifies the size of the record group to be fetched. A larger fetch size reduces the number of fetches required to obtain the record group. For example, a record group of 5000 records will require 500 trips to be fetched if Record Group Fetch Size is set to 10, but only 5 trips if Record Group Fetch Size is set to 1000.

Applies to record group functional

Set Form Builder

Default

20

Required/Optional required

Usage Notes

Only available when Record Group Type is set to Query.

Record Group Query property

Description

Specifies the SELECT statement for query associated with the record group.

Applies to record group

Set Form Builder, programmatically

Refer to Built-in

POPULATE_GROUP_WITH_QUERY

Required/Optional optional

Record Group Type property

Description

Specifies the type of record group, either Static or Query:

Static	Specifies that the record group is constructed of explicitly defined column names and column values. The values of a static record group are specified at design time and cannot be changed at runtime.
Query	Specifies that the record group is associated with a SELECT statement, and thus can be populated dynamically at runtime. When you select this option, enter the SELECT statement in the multi-line field provided, then choose Apply.

Applies to:

record group

Set Form Builder

Default

Query

Record Orientation property

Description

Determines the orientation of records in the block, either horizontal records or vertical records. When you set this property, Form Builder adjusts the display position of items in the block accordingly.

Applies to block

Set Form Builder

Default

Vertical records

Required/Optional optional

Usage Notes

You can also set this property when you create a block in the New Block window by setting the Orientation option to either Vertical or Horizontal.

Record Orientation restrictions

Valid only for a multi-record block (Number of Records Displayed property set greater than 1).

Records_to_Fetch property

Description

Returns the number of records Form Builder expects an On-Fetch trigger to fetch and create as queried records.

You can programmatically examine the value of Records_To_Fetch when you are using transactional triggers to replace default Form Builder transaction processing when running against a non-ORACLE data source.

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Usage Notes

Records_To_Fetch is defined only within the scope of an On-Fetch trigger.

The first time the On-Fetch trigger fires, the value of Records_To_Fetch is either the array size (as specified by the Query Array Size block property) or the number of records displayed + 1, whichever is larger.

If the On-Fetch trigger creates this many queried records, the next time the On-Fetch trigger fires, the value of Records_To_Fetch will be the same number.

If, however, the On-Fetch trigger creates fewer records than the value of Records_To_Fetch and returns without raising Form_trigger_Failure, Form Builder will fire the On-Fetch trigger again. Records_To_Fetch will be the set to its previous value minus the number of queried records created by the previous firing of the On-Fetch trigger.

This behavior continues until one of the following events occurs:

- The trigger does not create a single queried record (signaling a successful end of fetch).
- The expected number of queried records gets created.
- The trigger raises a Form_trigger_Failure (signaling that the fetch aborted with an error and fetch processing should halt).

Records_to_Fetch examples

Example

```
/*
** Call a client-side package function to retrieve
** the proper number of rows from a package cursor.
*/
DECLARE
   j NUMBER := Get_Block_Property(blk_name, RECORDS_TO_FETCH);
```

```
   emprow emp%ROWTYPE;
BEGIN
   FOR ctr IN 1..j LOOP
      /* Try to get the next row.*/
      EXIT WHEN NOT MyPackage.Get_Next_Row(emprow);
      Create_Queried_Record;
      :Emp.rowid := emprow.ROWID;
      :Emp.empno := emprow.EMPNO;
      :Emp.ename := emprow.ENAME;
         :
         :
   END LOOP;
END;
```

Relation Type property

Description

Specifies whether the link between the master block and detail block is a relational join or an object REF pointer.

Applies to master-detail relations

Set Form Builder

Default

Join

Usage Notes

Valid values are Join (indicating a relational join) or REF (indicating a REF column in one block pointing to referenced data in the other block).

When the link is via a REF, see also the Detail Reference property.

When the link is via a join, see also the Join Condition property.

Rendered property

Description

Specifies that the item is to be displayed as a rendered object when it does not have focus.

Applies to text item, display item

Set Form Builder

Default

Yes

Usage Notes

Use the Rendered property to conserve system resources. A rendered item does not require system resources until it receives focus. When a rendered item no longer has focus, the resources required to display it are released.

Report Destination Format property

Description

In bit-mapped environments, this property specifies the printer driver to be used when the Report Destination Type property is File. In character-mode environments, it specifies the characteristics of the printer named in report Destination name property.

Possible valus are any valid destination format not to exceed 1K in length. Examples of valid values for this keyword are hpl, hplwide, dec, decwide, decland, dec180, dflt, wide, etc. Ask your System Administrator for a list of valid destination formats. In addition, Report Builder supports the following destination formats:

PDF	Means that the report output will be sent to a file that can be read by a PDF viewer. PDF output is based upon the currently configured printer for your system. The drivers for the currently selected printer is used to produce the output; you must have a printer configured for the machine on which you are running the report.
HTML	Means that the report output will be sent to a file that can be read by an HTML 3.0 compliant browser (e.g., Netscape 2.2).
HTMLCSS	Means that the report output sent to a file will include style sheet extensions that can be read by an HTML 3.0 compliant browser that supports cascading style sheets.
HTMLCSSIE	Means that the report output sent to a file will include style sheet extensions that can be read by Microsoft Internet Explorer 3.x.
RTF	Means that the report output will be sent to a file that can be read by standard word processors (such as Microsoft Word). Note that when you open the file in MS Word, you must choose **View->Page Layout** to view all the graphics and objects in your report.
DELIMITED	Means that the report output will be sent to a file that can be read by standard spreadsheet utilities, such as Microsoft Excel. Note that you must also specify a DELIMITER.

For more information about this property, see DESFORMAT under the index category Command Line Arguments in the Report Builder online help.

Applies to report reports

Set Form Builder

Default

blank

Required/Optional optional

Report Destination Name property

Description

Name of the file, printer, Interoffice directory, or email user ID (or distribution list) to which the report output will be sent. Possible values are any of the following not to exceed 1K in length:

- a filename (if Report Destination Type property is File or Localfile)
- a printer name (if Report Destination Type property is Printer)
- email name or distribution name list (if Report Destination Type property is Mail).
  To send the report output via email, specify the email ID as you do in your email application (any MAPI-compliant application on Windows, such as Oracle InterOffice, or your native mail application on UNIX). You can specify multiple usernames by enclosing the names in parentheses and separating them by commas (e.g., (name, name, . . .name)). For printer names, you can optionally specify a port. For example:

  ```
  printer,LPT1:
  printer,FILE:
  ```

Or, if the Report Destination Type property is Interoffice:

```
/FOLDERS/directory/reportname
```

For more information about this property, see DESNAME under the index category Command Line Arguments in the Report Builder online help.

Applies to report reports

Set Form Builder

Default

blank

Required/Optional optional

Report Destination Type property

Description

Destination to which you want the output to be sent. Possible values are Screen, File, Printer, Preview, Mail, and Interoffice. For more information about this property, see DESTYPE under the index category Command Line Arguments in the Report Builder online help.

SCREEN	Screen routes the output to the Previewer for interactive viewing. This value is valid only for when running the report in Runtime mode (not Batch). Font aliasing is not performed.
FILE	File saves the output to a file named in Report Destination Name.
PRINTER	Printer routes the output to the printer named in Report Destination Name.
PREVIEW	Preview routes the output to the Previewer for interactive viewing. However, Preview causes the output to be formatted as Postscript output. The Previewer will use the Report Destination Name property to determine which printer's fonts to use to display the output. Font aliasing is performed for Preview.
MAIL	Mail routes the output to the mail users specified in Report Destination Name. You can send mail to any mail system that is MAPI-compliant or has the service provider driver installed. The report is sent as an attached file.
INTEROFFICE	routes the output to the Oracle InterOffice mail users specified in Report Destination Name. By specifying this value, you store the report output in InterOffice as a repository.

Applies to report reports

Set Form Builder

Default

File

Required/Optional required

Report Server property

Description

Specifies the Report Server against which you can run your Report.

Applies to report reports

Set Form Builder

Required/Optional optional

Default

blank

Required (Item) property

Description

When a new record is being entered, specifies that the item is invalid when its value is NULL.

Applies to list item, text item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_INSTANCE_PROPERTY
- GET_ITEM_PROPERTY
- SET_ITEM_INSTANCE_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Usage Notes

When an item has Required set to Yes, and item-level validation is in effect, by default Form Builder will not allow navigation out of the item until a valid value is entered. To allow the end user to move freely among the items in the record, set the Defer Required Enforcement property to Yes. This will postpone enforcement of the Required attribute from item validation to record validation.

Even when Required is set to Yes, there are circumstances when an item's value could be NULL. Form Builder checks for required items as part of its validation process: each item in a new record is subject to validation, but queried data is presumed to be valid and an item is not validated unless it is changed. For example, if the record already exists and is queried from the database, the item that would be Required could come in as NULL.

Setting a poplist's or T-list's Required property may affect the values the list will display: When selected, an instance of a poplist will display an extra null value if its current value is NULL or if its effective Required property is No (false). When selecting the current value of an instance of a T-list, it will be unselected (leaving the T-list with no selected value) if its effective Required property is No (false). But if its effective Required property is Yes (true), selecting a T-list instance's current value will have no effect. The value will stay selected.

Required (Menu Parameter) property

Description

Specifies that the end user is required to enter a value for the menu substitution parameter.

Applies to menu substitution parameter

Set Form Builder

Default

No

Resize Allowed property

Description

Specifies that the window is to be a fixed size and cannot be resized at runtime. This property is a GUI hint, and may not be supported on all platforms.

Applies to window

Set Form Builder

Default

No

Usage Notes

The Resize Allowed property prevents an end user from resizing the window, but it does not prevent you from resizing the window programmatically with RESIZE_WINDOW or SET_WINDOW_PROPERTY.

Resize Allowed restrictions

- Resize Allowed is only valid when the Maximize Allowed property is set to No

Return Item (LOV) property

Description

See Column Mapping Properties .

Rotation Angle property

Description

Specifies the graphic object's rotation angle. The angle at which the object is initially created is considered to be 0, and this property is the number of degrees clockwise the object currently differs from that initial angle. You can rotate an object to an absolute angle by setting this property.

Applies to graphics physical

Set Form Builder

Default 0

Required/Optional required

Runtime Compatibility Mode property

Description

Specifies the Form Builder version with which the current form's runtime behavior is compatible (either 4.5 or 5.0 +). By default, new forms created with Form Builder 5.0 and later are set to 5.0-compatible. Existing forms that are upgraded from 4.5 are 4.5-compatible. To get these forms to use the new runtime behavior of 5.0 +, set this property to 5.0. The runtime behavior that is affected by this property is primarily validation and initialization. For information about 5.0-and-later runtime behavior, see the Initialization and Validation sections in the Default Processing chapter of the online Form Builder Reference. For information about 4.5 runtime behavior, see the Form Builder 4.5 Runtime Behavior section in the Compatibility with Prior Releases chapter of the online Form Builder Reference.

Applies to forms compatibility

Set Form Builder

Default

5.0 for new forms, 4.5 for forms created using Form Builder 4.5.

Required/Optional required

Savepoint Mode property

Description

Specifies whether Form Builder should issue savepoints during a session. This property is included primarily for applications that will run against non-ORACLE data sources. For applications that will run against ORACLE, use the default setting.

The following table describes the settings for this property:

Setting	*Description*
Yes (the default)	Specifies that Form Builder should issue a savepoint at form startup and at the start of each Post and Commit process.
No	Specifies that Form Builder is to issue no savepoints, and that no rollbacks to savepoints are to be performed.

Applies to form module

Set Form Builder, programmatically

Refer to Built-in

- GET_FORM_PROPERTY
- SET_FORM_PROPERTY

Default

Yes

Required/Optional optional

Savepoint Mode restrictions

When Savepoint Mode is No, Form Builder does not allow a form that has uncommitted changes to invoke another form with the CALL_FORM procedure.

Savepoint_Name property

Description

Specifies the name of the savepoint Form Builder is expecting to be set or rolled back to.

Applies to application

Set not settable

Refer to Built-in

GET_APPLICATION_PROPERTY

Usage Notes

The value of this property should be examined only within an On-Savepoint or On-Rollback trigger:

- Use Savepoint_Name in an On-Savepoint trigger to determine the savepoint to be set by a call to ISSUE_SAVEPOINT.
- In an On-Rollback trigger, examine Savepoint_Name to determine the savepoint to which Form Builder should roll back by way of a call to ISSUE_ROLLBACK. A NULL savepoint name implies that a full rollback is expected.

Scroll Bar Alignment property

Description

Specifies whether the scroll bar is displayed at the start or the end of the frame.

Applies to frame

Set Form Builder

Default

End

Required/Optional optional

Scroll Bar Height property

Description

Specifies the height of the scroll bar.

Applies to scroll bar

Set Form Builder

Required/Optional optional

Scroll Bar Width property

Description

Specifies the width of the scroll bar.

Applies to scroll bar

Set Form Builder

Required/Optional optional

Secure (Menu Parameter) property

Description

Hides characters that the end user enters for the substitution parameter.

Applies to menu substitution parameter

Set Form Builder

Default

No

Required/Optional optional

Share Library with Form property

Description

Forms that have identical libraries attached can share library package data. (For more information, see the *data_mode* parameter for the CALL_FORM, OPEN_FORM, and NEW_FORM built-ins.) The Share Library with Form property enables menus associated with the forms to share the library package data as well.

Applies to menus

Set Form Builder

Default

Yes

Usage Notes

- If two forms share an object, and both forms are open at design time and you make changes to the object in Form A, those changes will not be seen in Form B until the changes are first saved by Form A, and Form B is then closed and reopened.
- If you use OPEN_FORM to open a form in a different database session, you cannot share library data with the form or its associated menus. Attempts to share library data by setting the property to Yes will be ignored.

Show Fast Forward Button property

Description

Determines whether the sound item control will display the fast forward button (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Required/Optional Required

Show Horizontal Scroll Bar property

Description

Determines whether a canvas, secondary window, or image item is displayed with a scroll bar.

Applies to canvas, window, editor, image item

Set Form Builder

Default

No

Required/Optional optional

Show Horizontal Scroll Bar restrictions

- For a window, only valid when the Modal property is set to No.
- Valid on window managers that support horizontal scroll bars.

Show Lines property

Description

Determines whether a hierarchical tree displays lines leading up to each node.

Applies to hierarchical tree

Set Form Builder

Default

True

Required/Optional required

Show OLE Popup Menu property

Description

Determines whether the right mouse button displays a popup menu of commands for interacting with the OLE object. The following settings are valid for this property:

YES	The default OLE popup menu selection. The OLE popup menu is displayed when the mouse cursor is on the OLE object and the right mouse button is pressed.
NO	The OLE popup menu is not displayed when mouse cursor is on the OLE object and the right mouse button is pressed.

Applies to OLE Container

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Required/Optional required

Usage Notes

- In addition to the Form Builder, you can programmatically set and get the OLE popup menu value by using the SHOW_POPUPMENU property. For the SET_ITEM_PROPERTY built-in, the OLE popup menu is shown when the SHOW_POPUPMENU property is set to PROPERTY_TRUE. When the SHOW_POPUPMENU property is set to PROPERTY_FALSE, the OLE popup menu is not shown. You can also use the SHOW_POPUPMENU property with the GET_ITEM_PROPERTY built-in to obtain the current OLE popup menu setting. The GET_ITEM_PROPERTY built-in returns TRUE when the OLE popup menu is shown, and GET_ITEM_PROPERTY returns FALSE when the OLE popup menu is not shown.
- Valid only on Microsoft Windows and Macintosh.

Show OLE Tenant Type property

Description

Determines whether a border defining the OLE object type surrounds the OLE container. The type of border varies according to the object type.

Applies to OLE Container

Set Form Builder

Default

Yes

Show OLE Tenant Type restrictions

Valid only on Microsoft Windows and Macintosh.

Show Palette property

Description

Determines whether Form Builder will display an image-manipulation palette adjacent to the associated image item at runtime. The palette provides three tools that enable end users to manipulate a displayed image:

- **Zoom**—click the tool, then repeatedly click the image to incrementally reduce the amount of the source image displayed within the image item's borders.
- **Pan**—click the tool and use the grab hand to pan unseen portions of the source image into view (valid only if the source image extends beyond at least one border of the image item).
- **Rotate**—click the tool, then repeatedly click the image to rotate it clockwise in 90-degree increments.

Applies to image item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Required/Optional Required

Show Play Button property

Description

Determines whether the sound item control will display the play button (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Required/Optional Required

Usage Notes

- If you set the Show Play Button property to No when the Show Record Button property already is set to No, Form Builder automatically will display at runtime.

Show Record Button property

Description

Determines whether the sound item control will display the record button (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Required/Optional Required

Usage Notes

- If you set the Show Record Button property to No when the Show Play Button property already is set to No, Form Builder will automatically display at runtime.

Show Rewind Button property

Description

Determines whether the sound item control will display the rewind button (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Required/Optional Required

Show Scroll Bar property

Description

The Show Scroll Bar option specifies whether Form Builder should create a block scroll bar for the block you are defining. When Show Scroll Bar is set to Yes, Form Builder creates the scroll bar on the canvas specified by the Scroll Bar Canvas property.

When you create a block scroll bar, you can set the properties of the scroll bar object itself, including Scroll Bar Canvas, Scroll Bar Orientation, Scroll Bar X Position, Scroll Bar Y Position, Scroll Bar Width, Scroll Bar Height, Reverse Direction, and Visual Attribute Group.

Applies to block

Set Form Builder

Default:

No

Required/Optional optional

Usage Notes

Setting Reverse Direction to Yes causes Form Builder to fetch the next set of records when the end user scrolls upward. If the end user scrolls downward, Form Builder displays already fetched records.

Property	Description
Scroll Bar Canvas	Specifies the canvas on which the block's scroll bar should be displayed. The specified canvas must exist in the form.
Scroll Bar Orientation	Specifies whether the block scroll bar should be displayed horizontally or vertically.
Scroll Bar X Position	Specifies the x position of a block scroll bar measured at the upper left corner of the scrollbar. The default value is 0.
Scroll Bar Y Position	Specifies the width of a block scroll bar measured at the upper left corner of the scrollbar. The default value is 0.
Scroll Bar Width	Specifies the width of a block scroll bar. The default value is 2.
Scroll Bar Height	Specifies the height of a block scroll bar. The default value is 10.
Reverse Direction	Specifies that the scroll bar scrolls in reverse. The default value is No.

Visual Attribute Group	Specifies the font, color, and pattern attributes to use for scroll bar. Refer to the Visual Attribute Group property for more information. The default setting is determined by the platform and resource file definition.

Show Slider property

Description

Determines whether the sound item control will display the Slider position control (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Required/Optional Required

Show Symbols property

Description

Indicates whether a hierarchical tree should display + or - symbols in front of each branch node. The + symbol indicates that the node has children but is not expanded. The - symbol indicates that the node is expanded.

Applies to hierarchical tree

Set Form Builder

Default

True

Required/Optional required

Show Time Indicator property

Description

Determines whether the sound item control displays the time indicator (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Required/Optional Required

Show Vertical Scroll Bar property

Description

Specifies that a vertical scroll bar is to appear on the side of a canvas or window.

Applies to canvas, window, image item, editor, item

Set Form Builder

Default

No

Required/Optional Optional

Show Vertical Scroll Bar restrictions

- Valid on window managers that support vertical scroll bars.
- Not valid for a root window: a root window cannot have scroll bars.
- Valid on window managers that support vertical scroll bars.
- For text item, the Multi-Line property must be YES.

Show Volume Control property

Description

Determines whether the sound item control will display the volume control (both in the Layout Editor and at runtime).

Applies to Sound item control

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Required/Optional Required

Shrinkwrap property

Description

Specifies whether blank space should be automatically removed from the frame. When Shrinkwrap is set to Yes, Form Builder automatically reduces, or "shrinkwraps", the frame around the items within the frame.

Note: Reisizing a frame has no effect when Shrinkwrap is set to Yes; when you, for example, increase the size of the frame, Form Builder automatically reduces the frame to its shrinwrap size. If you want to resize a frame, set Shrinkwrap to No.

Applies to frame

Set Form Builder

Default

Yes

Required/Optional Optional

Single Object Alignment property

Description

Specifies the alignment for single line objects when the Frame Alignment property is set to Fill.

Applies to frame

Set Form Builder

Default

Start

Required/Optional Required

Single Record property

Description

Specifies that the control block always should *contain* one record. Note that this differs from the number of records *displayed* in a block.

Applies to block

Set Form Builder

Default

No

Usage Notes

- Set Single Record to Yes for a control block that contains a summary calculated item, a VBX (on Microsoft Windows 3.x 16-bit) or ActiveX control (on 32-bit Windows). Conversely, Single Record must be set to No for the block that contains the item whose values are being summarized and displayed in the calculated item.
- You cannot set Single Record to Yes for a data block.

Size property

Description

Specifies the width and height of the canvas in the current form coordinate units specified by the Coordinate System form property.

Applies to canvas

Set Form Builder, programmatically

Refer to Built-in

- GET_CANVAS_PROPERTY
- SET_CANVAS_PROPERTY

Size (Item)

Specifies the width and height of the item in the current form coordinate units specified by the Coordinate System form property.

Applies to item

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Size (Editor)

Specifies the width and height of the editor in the current form coordinate units specified by the Coordinate System form property.

Applies to editor

Set Form Builder, programmatically

Refer to Built-in

- EDIT_TEXTITEM
- SHOW_EDITOR

Usage Notes

- For a text item or display item, the number of characters the item can store is determined by the Max Length property, and is not affected by the size property.

- In applications that will run on character mode platforms, the height of items that display text labels must be at least 2 character cells for the text to display properly.

Size (LOV)

Specifies the width and height of the LOV in the current form coordinate units specified by the Coordinate System form property.

Specifies the width and height of the LOV, in the current form coordinate units specified by the Coordinate System form property.

Applies to LOV

Set Form Builder, programmatically

Refer to Built-in

- GET_LOV_PROPERTY
- SET_LOV_PROPERTY

Restrictions

Form Builder will ensure that the minimum width of the LOV is set wide enough to display the buttons at the bottom of the LOV. (On platforms that allow LOVs to be resized, you can resize the LOV to a minimum that will not display all the buttons.)

Size (Window)

Specifies the width and height of the window in the current form coordinate units specified by the Coordinate System form property.

Applies to window

Set Form Builder, programmatically

Default

80 characters by 24 characters

Refer to Built-in

- GET_WINDOW_PROPERTY
- SET_WINDOW_PROPERTY

Size restrictions

Form Builder will ensure that the minimum width of the editor is set wide enough to display the buttons at the bottom of the editor. (On platforms that allow editors to be resized, you can resize the editor to a minimum that will not display all the buttons.)

Sizing Style property

Description

Determines the display style of an image when the image size does not match the size of the image item.

The following settings are valid for this property:

Crop	Displays only the portion of the full image that fits in the display rectangle.
Adjust	Scales the image to fit within the display rectangle.

Applies to image item

Set Form Builder

Default

Crop

Sound Format property

Description

Specifies the format in which the sound item will be stored in the database: either AU, AIFF, AIFF-C, or WAVE.

When you use the READ_SOUND_FILE or WRITE_SOUND_FILE built-in subprograms to work with sound data read from—or written to—the filesystem, use the *file_type* parameter to control the sound format of the sound data being read or written.

Applies to sound item

Set Form Builder

Refer to Built-in

- READ_SOUND_FILE
- WRITE_SOUND_FILE

Default

WAVE

Required/Optional required

Sound Quality property

Description

Specifies the quality with which the sound item will be stored in the database: either Automatic, Highest, High, Medium, Low, or Lowest.

When you use the WRITE_SOUND_FILE built-in subprogram to write sound data to the filesystem, use the *sound_quality* parameter to control the sound format of the sound data being written.

Applies to sound item

Set Form Builder

Refer to Built-in

- WRITE_SOUND_FILE

Default

Automatic

Required/Optional required

Start Angle property

Description

Specifies the starting angle of the arc, using the horizontal axis as an origin.

Applies to graphic arc

Set Form Builder

Default

90

Required/Optional required

Start Prompt Alignment property

Description

Specifies how the prompt is aligned to the item's horizontal edge, either Start, Center, or End. This property is valid when the Layout Style property is set to Form.

Applies to frame

Set Form Builder

Default

Start

Required/Optional required

Start Prompt Offset property

Description

Specifies the distance between the prompt and its item when the Start Prompt Alignment property is set to Start.

Applies to frame

Set Form Builder

Default

0 (character cell)

Required/Optional required

Startup Code property

Description

Specifies optional PL/SQL code that Form Builder executes when the menu module is loaded in memory at form startup. Think of startup code as a trigger that fires when the menu module is loaded.

Applies to menu module

Set Form Builder

Required/Optional optional

Usage Notes

Startup code does not execute when Form Builder is returning from a called form.

Status (Block) property

Description

Specifies the current status of an indicated block. Block status can be New, Changed, or Query.

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Usage Notes

You can determine the status of the *current* block in the form by examining the SYSTEM.BLOCK_STATUS system variable. Form status can be examined by way of the SYSTEM.FORM_STATUS system variable.

Status (Record) property

Description

Specifies the current status of the indicated record. Record status can be New, Changed, Query, or Insert.

Applies to record

Set programmatically

Refer to Built-in

- GET_RECORD_PROPERTY
- SET_RECORD_PROPERTY

Usage Notes

The status property allows you to examine the status of any indicated record. You can also examine the status of the *current* record in the form with the SYSTEM.RECORD_STATUS system variable.

In general, any assignment to a database item will change a record's status from QUERY to CHANGED (or from NEW to INSERT), even if the value being assigned is the same as the previous value. Passing an item to a procedure as OUT or IN OUT parameter counts as an assignment to it.

Subclass Information property

Description

Specifies the following information about the source object and source module for a referenced objects.

Module	The name of the source module.
Storage	The source module type (Form or Menu) and location (File System or Database)
Name	The name of the source object in the source module. (The name of a reference object can be different than the name of its source object.)

Applies to any reference object

Set Form Builder

Required/Optional optional

Submenu Name property

Description

Specifies the name of the submenu associated with a main menu. The Command Type property must be set to Menu to invoke the Submenu property.

Applies to menu items

Set Form Builder

Required/Optional required

Default

Null

Summarized Block property

Description

Specifies a Form Builder block, over which all rows of a summary calculated item is summarized (e.g., summed, averaged, etc.) in order to assign a value to the item.

Applies to block

Set Form Builder

Required/Optional required if the associated item's Calculation Mode property is set to Summary

Summarized Item property

Description

Specifies a Form Builder item, the value of which is summarized (e.g., summed, averaged, etc.) in order to assign a value to a summary calculated item.

Applies to item

Set Form Builder

Required/Optional required if the associated item's Calculation Mode property is set to Summary

Summarized Item restrictions

- The summarized item cannot be a summary item.
- If the summarized item does not reside in the same block as the summary item, the summary item must reside in a control block with the Single Record property set to Yes.

Summary Function property

Description

Specifies the type of computation function Form Builder will perform on the summarized item.

Avg	The average value (arithmetic mean) of the summarized item over all records in the block.
Count	Count of all non-null instances of the summarized item over all records in the block.
Max	Maximum value of the summarized item over all records in the block.

MiMinimum value of the summarized item over all records in the block.

Stddev	The standard deviation of the summarized item's values over all records in the block.
Sum	Sum of all values of the summarized item over all records in the block.
Variance	The variance of the summarized item's values over all records in the block. (Variance is defined as the square of the standard deviation.)

Note: For more information about these arithmetic operations, refer to the *Oracle8 Server SQL Language Reference Manual.*

Applies to item

Set Form Builder

Required/Optional required (only if associated item's Calculation Mode property is set to Summary)

Default

None

Summary Function restrictions

You must set the Parameter Data Type property to Number, unless the item's Summary Type is Max or Min, in which case the datatype must mirror that of its associated summarized item. For example, a calculated item that displays the most recent (i.e., maximum) date in the HIRE_DATE column must have a datatype of Date.

Synchronize with Item property

Description

Specifies the name of the item from which the current item should derive its value. Setting this property synchronizes the values of the two items, so that they effectively mirror each other. When the end user or the application changes the value of either item, the value of the other item changes also.

Applies to all items except OLE containers

Set Form Builder

Required/Optional Optional

Default

NULL

Usage Notes

- In earlier releases, this property was called the Mirror Item property.
- You can set Synchronize with Item for base table or control blocks. When Synchronize with Item is specified, the current item's Base Table Item property is ignored, and the item derives its value from the mirror item specified, rather than from a column in the database.
- If you use the GET_ITEM_PROPERTY built-in to obtain a Base Table Item property, it will obtain the value from the mirror item specified.
- You can use mirror item to create more than one item in a block that display the same database column value.

Synchronize with Item restrictions

- The maximum number of items in a form that can point to the same mirror item is 100.

Tab Attachment Edge property

Description

Specifies the location where tabs are attached to a tab canvas.

Applies to tab canvas

Set Form Builder

Default

Top

Required/Optional required

Tab Attachment Edge restrictions

Valid only for tab canvas.

Tab Page property

Description

The name of the tab page on which the item is located.

Applies to item

Set Form Builder

Default

none

Refer to Built-in

- GET_ITEM_PROPERTY (programmatic property name is Item_Tab_Page)

Required/Optional required (if the item is located on a tab canvas)

Tab Page X Offset property

Description

The distance between the left edge of the tab canvas and the left edge of the tab page. The value returned depends on the form coordinate system—pixel, centimeter, inch, or point.

Applies to tab canvas

Refer to Built-in

- GET_CANVAS_PROPERTY

Tab Page X Offset restrictions

- you can *get* the property value, but you cannot *set* it
- valid only for tab canvas. 0 is returned for all other canvas types

Tab Page Y Offset property

Description

Specifies the distance between the top edge of the tab canvas and the top edge of the tab page. The value returned depends on the form coordinate system used -- pixel, centimeter, inch, or point.

Applies to tab canvas

Refer to Built-in

- GET_CANVAS_PROPERTY

Tab Page Y Offset restrictions

- you can *get* the property value, but you cannot *set* it
- valid only for tab canvas. 0 is returned for all other canvas types

Tab Style property

Description

Specifies the shape of the labelled tab(s) on a tab canvas.

Applies to tab canvas

Set Form Builder

Default

Chamfered

Required/Optional required

Tear-Off Menu property

Description

Defines a menu as a tear-off menu.

Applies to menu

Set Form Builder

Default

No

Tear-Off Menu restrictions

Only supported in the pull-down menu style, on window managers that support this feature.

Timer_Name property

Description

Specifies the name of the most recently expired timer.

Applies to application

Set not settable

Refer to Built-in

GET_APPLICATION_PROPERTY

Required/Optional optional

Timer_Name restrictions

Only valid when examined in a When-Timer-Expired trigger.

Title property

Description

Specifies the title to be displayed for the object.

Applies to alert, form module, LOV, window

Set Form Builder

Required/Optional optional

Title (LOV)

Default

NULL

Required/Optional optional

Title (Window)

Refer to Built-in

GET_WINDOW_PROPERTY

- SET_WINDOW_PROPERTY

Required/Optional optional

Usage Notes

- Length limit of a window title depends on the display driver used. (For example, for an SVGA 1280 x 1024, running under NT, the limit is 78 characters.)
- If you do not specify a title for a window that is not a root window, Form Builder uses the window's object name, as indicated by the window Name property.
- If you do not specify a title for a root window, and the current menu is the Default menu, Form Builder uses the name of the form module for the root window title, as indicated by the form module Name property. When the current menu is a custom menu running in Pull-down or Bar display style, Form Builder uses the name of the main menu in the module for the root window title, as indicated by the menu module Main property.

Tooltip property

Description

Specifies the help text that should appear in a small box beneath the item when the mouse enters the item.

Applies to item

Set Form Builder, programmtically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

blank

Required/Optional optional

Tooltip Background Color property

Description

Specifies the color of the object’s background region.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Fill Pattern property

Description

Specifies the pattern to be used for the object's fill region. Patterns are rendered in the two colors specified by Background_Color and Foreground_Color.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Font Name property

Description

Specifies the font family, or typeface, to be used for text in the object. The list of fonts available is system-dependent.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Font Size property

Description

Specifes the size of the font in points.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Font Spacing property

Description

Specifies the width of the font (i.e., the amount of space between characters, or kerning).

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Font Style property

Description

Specifies the style of the font.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Font Weight property

Description

Specifies the weight of the font.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Foreground Color property

Description

Specifies the color of the object’s foreground region. For items, defines the color of the text displayed in the item.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Tooltip Visual Attribute Group property

Description

Specifies the named visual attribute that should be applied to the tooltip at runtime.

Applies to item tooltip

Set Form Builder

Default

Default

Required/Optional required

Tooltip White on Black property

Description

Specifies that the object is to appear on a monochrome bitmap display device as white text on a black background.

Applies to item

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Top Prompt Alignment property

Description

Specifies how the prompt is aligned to the item's top edge, either Start, End, or Center. This property is valid when the Layout Style property is set to Tabular.

Applies to frame

Set Form Builder

Default

Start

Required/Optional required

Top Prompt Offset property

Description

Specifies the distance between the prompt and its item when the Top Prompt Alignment property is set to Top.

Applies to frame

Set Form Builder

Default

0 (character cell)

Required/Optional required

Top_Record property

Description

Specifies the record number of the topmost record that is visible in the block. (Records are numbered in the order they appear on the block's internal list of records.)

Applies to block

Set not settable

Refer to Built-in

GET_BLOCK_PROPERTY

Usage Notes

Together, the TOP_RECORD and RECORDS_DISPLAYED properties allow you to determine the number of the bottom record in the display, that is, the record having the highest record number among records that are currently displayed in the block.

Top Title property

Description

Specifies a title of up to 72 characters to appear at the top of the editor window.

Applies to editor

Set Form Builder

Required/Optional optional

Topmost_Tab_Page property

Description

Specifies the top most tab page in a tab canvas.

Applies to tab canvas

Set Form Builder, programmatically

Refer to Built-in

- GET_CANVAS_PROPERTY
- SET_CANVAS_PROPERTY

Default

First tab page that appears under the Tab Page node.

Topmost_Tab_Page restrictions

Valid only for tab canvas.

Transactional Triggers property

Description

Identifies a block as a *transactional control block*; that is, a non-database block that Form Builder should manage as a transactional block. This property is included for applications that will run against non-ORACLE data sources, and that will include transactional triggers. If your application will run against ORACLE, leave this property set to No.

When you create a non-ORACLE data source application, you are essentially simulating the functionality of a data block by creating a transactional control block. Such a block is a control block because its base table is not specified at design time (the Base Table block property is NULL), but it is transactional because there are transactional triggers present that cause it to function as if it were a data block.

For more information, see *Form Builder Advanced Techniques*, Chapter 4, "Connecting to Non-ORACLE Data Sources."

Applies to block

Set Form Builder

Default

No

Usage Notes

- Transactional Triggers applies only when the Base Table property is NULL.
- Setting Transactional Triggers to Yes enables the Enforce Primary Key and Enforce Column Security properties.

Trigger Style property

Description

Specifies whether the trigger is a PL/SQL trigger or a V2-style trigger. Oracle Corporation recommends that you write PL/SQL triggers only. V2-style trigger support is included only for compatibility with previous versions.

Applies to trigger

Set Form Builder

Default

PL/SQL

Usage Notes

Choosing V2-Style trigger enables the Zoom button, which opens the trigger Step property sheet.

Trigger Text property

Description

Specifies the PL/SQL code that Form Builder executes when the trigger fires.

Applies to trigger

Set Form Builder

Required/Optional required

Trigger Type property

Description

Specifies the type of trigger, either built-in or user-named. User-named triggers are appropriate only in special situations, and are not required for most applications.

Applies to:

trigger

Set Form Builder

Default

PL/SQL

Required/Optional required

Usage Notes

trigger type can be one of the following:

Built-i	Specifies that the trigger is one provided by Form Builder and corresponds to a specific, pre-defined runtime event.
User-named	Specifies that the trigger is not provided by Form Builder. A user-named trigger can only be executed by a call to the EXECUTE_TRIGGER built-in procedure.

Update Allowed (Block) property

Description

Determines whether end users can modify the values of items in the block that have the Update Allowed item property set to Yes. (Setting Update Allowed to No for the block overrides the Update Allowed setting of any items in the block.)

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default

Yes

Update Allowed (Block) restrictions

When the Update Allowed block property is set to Yes, at least one item in the block must have the Update Allowed item property set to Yes for the block to be updateable.

Update Allowed (Item) property

Description

Specifies whether end users should be allowed to change the value of the base table item in a queried record. When Update Allowed is set to No, end users can navigate to the item in a queried record, but if they attempt to change its value, Form Builder displays error FRM-40200: Field is protected against update.

Setting Update Allowed to Yes does not prevent end users from entering values in a NEW (INSERT) record.

Applies to all items except buttons, chart items, and image items

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_INSTANCE_PROPERTY
- GET_ITEM_PROPERTY
- SET_ITEM_INSTANCE_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Usage Notes

To set the Update Allowed (Item) property programmatically, you can use the constant UPDATE_ALLOWED or UPDATEABLE. The constant UPDATEABLE is for compatibility with prior releases.

If Enabled is set to PROPERTY_FALSE at runtime, then the items' or item instance's Update Allowed property is also set to PROPERTY_FALSE.

- When Update Allowed is specified at multiple levels (item instance, item, and block), the values are ANDed together. This means that setting Update Allowed to Yes (PROPERTY_TRUE for runtime) has no effect at the item instance level unless it is set consistently at the block and item levels. For example, your user cannot update an item instance if Update Allowed is true at the instance level, but not at the item or block levels.

Update Allowed (Item) restrictions

- If you are using SET_ITEM_PROPERTY to set UPDATE_ALLOWED to true, then you must set item properties as follows:
  Enabled to Yes (PROPERTY_TRUE for runtime)
  Visible to Yes (PROPERTY_TRUE for runtime)
  Base Table Item to Yes (PROPERTY_TRUE for runtime)
  Update Only If Null to No (PROPERTY_FALSE for runtime)

Update Changed Columns Only property

Description

When queried records have been marked as updates, specifies that only columns whose values were actually changed should be included in the SQL UPDATE statement that is sent to the database during a COMMIT. By default, Update Changed Columns Only is set to No, and all columns are included in the UPDATE statement.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Default :

No

Required/Optional optional

Usage Notes

- If the DML Array Size property is set to a value greater than 1, this Update Changed Columns Only property will be ignored at runtime. That is, a DML Array Size greater than 1 causes all columns to be updated – even if Update Changed Columns Only was set to Yes.
- When Update Changed Columns Only is No, Form Builder can reuse the same SQL statement for multiple updates, without having to reparse each time. Setting Update Changed Columns Only to Yes can degrade performance because the UPDATE statement must be reparsed each time. In general, you should only set Update Changed Columns Only to Yes when you know that operators will seldom update column values that will take a long time to transfer over the network, such as LONGs.
- Set Update Changed Columns Only to Yes in the following circumstances:

  To save on network traffic, if you know an operator will primarily update only one or two columns.

  To avoid re-sending large items that are not updated, such as images or LONGs.

  To fire database triggers on changed columns only. For example, if you implement a security scheme with a database trigger that fires when a column has been updated and writes the userid of the person performing the update to a table.

Update_Column property

Description

When set to Yes, forces Form Builder to treat this item as updated.

If the Update Changed Columns Only block property is set to Yes, setting Update Column to Property_True specifies that the item has been updated and its corresponding column should be included in the UPDATE statement sent to the database.

If the Update Changed Columns Only block property is set to Yes, and Update Column is set to Property_False, the item's column will not be included in the UPDATE statement sent to the database.

If the Updated Changed Columns block property is set to No, the Update Column setting is ignored, and all base table columns are included in the UPDATE statement.

Applies to item

Set programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Required/Optional optional

Usage Notes

The main use for this property is in conjunction with Update Changed Columns Only. However, whether or not Update Changed Columns Only is set to Yes, you can use this property to check whether a given column was updated.

Note: Although Update Column affects Record Status, setting this property to Property_Off for all columns will not return Record Status to QUERY. If you want Record Status to revert to QUERY, you must set it explicitly with SET_RECORD_PROPERTY.

Update Commit property

Description

Specifies whether a chart item is updated to reflect changes made by committing new or updated records to its source block.

Applies to chart item

Set Form Builder

Default

Yes

Required/Optional required

Update Layout property

Description

Specifies when the frame's layout is updated.

Automatically	The layout is updated whenever the frame is moved or resized or whenever any frame layout property is modified.
Manually	The layout is updated whenever the Layout Wizard is used to modify a frame or whenever the user clicks the Update Layout button or menu option.
Locked	The layout is locked and cannnot be updated.

Applies to frame

Set Form Builder

Default

Yes

Required/Optional required

Update Only if NULL property

Description

Indicates that operators can modify the value of the item only when the current value of the item is NULL.

Applies to image items, list items, sound items, text items

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

No

Required/Optional optional

Usage Notes

To set the Update Only if NULL property programmatically, use the constant UPDATE_NULL.

Update Only if NULL restrictions

Item properties must be set as follows:

- Enabled set to Yes
- Visible set to Yes
- Update Allowed set to No

Update_Permission property

Description

Setting Update_Permission to No performs the following three actions:

- Sets the Update_If_Null property to No.
- Sets the Update Allowed property to No.
- Specifies that this column should not be included in any UPDATE statements issued by Form Builder, by removing that column from the SET clause of the UPDATE statements.

Applies to all items except buttons and chart items

Set programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY

Default

Yes

Required/Optional optional

Usage Notes

Update_Permission allows form developers to implement their own security mechanism, overriding the Form Builder default Enforce Column Security property. This property is included primarily for applications that will run against non-ORACLE data sources. Use Update_Permission when you want to exclude certain columns from any UPDATE statements: for example, when using an On-Column-Security trigger to implement a custom security scheme.

Update Procedure Arguments property

Description

Specifies the names, datatypes, and values of the arguments that are to be passed to the procedure for updating data. The Update Procedure Arguments property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Update Procedure Name property

Description

Specifies the name of the procedure to be used for updating data. The Update Procedure Name property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Update Procedure Result Set Columns property

Description

Specifies the names and datatypes of the result set columns associated with the procedure for updating data. The Update Procedure Result Set Columns property is valid only when the DML Data Target Type property is set to Procedure.

Applies to block

Set Form Builder

Default

NULL

Required/Optional optional

Update Query property

Description

Specifies whether a chart item is updated to reflect changes made by querying records in its source block.

Applies to chart item

Set Form Builder

Default

Yes

Required/Optional required

Use Security property

Description

Specifies that Form Builder should enforce the security scheme defined for the menu module, using the Menu Module Roles property.

Applies to menu module

Set Form Builder

Default

No

Usage Notes

This property can be set to No so that developers can test a menu module without having to be members of any database role. Use Security can then be set to Yes at production to enforce those roles.

Use Security restrictions

none

Use 3D Controls property

Description

On Microsoft Windows, specifies that Form Builder displays items with a 3-dimensional, beveled look.

When Use 3D Controls is set to Yes, any canvas that has Visual Attribute Group set to Default will automatically be displayed with background color grey.

In addition, when Use 3D Controls is set to Yes, the bevel for each item automatically appears lowered, even if an item-level property is set, for example, to raised.

Applies to form

Set Form Builder

Default

For a new form, Yes. For a form upgraded from a previous version of Form Builder, No.

Use 3D Controls restrictions

Valid only on Microsoft Windows.

Username property

Description

Specifies the username of the current operator.

Applies to application

Set not settable

Refer to Built-in

GET_APPLICATION_PROPERTY

Usage Notes

May be used with the LOGON built-in in an On-Logon trigger or for connecting to a non-ORACLE data source.

The Username property returns only the username. If you want a connect string as well, examine the Connect_String property.

User_Date/Datetime_Format property

Description

Holds the current date or datetime format mask established by the environment variable FORMSnn_USER_DATE_FORMAT or FORMSnn_USER_DATETIME_FORMAT.

There are two separate properties: User_Date_Format and User_Datetime_Format.

Applies to application

Set Not settable from within Form Builder.

Refer to Built-in

GET_APPLICATION_PROPERTY

User_Interface property

Description

Specifies the name of the user interface currently in use.

Applies to application

Set not settable

Refer to Built-in

GET_APPLICATION_PROPERTY

Usage Notes

This property returns one of the following values:

- BLOCKMODE
- CHARMODE
- MACINTOSH
- MOTIF
- MSWINDOWS
- MSWINDOWS32
- PM
- WIN32COMMON
- WEB
- X

User_NLS_Date_Format property

Description

Obtains the current NLS date format mask.

Applies to application

Set Not settable from within Form Builder.

Refer to Built-in

GET_APPLICATION_PROPERTY

Note that this property is read-only. That is, you cannot specify it in SET_APPLICATION_PROPERTY.

For example, if you wanted to set the PLSQL_DATE_FORMAT property to the current NLS date format mask value, you could code the following in a WHEN-NEW-FORM-INSTANCE trigger in your application:

```
SET_APPLICATION_PROPERTY(PLSQL_DATE_FORMAT,
GET_APPLICATION_PROPERTY(USER_NLS_DATE_FORMAT));
```

Default

None.

User_NLS_Lang property

Description

Specifies the complete value of the NLS_LANG environment variable defined for the current Runform session, for national language support. USER_NLS_LANG is the equivalent of concatenating the following properties:

- USER_NLS_LANGUAGE (language only)
- USER_NLS_TERRITORY (territory only)
- USER_NLS_CHARACTER_SET (character set only)

Applies to application

Set Not settable from within Form Builder. Set at your operating system level.

Refer to Built-in

GET_APPLICATION_PROPERTY

Default

Default is usually "America_American.WE8ISO8859P1," but all the defaults can be port-specific.

Validate from List property

Description

Specifies whether Form Builder should validate the value of the text item against the values in the attached LOV.

Applies to text item

Set Form Builder

Default

No

Required/Optional optional

Restrictions:

List of Values property must be specified.

Usage Notes

When Validate from List is Yes, Form Builder compares the current value of the text item to the values in the first column displayed in the LOV whenever the validation event occurs:

- If the value in the text item matches one of the values in the first column of the LOV, validation succeeds, the LOV is not displayed, and processing continues normally.
- If the value in the text item does not match one of the values in the first column of the LOV, Form Builder displays the LOV and uses the text item value as the search criteria to automatically reduce the list.

  For example, if the operator enters the first three digits of a 6-digit product code and then tries to navigate to the next item, Form Builder displays the LOV and auto-reduces the list to display all of the codes that have the same first three digits.
- If the operator selects a value from the LOV, Form Builder dismisses the LOV and assigns the selected values to their corresponding return items.

When you use an LOV for validation, Form Builder generally marks a text item as Valid if the operator selects a choice from the LOV. Thus, it is your responsibility to ensure that:

- the text item to which the LOV is attached is defined as a return item for the first column displayed in the LOV *and*
- the values in the LOV are valid

Note, however, that a When-Validate-Item trigger on the item still fires, and any validation checks you perform in the trigger still occur.

Note also that the first column displayed in the LOV may not be the first column in the LOV's underlying record group, as some record group columns may not have been included in the LOV structure, or may be hidden columns.

Validation property

Description

Specifies whether default Form Builder validation processing has been enabled or disabled for a form.

Applies to form module

Set programmatically

Refer to Built-in

- GET_FORM_PROPERTY
- SET_FORM_PROPERTY

Default

Yes

Usage Notes

Use this property with caution, because when you set Validation to No all internal form validation will be bypassed and no WHEN-VALIDATE triggers will fire.

You can programmatically set Validation to No for only brief periods of time when you specifically want to avoid all default Form Builder validation behavior. Once you set Validation to Yes again, any text items left in an unvalidated state will be validated according to normal processing rules.

When Validation is set to No, the Post-Change trigger will fire during query processing but will *not* fire elsewhere.

Validation Unit property

Description

Specifies the scope of form validation at runtime. Specifically, the validation unit defines the maximum amount of data that an operator can enter in the form before Form Builder initiates validation. For most applications, the Validation Unit is Item (default setting on most platforms), which means that Form Builder validates data in an item as soon as the operator attempts to navigate out of the item.

Applies to form module

Set Form Builder, programmatically

Refer to Built-in

- GET_FORM_PROPERTY
- SET_FORM_PROPERTY

Default

Default

Usage Notes

The following settings are valid for this property:

- Default
- Form
- Block
- Record
- Item

Value when Checked property

Description

Specifies the value you want the check box to display as the checked state. For example, Y, 1, MANAGER, or 1992. When a value that matches the checked value is fetched or assigned to the check box, the check box is displayed checked. Similarly, when the operator toggles the check box to the checked state, the value of the check box becomes the checked value.

Applies to check box

Set Form Builder

Default

NULL

Required/Optional optional

Value when Checked restrictions

The value must be compatible with the datatype specified by the Parameter Data Type property.

Value when Unchecked property

Description

Specifies the value you want the check box to display as the unchecked state. For example, Y, 1, MANAGER, or 1992. When a value that matches the unchecked value is fetched or assigned to the check box, the check box is displayed unchecked. Similarly, when the operator toggles the check box to the unchecked state, the value of the check box becomes the unchecked value.

Applies to check box

Set Form Builder

Default

NULL

Required/Optional Optional; leaving this property blank makes the Unchecked value NULL.

Value when Unchecked restrictions

The value must be compatible with the datatype specified by the Parameter Data Type property.

VBX Control File property

Description

Specifies the VBX file selection.

Applies to VBX Control

Set Form Builder

Default

none

Required/Optional required

Usage Notes

The selection of a VBX file determines which VBX controls are available for use. The number and type of VBX files available for selection depends on the third-party VBX controls that are installed on your system.

Because moving a form module with hard-coded paths to another computer system can make the VBX file and location invalid, you should avoid specifying an absolute path for the VBX Control File property.

For a VBX control file that is not associated with an absolute path, the search criteria is the system default search path. If all default search paths fail to locate the specified VBX control file, the FORMS60_PATH parameter in the ORACLE.INI file becomes the search criteria for finding the VBX control file. If all search paths in the FORMS60_PATH parameter fail to locate the VBX control file, a runtime error message informs you that the VBX control cannot be found.

VBX Control File restrictions

Valid only on Microsoft Windows 3.x (16-bit).

VBX Control Name property

Description

Specifies the VBX control selection from a VBX file. Some VBX files contain more than a single VBX control. You must specify which VBX control to use even when a VBX file contains on a single VBX control.

Applies to VBX Control

Set Form Builder

Default

none

VBX Control Name restrictions

Valid only on Microsoft Windows 3.x (16-bit).

VBX Control Value property

Description

Specifies the value property of a VBX control. This property determines the value of the VBX custom item in Form Builder.

Applies to VBX Control

Set Form Builder

Refer to Built-in

- VBX.GET_VALUE_PROPERTY
- VBX.SET_VALUE_PROPERTY

Default

Most VBX controls have a default value property. If the default value property exists, it is the default Form Builder VBX Control Value Property. If the VBX control does not have a default value property, the Form Builder VBX Control Value Property is the VBX property named "value". If the VBX property "value" does not exist, a default value property is not assigned to the Form Builder VBX Control Value Property.

Required/Optional required

Usage Notes

The VBX CONTROL VALUE PROPERTY automatically synchronizes itself with a VBX property. Changes to the VBX property are reflected in the VBX CONTROL VALUE PROPERTY.

VBX Control Value restrictions

Valid only on Microsoft Windows 3.x (16-bit).

Vertical Fill property

Description

Specifies whether the Layout Wizard uses the empty space surrounding an object when the Layout Style property is set to Form.

Yes	Specifies that the Layout Wizard should use all available space when arranging frame objects. Consequently, Form Builder ignores the Maximum Objects Per Line property.
No	Specifies that the Layout Wizard should not use all available space when arranging frame objects. When objects wrap, they begin on the next frame line.

Applies to frame

Set Form Builder

Default

Yes

Required/Optional required

Vertical Justification property

Description

Specifies the vertical justification of the text object as either Top, Center, or Bottom.

Applies to graphic text

Set Form Builder

Default

Top

Required/Optional required

Vertical Margin property

Description

Specifies the distance between the frame's top and bottom borders and the objects within the frame.

Applies to frame

Set Form Builder

Default

1 character cell (or the equivalent based on the form coordinate system)

Required/Optional required

Vertical Object Offset property

Description

Specifies the vertical distance between the objects within a frame.

Applies to frame

Set Form Builder

Default

0

Required/Optional required

Vertical Origin property

Description

Specifies the vertical position of the text object relative to its origin point as either Top, Center, or Bottom.

Applies to graphic text

Set Form Builder

Default

Top

Required/Optional required

Vertical Toolbar Canvas property

Description

Specifies the canvas that should be displayed as a vertical toolbar on the window. The canvas you specify must be a vertical toolbar canvas (Canvas Type property set to Vertical Toolbar) and must be assigned to the current window by setting the Window property.

Applies to window

Set Form Builder

Default

Null

Required/Optional required if you are creating a vertical toolbar

Usage Notes

- In the Properties window, the poplist for this property shows only canvases that have the Canvas Type property set to Vertical Toolbar.
- At runtime, Form Builder attempts to display the specified vertical toolbar on the window. However, if more than one toolbar of the same type has been assigned to the same window (by setting the canvas Window property to point to the specified window), Form Builder may display a different toolbar in response to navigation events or programmatic control.
- On Microsoft Windows, the specified vertical toolbar canvas will not be displayed on the window if you have specified that it should instead be displayed on the MDI application window by setting the Form Vertical Toolbar Canvas form property.

Viewport Height, Viewport Width property

Description

Specifies the width and height of the view for a stacked canvas. The size and position of the view define the part of the canvas that is actually displayed in the window at runtime.

Note: For a content or toolbar canvas, the view is represented by the window to which the canvas is assigned, and so the Viewport Height and Viewport Width properties do not apply.

Applies to canvas

Set Form Builder, programmatically

Refer to Built-in

SET_VIEW_PROPERTY

Default

0,0

Required/Optional optional

Viewport Height, Viewport Width restrictions

Valid only for a stacked view (Canvas Type property set to Stacked). For a content view, the viewport size is determined by the runtime size of the window in which the content view is displayed.

Viewport X Position, Viewport Y Position property

Description

Specifies the x,y coordinates for the stacked canvas's upper left corner relative to the upper left corner of the window's current content view.

Applies to canvas

Set Form Builder, programmatically

Refer to Built-in

- GET_VIEW_PROPERTY
- SET_VIEW_PROPERTY

Default

0,0

Required/Optional optional

Viewport X Position, Viewport Y Position restrictions

Not valid for a content canvas view; that is, a canvas view that has the Canvas Type property set to Content.

Viewport X Position on Canvas, Viewport Y Position on Canvas property

Description

Specifies the location of the view's upper left corner relative to the upper left corner of the canvas. The size and location of the viewport define the *view*; that is, the part of the canvas that is actually visible in the window to which the canvas is assigned.

Applies to canvas

Set Form Builder, programmatically

Refer to Built-in

- GET_VIEW_PROPERTY
- SET_VIEW_PROPERTY

Default

0,0

Visible property

Description

Indicates whether the object is currently displayed or visible. Set Visible to Yes or No to show or hide a canvas or window.

Applies to canvas, window

Set programmatically

Refer to Built-in

- GET_VIEW_PROPERTY
- GET_WINDOW_PROPERTY
- SET_VIEW_PROPERTY
- SET_WINDOW_PROPERTY

Default

TRUE

Usage Notes

- You cannot hide the canvas that contains the current item.
- You can hide a window that contains the current item.

  NOTE: In some operating systems, it is possible to hide the only window in the form.
- When you use GET_WINDOW_PROPERTY to determine window visibility, Form Builder uses the following rules:
- A window is considered visible if it is displayed, even if it is entirely hidden behind another window.
- A window that has been iconified (minimized) is reported as visible to the operator because even though it has a minimal representation, it is still mapped to the screen.
- When you use GET_VIEW_PROPERTY to determine canvas visibility, Form Builder uses the following rules:
- A view is reported as visible when it is a) in front of all other views in the window or b) only partially obscured by another view.
- A view is reported as not visible when it is a) a stacked view that is behind the content view in the window or b) completely obscured by *a single stacked view*. Note that a view is reported as visible even if it is completely obscured by a combination of *two or more stacked views*.
- The display state of the window does not affect the setting of the canvas VISIBLE property. That is, a canvas may be reported visible even if the window in which it is displayed is not currently mapped to the screen.

Visible (Canvas) property

Description

Determines whether a stacked canvas is initially shown or hidden in the window to which it is assigned.

Applies to:

stacked canvas

Set:

Form Builder, programmatically

Refer to Built-in

- GET_VIEW_PROPERTY (VISIBLE)
- SET_VIEW_PROPERTY (VISIBLE)

Default:

Yes

Visible (Canvas) restrictions

- A displayed view may not be visible if it is behind the content view or another stacked view assigned to the same window.

Visible (Item) property

Description

Determines whether an item that is assigned to a canvas is shown or hidden at runtime.

Applies to all items

Set Form Builder, programmatically

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Default

Yes

Usage Notes

To set the Visible (Item) property programmatically, you can use the constant VISIBLE or DISPLAYED. The constant DISPLAYED is for compatibility with prior releases.

Visible (Item) restrictions

When the item is part of the foreign key in a default master-detail relation, the default is No.

Visible (Tab Page) property

Description

Determines whether a tab page is shown or hidden at runtime.

Applies to:

tab page

Applies to:

Form Builder, programmatically

Refer to Built-in

- GET_TAB_PAGE_PROPERTY
- SET_TAB_PAGE_PROPERTY

Default:

Yes

Visible in Horizontal/Vertical Menu Toolbar property

Description

Specifies whether the menu item should appear (represented by an icon) on the horizontal or vertical menu toolbar (or both) of a form.

Applies to menu item

Set Form Builder

Default

No

Required/Optional optional

Visible in Horizontal/Vertical Menu Toolbar restrictions

Developers must provide icons to associate with each menu item that appears on a menu toolbar.

Visible in Menu property

Description

Determines whether the menu item is shown or hidden at runtime.

Applies to:

menu item

Set:

Form Builder, programmatically

Refer to Built-in

- GET_MENU_ITEM_PROPERTY
- SET_MENU_ITEM_PROPERTY

Default:

Yes

Visual Attribute property

Description

Specifies the named visual attribute that should be applied to the object at runtime. A visual attribute defines a collection of font, color, and pattern attributes that determine the appearance of the object.

Applies to canvas, tab page, item, radio button

Set programmatically

Refer to Built-in

- GET_ITEM_INSTANCE_PROPERTY
- GET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- GET_TAB_PAGE_PROPERTY
- SET_CANVAS_PROPERTY
- SET_ITEM_INSTANCE_PROPERTY
- SET_ITEM_PROPERTY
- SET_RADIO_BUTTON_PROPERTY
- SET_TAB_PAGE_PROPERTY

Usage Notes

When you execute the appropriate GET_ built-in function to determine the setting of this property at runtime, the return value is one of the following:

- the name of a named visual attribute
- the name of a logical attribute defined in the resource file
- DEFAULT (the item uses the default attributes defined in the resource file)

Visual Attribute restrictions

The visual attribute must be a named visual attribute defined in the form module or a logical attribute defined in the runtime resource file.

Visual Attribute Group property

Description

Specifies how the object's individual attribute settings (Font Name, Background Color, Fill Pattern, etc.) are derived. The following settings are valid for this property:

Default	Specifies that the object should be displayed with default color, pattern, and font settings. When Visual Attribute Group is set to Default, the individual attribute settings reflect the current system defaults. The actual settings are determined by a combination of factors, including the type of object, the resource file in use, and the platform.
Named visual attribute	Specifies a named visual attribute that should be applied to the object. Named visual attributes are separate objects that you create in the Object Navigator and then apply to interface objects, much like styles in a word processing program. When Visual Attribute Group is set to a named visual attribute, the individual attribute settings reflect the attribute settings defined for the named visual attribute object. When the current form does not contain any named visual attributes, the poplist for this property will show Default.

Applies to all interface objects

Set Form Builder

Default

Default

Usage Notes

- Default and named visual attributes can include the following individual attributes, listed in the order they appear in the Property Palette:

  Font Name The font family, or typeface, that should be used for text in the object. The list of fonts available is system-dependent.

  Font Size The size of the font, specified in points.

  Font Style The style of the font.

  Font Spacing The width of the font, that is, the amount of space between characters (kerning).

  Font Weight The weight of the font.

  Foreground Color The color of the object's foreground region. For items, the Foreground Color attribute defines the color of text displayed in the item.

Background Color The color of the object's background region.

Fill Pattern The pattern to be used for the object's fill region. Patterns are rendered in the two colors specified by Background Color and Foreground Color.

Character Mode Logical Attribute Specifies the name of a character mode logical attribute defined in an Oracle Terminal resource file that is to be used as the basis of device attributes for a character mode version of your application.

White on Black Specifies that the object is to appear on a monochrome bitmap display device as white text on a black background.

Not all attributes are valid for each object type. For example, setting font attributes for a window object has no effect. (The font used in a window's title bar is derived from the system.)

A new object in a new form has Default visual attributes. The default settings are defined internally. You can override the default font for new items and boilerplate by setting the optional FORMS60_DEFAULTFONT environment variable. For example, on Microsoft Windows, you can set this variable in the registry, as follows: `FORMS60_DEFAULTFONT="COURIER.10".` The default font specified determines the font used for new boilerplate text generated by the New Block window, and for any items that have Visual Attribute Group set to Default.

When you create an item in the Layout Editor, its initial visual attribute settings are determined by the current Layout Editor settings for fonts, colors, and patterns, as indicated by the Font dialog and Color and Pattern palettes.

On Microsoft Windows, the colors of buttons, window title bars, and window borders are controlled by the Windows Control Panel color settings specified for these elements. You cannot override these colors in Form Builder.

When the Use 3D Controls form property is set to Yes on Microsoft Windows (the default), items are rendered with shading that provides a sculpted, three-dimensional look. A side effect of setting this property is that any canvases that have Visual Attribute Group set to Default derive their color setting from the Windows Control Panel (gray for most color schemes). You can override this setting by explicitly applying named visual attributes to the canvas.

An item that has Visual Attribute Group set to Default, or that has individual attribute settings left unspecified, inherits those settings from the canvas to which it is assigned. Similarly, a canvas that has Visual Attribute Group set to Default, or that has individual attribute settings left unspecified, inherits those settings from the window in which it is displayed. For example, if you set a window's Background Color to CYAN, and then leave Background Color unspecified for the canvas assigned to the window, at runtime, that canvas will inherit the CYAN background from its window. Visual attribute settings derived through window--canvas or canvas--item inheritance are apparent at design time if the Layout Editor is reopened.

You can apply property classes to objects to specify visual attribute settings. A property class can contain either the Visual Attribute Group property, or one or more of the individual attribute properties. (If a property class contains both Visual Attribute Group and individual attributes, the Visual Attribute Group property takes precedence.)

If you apply both a named visual attribute and a property class that contains visual attribute settings to the same object, the named visual attribute settings take precedence, and the property class visual attribute settings are ignored.

Logical attribute definitions defined in the resource file take precedence over visual attributes specified in the Form Builder, local environment variable definitions, and default Form Builder attributes. To edit the resource file, use the Oracle Terminal utility.

Visual Attribute Type property

Description

Specifies the type of the visual attribute during design time as either Common, Prompt, or Title.

Applies to visual attribute general

Set Form Builder

Default Common

Required/Optional required

WHERE Clause/ORDER BY Clause properties

Description

The default WHERE Clause and default ORDER BY Clause properties specify standard SQL clauses for the default SELECT statement associated with a data block. These clauses are automatically appended to the SELECT statement that Form Builder constructs and issues whenever the operator or the application executes a query in the block.

Applies to block

Set Form Builder, programmatically

Refer to Built-in

- GET_BLOCK_PROPERTY
- SET_BLOCK_PROPERTY

Required/Optional optional

Usage Notes

- The reserved words WHERE and ORDER BY are optional. If you do not include them, Form Builder automatically prefixes the statement with these words.
- WHERE Clause can reference the following objects:

  columns in the block's data block table (except LONG columns)

  form parameters (:PARAMETER.my_parameter)
- ORDER BY Clause can reference the following objects:

  columns in the block's data block table (except LONG columns)
- Embedded comments are not supported in WHERE Clause and ORDER BY Clause.

WHERE Clause/ORDER BY Clause restrictions

- Maximum length for WHERE Clause is 32,000 bytes.
- ORDER BY clause cannot reference global variables or form parameters.

WHERE Clause/ORDER BY Clause examples

Example

In the following example from an order tracking system, the WHERE Clause limits the retrieved records to those whose *shipdate* column is NULL. The ORDER BY Clause arranges the selected records from the lowest (earliest) date to the highest (latest) date.

```
WHERE shipdate IS NULL
ORDER BY orderdate
```

This WHERE Clause/ORDER BY Clause statement specifies the base conditions for record retrieval. The operator can further restrict the records retrieved by placing the form in Enter Query mode and entering ad hoc query conditions.

White on Black property

Description

Specifies that the object is to appear on a monochrome bitmap display device as white text on a black background.

Applies to item, tab page, canvas, window, radio button

Set Programmatically

Default

Unspecified

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY
- GET_TAB_PAGE_PROPERTY
- SET_TAB_PAGE_PROPERTY
- GET_CANVAS_PROPERTY
- SET_CANVAS_PROPERTY
- GET_WINDOW_PROPERTY
- SET_WINDOW_PROPERTY

Width/Height (WD, HT) properties

Description

See Size property

Window property

Description

Specifies the window in which the canvas will be displayed at runtime.

Applies to canvas

Set Form Builder

Refer to Built-in

GET_VIEW_PROPERTY

Default

ROOT_WINDOW, if there is a root window in the form, else the first window listed under the Windows node in the Object Navigator.

Required/Optional required for the canvas to be displayed at runtime

Window_Handle property

Description

On Microsoft Windows, a window handle is a unique internal character constant that can be used to refer to objects. It is possible to obtain a window handle for any item or window.

Applies to form, block, item

Refer to Built-in

- GET_ITEM_PROPERTY
- GET_WINDOW_PROPERTY
- GET_RADIO_BUTTON_PROPERTY

Default

NULL

Usage Notes

- Specify the name of the item and the WINDOW_HANDLE property in GET_ITEM_PROPERTY to obtain the window handle to an item.
- Specify the name of the window and the WINDOW_HANDLE property in GET_WINDOW_PROPERTY to obtain the window handle to a window. If the name of the window of GET_WINDOW_PROPERTY is FORMS_MDI_WINDOW, the return value is a handle to the MDI client window. The handle to a MDI client window is used to create child MDI windows and controls.
- Specify the item name or item id of the radio group, the name of the radio button, and the WINDOW_HANDLE property in GET_RADIO_BUTTON_PROPERTY to obtain a window handle to a radio button.
- To obtain a window handle to a radio group, use the name of the radio group as the item name in GET_ITEM_PROPERTY. A window handle to the button that is in focus is returned. If no button is in focus, the window handle to the button that is selected is returned. If neither a focused or selected button exists, the window handle to the first button is returned.

Window_Handle restrictions

Valid only on Microsoft Windows. (Returns NULL on other platforms.)

Window_State property

Description

Specifies the current display state of the window:

NORMAL	Specifies that the window should be displayed normally, according to its current Width, Height, X Position, and Y Position property settings.
MINIMIZE	Specifies that the window should be minimized, or iconified so that it is visible on the desktop s a bitmap graphic.
MAXIMIZE	Specifies that the window should be enlarged to fill the screen according to the display style of the window manager.

Applies to window

Set Programmatically

Refer to Built-in

- GET_WINDOW_PROPERTY
- SET_WINDOW_PROPERTY

Default

NORMAL

Usage Notes

The minimize and maximize display states are managed by the window manager and do not affect the window's current width and height settings, as defined by the Width and Height properties. Thus, if a window display state is currently minimized or maximized, any call to SET_WINDOW_PROPERTY or RESIZE_WINDOW that changes the Width or Height properties will be applied, but will not become apparent to the operator until the window is returned to the Normal state.

Similarly, GET_WINDOW_PROPERTY always returns the window's current Width and Height property settings, even if the window is currently in the minimized or maximized display state.

Window_State restrictions

Setting Window_State to MAXIMIZE is not supported on Motif.

Window Style property

Description

Specifies whether the window is a Document window or a Dialog window. Document and dialog windows are displayed differently on window managers that support a Multiple Document Interface (MDI) system of window management.

Applies to window

Set Form Builder

Default

Document

Restrictions:

Valid only for a secondary window. (A root window is always a document window.)

Usage Notes

MDI applications display a default parent window, called the *application* window. All other windows in the application are either *document* windows or *dialog* windows.

Document windows always remain within the application window frame. If the operator resizes the application window so that it is smaller than a document window, the document window is clipped. An operator can maximize a document window so that is occupies the entire workspace of the application window.

Dialog windows are free-floating, and the operator can move them outside the application window if they were defined as Movable. If the operator resizes the application window so that it is smaller than a dialog window, the dialog window is not clipped.

Wrap Style property

Description

Specifies how text is displayed when a line of text exceeds the width of a text item or editor window.

The following list describes the allowable values for this property:

NONE	No wrapping: text exceeding the right border is not shown.
CHARACTER	Text breaks following the last visible character, and wraps to the next line.
WORD	Text breaks following last visible complete word, and wraps to the next line.

Applies to text item, editor

Set Form Builder

Refer to Built-in

GET_ITEM_PROPERTY

Default

WORD

Wrap Style restrictions

Valid only for multi-line text items.

Wrap Text property

Description

Specifies whether the text in the text object wraps to the next line to fit within the bounding box.

Applies to graphic text

Set Form Builder

Default

Yes

Required/Optional required

X Corner Radius property

Description

Specifies the amount of horizontal rounding (in layout units) of the corners of the rounded rectangle.

Applies to graphic rounded rectangle

Set Form Builder

Default

10

Required/Optional required

X Position, Y Position property

Description

For an object, specifies where it appears on the screen. For an item, specifies the position of the item's upper left corner relative to the upper left corner of the item's canvas. The values you specify are interpreted in the current form coordinate units (character cells, centimeters, inches, pixels, or points), as specified by the Coordinate System form property.

Applies to all items, editors, LOVs, windows, canvases

Set Form Builder, programmatically

Usage Notes

The following information is specific to the current object.

ITEM

Determines where the item appears on the owning canvas.

Refer to Built-in

- GET_ITEM_PROPERTY
- SET_ITEM_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- SET_RADIO_BUTTON_PROPERTY

Default

x,y(0,0)

LOV

Determines where the LOV appears on the screen: (0,0) is the upper left corner of the entire screen, regardless of where the root window appears on the screen. The LOV can be displayed anywhere on the screen, including locations outside the form.

Refer to Built-in

- GET_LOV_PROPERTY
- SET_LOV_PROPERTY

Default

x,y(0,0)

WINDOW

Determines where the window appears on the screen: (0,0) is the upper left corner of the entire screen.

Refer to Built-in

- GET_WINDOW_PROPERTY
- SET_WINDOW_PROPERTY

Default

x,y(0,0)

X Position, Y Position restrictions

- Values for all items, editors, and LOVs must be non-negative.
- Precision allowed is based on the current form coordinate units. Rounding may occur when necessary.

Y Corner Radius property

Description

Specifies the amount of vertical rounding (in layout units) of the corners of the rounded rectangle.

Applies to graphic rounded rectangle

Set Form Builder

Default

10

Required/Optional required

System Variables

About system variables

A system variable is an Form Builder variable that keeps track of an internal Form Builder state. You can reference the value of a system variable to control the way an application behaves.

Form Builder 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 names of the available system variables are:

- SYSTEM.BLOCK_STATUS
- SYSTEM.COORDINATION_OPERATION
- SYSTEM.CURRENT_BLOCK
- SYSTEM.CURRENT_DATETIME
- SYSTEM.CURRENT_FORM
- SYSTEM.CURRENT_ITEM
- SYSTEM.CURRENT_VALUE
- SYSTEM.CURSOR_BLOCK
- SYSTEM.CURSOR_ITEM
- SYSTEM.CURSOR_RECORD
- SYSTEM.CURSOR_VALUE
- SYSTEM.CUSTOM_ITEM_EVENT
- SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS
- SYSTEM.DATE_THRESHOLD*
- SYSTEM.EFFECTIVE_DATE*
- SYSTEM.EVENT_WINDOW
- SYSTEM.FORM_STATUS
- SYSTEM.LAST_QUERY
- SYSTEM.LAST_RECORD
- SYSTEM.MASTER_BLOCK
- SYSTEM.MESSAGE_LEVEL*
- SYSTEM.MODE
- SYSTEM.MOUSE_BUTTON_PRESSED
- SYSTEM.MOUSE_BUTTON_SHIFT_STATE

- SYSTEM.MOUSE_ITEM
- SYSTEM.MOUSE_CANVAS
- SYSTEM.MOUSE_X_POS
- SYSTEM.MOUSE_Y_POS
- SYSTEM.MOUSE_RECORD
- SYSTEM.MOUSE_RECORD_OFFSET
- SYSTEM.RECORD_STATUS
- SYSTEM.SUPPRESS_WORKING*
- SYSTEM.TAB_NEW_PAGE
- SYSTEM.TAB_PREVIOUS_PAGE
- SYSTEM.TRIGGER_BLOCK
- SYSTEM.TRIGGER_ITEM
- SYSTEM.TRIGGER_RECORD

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. Form Builder also supplies 6 default values for date and time. (See Date and Time System Default Values).

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 instead of getting the system variable value each time.

Date and Time System Default Values

Form Builder supplies six special default values $$DATE$$, $$DATETIME$$, $$TIME$$, $$DBDATE$$, $$DBDATETIME$$, and $$DBTIME$$ that supply date and time information. These variables have the and the following special restrictions on their use:

- For client/server applications, consider the performance implications of going across the network to get date and time information.
- When 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 when, for example, connecting to a remote database in a different time zone.
- Use these variables only to set the value of the Initial Value, Highest Allowed Value or Lowest Allowed Value property.

About system variables examples

Assume that you want to create a Key-NXTBLK trigger at the form level that navigates depending on what the current block is. The following trigger performs this function, using :SYSTEM.CURSOR_BLOCK stored in a local variable.

```
DECLARE
   curblk VARCHAR2(30);
BEGIN
   curblk := :System.Cursor_Block;
   IF curblk = 'Orders'
      THEN Go_Block('Items');
   ELSIF curblk = 'Items'
      THEN Go_Block('Customers');
   ELSIF curblk = 'Customers'
      THEN Go_Block('Orders');
   END IF;
END;
```

Uppercase Return Values

All system variables are case-sensitive, and most return their arguments as *uppercase* values. This will affect the way you compare results in IF statements.

$$DATE$$ system variable

Syntax

```
$$DATE$$
```

Description

$$DATE$$ retrieves the current operating system date (client-side). Use $$DATE$$ to designate a default value or range for a text item using the Initial Value or Lowest/Highest Allowed Value properties. The text item must be of the CHAR, DATE, or DATETIME data type.

Use $$DATE$$ as a default value for form parameters. In this case, the parameter's value is computed once, at form startup.

Usage Notes

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.

The display of system variables is governed by the format mask, either a default data type format mask or one you specify. For example, to use the default DD-MON-YY format, specify a DATE data type. (Note that the default format mask depends on the value of NLS_LANG.)

Although $$DATE$$ *displays* only the date, its underlying value includes a time component which is saved at commit time. If you specify a DATETIME data type and provide $$DATE$$ as the default, the underlying value will be DD-MON-YYYY HH:MM:SS: for example, 01-DEC-1994 00:00:00 (although only 01-DEC-1994 will be displayed).

Use $$DATE$$ when you want to compare the contents of this field with a field whose format mask does not have a time component, such as a SHIPDATE field of data type DATE. In this case, both $$DATE$$ and SHIPDATE will have a time component of 00:00:00, so the comparison of two dates evaluating to the same day will be successful.

$$DATE$$ examples

Example 1

Assume that you want the value of a DATE text item, called ORDERDATE, to default to the current date. When you define the ORDERDATE text item, specify $$DATE$$ in the text item Initial Value property.

Example 2

If you use $$DATE$$ in a parameter, such as :PARAMETER.STARTUP_DATE, then every time you reference that parameter, the date you started the application will be available:

```
IF :PARAMETER.Startup_Date + 1 < :System.Current_Datetime
   THEN Message ('You have been logged on for more than a
day.');
ELSE Message ('You just logged on today.');
END IF;
```

$$DATETIME$$ system variable

Syntax

```
$$DATETIME$$
```

Description

$$DATETIME$$ 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 Initial Value or Lowest/Highest Allowed Value properties. The text item must be of the CHAR or DATETIME data type.

Use $$DATETIME$$ as a default value for form parameters. In this case, the parameter's value is computed once, at form startup.

Usage Notes

The display of system variables is governed by the format mask, either a default data type format mask or one you specify. For example, if you want the default DD-MON-YY HH:MM:SS format, you must specify a DATETIME data type. (Note that the default format mask depends on the value of NLS_LANG.)

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 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.

$$DATETIME$$ examples

Assume that you want the value of a DATETIME text item, called ORDERDATE, to default to the current operating system date and time. When you define the ORDERDATE text item, specify $$DATETIME$$ in the Initial Value property.

$$DBDATE$$ system variable

Syntax

```
$$DBDATE$$
```

Description

$$DBDATE$$ retrieves the current database date. Use $$DBDATE$$ to designate a default value or range for a text item using the Initial Value or Lowest/Highest Allowed Value properties. The text item must be of the CHAR, DATE, or DATETIME data type.

Usage Notes

The difference between $$DBDATE$$ and $$DBDATETIME$$ is that the time component for $$DBDATE$$ is always fixed to 00:00:00, compared to $$DBDATETIME$$, which includes a meaningful time component, such as 09:17:59.

Use $$DBDATE$$ to default a DATE item to the current date on the server machine, for example, when connecting to a remote database that may be in a different time zone from the client's time zone.

The display of system variables is governed by the format mask, either a default data type format mask or one you specify. For example, if you want the default DD-MON-YY format, you must specify a DATE data type. (Note that the default format mask depends on the value of NLS_LANG.)

Although $$DBDATE$$ *displays* only the date, its underlying value includes a time component which is saved at commit time. If you specify a DATETIME data type and provide $$DBDATE$$ as the default, the underlying value will be DD-MON-YYYY HH:MM:SS: for example, 01-DEC-1994 00:00:00 (although only 01-DEC-1994 will be displayed).

$$DBDATE$$ restrictions

- If you are accessing a non-ORACLE datasource, avoid using $$DBDATE$$. Instead, use a When-Create-Record trigger to select the current date in a datasource-specific manner.

$$DBDATE$$ examples

To have the value of a DATE text item called ORDERDATE default to the current database date, for the ORDERDATE text item, specify $$DBDATE$$ in the Initial Value property.

$$DBDATETIME$$ system variable

Syntax

```
$$DBDATETIME$$
```

Description

$$DBDATETIME$$ retrieves the current date and time from the local database. Use $$DBDATETIME$$ to designate a default value or range for a text item using the Initial Value or Lowest/Highest Allowed Value properties. The text item must be of the CHAR or DATETIME data type.

Usage Notes

Use $$DBDATETIME$$ to default a DATE item to the current date on the server machine, for example, when connecting to a remote database that may be in a different time zone from the client's time zone.

The display of system variables is governed by the format mask, either a default data type format mask or one you specify. For example, if you want a DD-MON-YY HH:MM:SS format, you must specify a DATETIME or CHAR data type. (Note that the default format mask depends on the value of NLS_LANG.)

If you are building a client/server application, using $$DBDATETIME$$ could have performance implications, depending on the complexity of your network configuration.

Note: Do not use $$DBDATETIME$$ instead of $$DBDATE$$ unless you plan to specify the time component. If, for example, you use $$DBDATETIME$$ 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.

$$DBDATETIME$$ restrictions

If you are accessing a non-ORACLE datasource, avoid using $$DBDATETIME$$. Instead, use a When-Create-Record trigger to select the current date and time in a datasource-specific manner.

$$DBDATETIME$$ examples

Assume that you want the value of a DATETIME text item, called ORDERDATE, to default to the current database date and time. When you define the ORDERDATE text item, specify $$DBDATETIME$$ in the Lowest/Highest Allowed Value properties.

$$DBTIME$$ system variable

Syntax

```
$$DBTIME$$
```

Description

$$DBTIME$$ retrieves the current time from the local database. Use $$DBTIME$$ to designate a default value or range for a text item using the Initial Value or Lowest/Highest Allowed Value properties. The text item must be of the CHAR or TIME data type.

Usage Notes

Use $$DBTIME$$ when connecting to a remote database that may be in a different time zone from the client's time zone.

The display of system variables is governed by the format mask, either a default data type format mask or one you specify. For example, if you want the default HH:MM:SS format, you must specify a TIME data type. (Note that the default format mask depends on the value of NLS_LANG.)

If you are building a client/server application, using $$DBTIME$$ could have performance implications, depending on the complexity of your network configuration.

$$DBTIME$$ restrictions

If you are accessing a non-ORACLE datasource, avoid using $$DBTIME$$. Instead, use a When-Create-Record trigger to select the current time in a datasource-specific manner.

$$DBTIME$$ examples

Assume that you want the value of a TIME text item, called ORDERTIME, to default to the current database time. When you define the ORDERTIME text item, specify $$DBTIME$$ in the Initial Value property.

$$TIME$$ system variable

Syntax

```
$$TIME$$
```

Description

$$TIME$$ retrieves the current operating system time. Use $$TIME$$ to designate a default value or range for a text item using the Initial Value or Lowest/Highest Allowed Value properties. 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.

Usage Notes

The display of system variables is governed by the format mask, either a default data type format mask or one you specify. For example, if you want the default HH:MM:SS format, you must specify a TIME data type. (Note that the default format mask depends on the value of NLS_LANG.)

$$TIME$$ examples

Assume that you want the value of a TIME text item, called ORDERTIME, to default to the current operating system time. When you define the ORDERTIME item, specify $$TIME$$ in the Initial Value property.

SYSTEM.BLOCK_STATUS system variable

Syntax

```
SYSTEM.BLOCK_STATUS
```

Description

SYSTEM.BLOCK_STATUS represents the status of a Data block where the cursor is located, or the current data 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.

Usage Notes

Each time this value is referenced, it must be constructed by Form Builder. If a block contains a large number of records, using SYSTEM.BLOCK_STATUS could adversely affect performance.

SYSTEM.BLOCK_STATUS examples

Assume that you want to create a trigger that performs a commit before clearing a block if there are changes to commit within that block. The following Key-CLRBLK trigger performs this function.

```
IF :System.Block_Status = 'CHANGED'
   THEN Commit_Form;
END IF;
Clear_Block;
```

SYSTEM.COORDINATION_OPERATION system variable

Syntax

```
SYSTEM.COORDINATION_OPERATION
```

Description

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.

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.

The Clear_All_Master_Details procedure, which is automatically generated when a relation is created, checks the value of SYSTEM.COORDINATION_OPERATION to provide special handling for the CLEAR_RECORD and SYNCHRONIZE events, which may be different from the handling for other coordination-causing events. The Clear_All_Master_Details procedure also checks the value of SYSTEM.MASTER_BLOCK , to verify that while it is processing the master block of a relation coordination, it is searching only for blocks containing changes.

For example, given the relation hierarchy between blocks shown below, moving to the next record using the [Next Record] key or the Record, Next menu command while in Block C would cause blocks E, F, G, and H to be cleared (and perhaps subsequently queried, depending on the Deferred_Coordination property of the CE and the CF relations).

When the On-Clear-Details trigger fires for block C, the result is:

```
:System.Cooordination_Operation   = 'NEXT_RECORD'
:System.Master_Block              = 'C'
```

The Clear_All_Master_Details procedure will clear all of block C's details, causing a "chain reaction" of Clear_Block operations. Consequently, block F is cleared.

Since F is a master for both G and H, and it is being cleared, an On-Clear-Details trigger will fire for block F as well. However, since the clearing of block F was driven (indirectly) by a coordination-causing event in block C, these remain the values in the On-Clear-Details trigger for block F:

```
:System.Cooordination_Operation   = 'NEXT_RECORD'
:System.Master_Block              = 'C'
```

Note: The values of these two system variables are well-defined only in the scope of an On-Clear-Details trigger, or any program unit called by that trigger. Outside this narrow context, the values of these two variables are undefined and should not be used.

The possible values of SYSTEM.COORDINATION_OPERATION, when it is appropriate to check that variable, are described in the following table.

Value	*Description*	*Caused By*
MOUSE	Mouse to non-current record	Mouse

UP	Move up a record	Menu, key, PL/SQL
DOWN	Move down a record	Menu, key, PL/SQL
SCROLL_UP	Scroll up records	Menu, key, PL/SQL
SCROLL_DOWN	Scroll down records	Mouse, key, PL/SQL
CLEAR_BLOCK	Clear current block	Menu, key, PL/SQL
CLEAR_RECORD	Clear current record	Menu, key, PL/SQL
CREATE_RECORD	Create new record	Mouse, menu, key, PL/SQL
DELETE_RECORD	Delete current record	Menu, key, PL/SQL
DUPLICATE_RECORD	Duplicate current record	Menu, key, PL/SQL
FIRST_RECORD	Move to first record	PL/SQL
LAST_RECORD	Move to last record	PL/SQL
NEXT_RECORD	Move to next record	Mouse, menu, key, PL/SQL
PREVIOUS_RECORD	Move to previous record	Mouse, menu, key, PL/SQL
GO_RECORD	Jump to record by number	PL/SQL
ENTER_QUERY	Enter Query mode	Menu, key, PL/SQL
EXECUTE_QUERY	Execute query	Menu, key, PL/SQL
COUNT_QUERY	Count queried records	Menu, key, PL/SQL
NEXT_SET	Fetch next set of records	Menu, key, PL/SQL
SYNCHRONIZE_ BLOCKS	Resume after commit error	Internal only

SYSTEM.CURRENT_BLOCK system variable

Syntax

```
SYSTEM.CURRENT_BLOCK
```

Description

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 Form Builder 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.

SYSTEM.CURRENT_DATETIME system variable

Syntax

```
SYSTEM.CURRENT_DATETIME
```

Description

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:MM:SS
```

Default

current date

Usage Notes

SYSTEM.CURRENT_DATETIME is useful when you want to use the current operating system date and time in a PL/SQL trigger or procedure. By using SYSTEM.CURRENT_DATETIME instead of $$DBDATETIME$$, you can avoid the performance impact caused by querying the database.

Note: Local time and database time may differ.

SYSTEM.CURRENT_DATETIME examples

```
/*
**
** trigger:  WHEN-TIMER-EXPIRED
** Example:  Update on-screen time every 30 seconds
*/
DECLARE
   time VARCHAR2(20);
BEGIN
   time := :System.Current_Datetime;
   :control.onscreen := SUBSTR(time, instr(time,' ')+1);
END;
```

SYSTEM.CURRENT_FORM system variable

Syntax

```
SYSTEM.CURRENT_FORM
```

Description

SYSTEM.CURRENT_FORM represents the name of the form that Form Builder is executing. The value is always a character string.

Usage Notes

You can use the GET_APPLICATION_PROPERTY built-in to obtain the name of the current form.

SYSTEM.CURRENT_FORM examples

Assume that you want any called form to be able to identify the name of the form that called it. You can invoke the following user-defined procedure before Form Builder issues a call. This procedure stores the name of the current form in a global variable named CALLING_FORM.

```
PROCEDURE STORE_FORMNAME IS
BEGIN
   :GLOBAL.Calling_Form := :System.Current_Form;
END;
```

SYSTEM.CURRENT_ITEM system variable

Syntax

```
SYSTEM.CURRENT_ITEM
```

Description

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 Form Builder 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.

SYSTEM.CURRENT_VALUE system variable

Syntax

```
SYSTEM.CURRENT_VALUE
```

Description

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.

SYSTEM.CURSOR_BLOCK system variable

Syntax

```
SYSTEM.CURSOR_BLOCK
```

Description

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.

SYSTEM.CURSOR_BLOCK examples

Assume that you want to create a Key-NXTBLK trigger at the form level that navigates depending on what the current block is. The following trigger performs this function, using :SYSTEM.CURSOR_BLOCK stored in a local variable.

```
DECLARE
   curblk VARCHAR2(30);
BEGIN
   curblk := :System.Cursor_Block;
   IF curblk = 'ORDERS'
      THEN Go_Block('ITEMS');
   ELSIF curblk = 'ITEMS'
      THEN Go_Block('CUSTOMERS');
   ELSIF curblk = 'CUSTOMERS'
      THEN Go_Block('ORDERS');
   END IF;
END;
```

SYSTEM.CURSOR_ITEM system variable

Syntax

```
SYSTEM.CURSOR_ITEM
```

Description

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.

Usage Notes

Within a given trigger, the value of SYSTEM.CURSOR_ITEM changes when navigation takes place. This differs from SYSTEM.TRIGGER_ITEM, which remains the same from the beginning to the end of single trigger.

SYSTEM.CURSOR_ITEM restrictions

Avoid using SYSTEM.CURSOR_ITEM in triggers where the current navigation unit is not the item, such as Pre- and Post-Record, Block, and Form triggers. In these triggers, the value of SYSTEM.CURSOR_ITEM is NULL.

SYSTEM.CURSOR_ITEM examples

Assume that you want to create a user-defined procedure that takes the value of the item where the cursor is located (represented by SYSTEM.CURSOR_VALUE), then multiplies the value by a constant, and then reads the modified value into the same item. The following user-defined procedure uses the COPY built-in to perform this function.

```
PROCEDURE CALC_VALUE IS
   new_value NUMBER;
BEGIN
   new_value := TO_NUMBER(:System.Cursor_Value) * .06;
   Copy(TO_CHAR(new_value), :System.Cursor_Item);
END;
```

SYSTEM.CURSOR_RECORD system variable

Syntax

```
SYSTEM.CURSOR_RECORD
```

Description

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.

SYSTEM.CURSOR_RECORD examples

Assume that you want to redefine [Previous Item] on the first text item of the ITEMS block so that it navigates to the last text item of the ORDERS block if the current record is the first record. The following Key-PRV-ITEM trigger on the ITEMS.ORDERID text item performs this function.

```
IF :System.Cursor_Record = '1'
   THEN Go_Item('orders.total');
   ELSE Previous_Item;
END IF;
```

SYSTEM.CURSOR_VALUE system variable

Syntax

```
SYSTEM.CURSOR_VALUE
```

Description

SYSTEM.CURSOR_VALUE represents the value of the item where the cursor is located. The value is always a character string.

Usage Notes

Be aware that in triggers where the current navigation unit is *not* the item, such as Pre-Record , and Pre-Block triggers, SYSTEM.CURSOR_VALUE will contain the value of the item navigated *from*, rather than the value of the item navigated *to*.

SYSTEM.CURSOR_VALUE restrictions

- Avoid using SYSTEM.CURSOR_VALUE in Pre-Form and Post-Form triggers, where the value of SYSTEM.CURSOR_VALUE is NULL.

SYSTEM.CURSOR_VALUE examples

Assume that you want to create a user-defined procedure that takes the value of the item where the cursor is located, multiplies the value by a constant, and then reads the modified value into the same item. The following user-defined procedure uses the COPY built-in to perform this function.

```
PROCEDURE CALC_VALUE IS
   new_value NUMBER;
BEGIN
   new_value := TO_NUMBER(:System.Cursor_Value) * .06;
   Copy(TO_CHAR(new_value), :System.Cursor_Item);
END;
```

SYSTEM.CUSTOM_ITEM_EVENT system variable

Syntax

```
SYSTEM.CUSTOM_ITEM_EVENT
```

Description

SYSTEM.CUSTOM_ITEM_EVENT stores the name of the event fired by a VBX (in 16-bit Microsoft Windows) or ActiveX (in 32-bit Windows) control.

SYSTEM.CUSTOM_ITEM_EVENT examples

Checks to see if the SpinDown event was fired by the SpinButton VBX control before navigating to the previous item.

```
IF  :System.Custom_Item_Event = 'SpinDown' THEN
 :QTY := :QTY -1;
END IF;
```

SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS system variable

Syntax

```
SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS
```

Description

SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS stores the supplementary arguments for an event fired by a VBX (in 16-bit Microsoft Windows) or ActiveX (in 32-bit Windows) control.

SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS examples

Obtains the value of the 'Button' parameter that stores the value of a VBX control event, and passed the 'Button' value to the user-defined Move_Image subprogram.

```
DECLARE
 parmType  NUMBER;
 parmValue VARCHAR2(80);
BEGIN
 Get_Parameter_Attr(:System.Custom_Item_Event_Parameters,
                    'Button',parmType,parmValue);
 /*
 ** The value of the 'Button' parameter represents the
 ** direction to move an image.  The user-defined Move_Image
 ** subprogram moves an image two pixels in the direction
 ** specified by 'Button' parameter.
 */
 Move_Image(parmValue);
END;
```

SYSTEM.DATE_THRESHOLD system variable

Syntax

```
SYSTEM.DATE_THRESHOLD
```

Description

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 Form Builder 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.

Default

01:00 (Synchronization occurs after one minute of elapsed time.)

This does not mean that Form Builder polls the RDBMS once every minute. It means that whenever Form Builder needs to generate the value for the system variables $$DBDATE$$, $$DBDATETIME$$, $$DBTIME$$, or SYSTEM.EFFECTIVE_DATE, it updates the effective date by adding the amount of elapsed time (as measured by the local operating system) to the most previously queried RDBMS value.

If the amount of elapsed time exceeds the date threshold, then a new query is executed to retrieve the RDBMS time and the elapsed counter is reset.

Usage Notes

If a form never references the database date, Form Builder never executes a query to retrieve the RDBMS date, regardless of the value of SYSTEM.DATE_THRESHOLD.

The operating system clock and the RDBMS clock rarely drift by more than one or two seconds, even after hours of elapsed time. However, since your database administrator can reset the RDBMS clock at any time, it is safest to set the threshold no higher than a few minutes.

Often, a Form Builder block may contain multiple references to these $$DBDATE$$, $$DBDATETIME$$, or $$DBTIME$$ defaults. By setting SYSTEM.DATE_THRESHOLD to the default of one minute, nearly all such references in a form can be satisfied with a single query of the RDBMS.

SYSTEM.EFFECTIVE_DATE system variable

Syntax

```
SYSTEM.EFFECTIVE_DATE
```

Description

SYSTEM.EFFECTIVE_DATE represents the effective database date. The variable value must always be in the following format:

```
DD-MON-YYYY HH24:MI:SS
```

Default

RDBMS date

Usage Notes

This system variable is convenient for testing. Since you can use it to set a specific time, the time on the screen during an application would not cause subsequent test results to appear different than the known valid output.

SYSTEM.EFFECTIVE_DATE restrictions

This variable is only valid when the database contains a definition of the DUAL table.

SYSTEM.EFFECTIVE_DATE examples

Assume you have set a DATE or TIME text item to one of the three system variables $$DBDATE$$, $$DBDATETIME$$, or $$DBTIME$$. To override that date or time, set the SYSTEM.EFFECTIVE_DATE system variable to a specific date and/or time.

```
:System.Effective_Date := '31-DEC-1997 10:59:00'
```

Note that the effective date "rolls forward" with the database clock. For example, if you were to set the date as in the immediately preceding example, in an hour, the date would appear as follows:

```
31-DEC-1997 11:59:00
```

The value is synchronized to the RDBMS date. If your database administrator changes the RDBMS date, the SYSTEM.EFFECTIVE_DATE is automatically changed by the same amount of change between old and new RDBMS dates. Form Builder polls the RDBMS whenever a reference to the effective date is required by the application.

SYSTEM.EVENT_WINDOW system variable

Syntax

```
SYSTEM.EVENT_WINDOW
```

Description

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, assign the value of the variable to any of the following:

- global variable
- parameter
- variable
- item, including a null canvas item

SYSTEM.EVENT_WINDOW examples

The following example sets the input focus into a particular item, depending on the window affected:

```
IF :System.Event_Window = 'ROOT_WINDOW' THEN
   Go_Item('EMPNO');
ELSIF :System.Event_Window = 'DEPT_WINDOW' THEN
   Go_Item('DEPTNO');
END IF;
```

SYSTEM.FORM_STATUS system variable

Syntax

```
SYSTEM.FORM_STATUS
```

Description

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.

Usage Notes

Each time this value is referenced, it must be constructed by Form Builder. If a form contains a large number of blocks and many records, using SYSTEM.FORM_STATUS could affect performance.

SYSTEM.FORM_STATUS examples

Assume that you want to create a trigger that performs a commit before clearing a form if there are changes to commit within that form. The following Key-CLRFRM trigger performs this function.

```
IF :System.Form_Status = 'CHANGED'
   THEN Commit_Form;
END IF;
Clear_Form;
```

SYSTEM.LAST_FORM system variable

Syntax

```
SYSTEM.LAST_FORM
```

Description

SYSTEM.LAST_FORM represents the form document 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 document ID or NULL.

Usage Notes

SYSTEM.LAST_FORM is not valid with CALL_FORM.

SYSTEM.LAST_QUERY system variable

Syntax

```
SYSTEM.LAST_QUERY
```

Description

SYSTEM.LAST_QUERY represents the query SELECT statement that Form Builder most recently used to populate a block during the current Runform session. The value is always a character string.

SYSTEM.LAST_QUERY examples

Example 1

Assume that you want to generate a report in Report Builder that retrieves information identical to a query you perform in Form Builder. The following examples show how to use SYSTEM.LAST_QUERY to extract the WHERE/ORDER BY clause from the last query so you can pass the results to Report Builder using the RUN_PRODUCT built-in.

```
FUNCTION Last_Where_Clause
RETURN VARCHAR2
  IS
    tmp_lstqry VARCHAR2(10000) := :System.Last_Query;
    tmp_curblk VARCHAR2(40);
    tmp_index  NUMBER;
    tmp_where  VARCHAR2(2000);
  BEGIN
    /*
    ** See if we can find the word 'WHERE' in
    ** the text of the Last Query
    */
    tmp_index:= INSTR(tmp_lstqry,'WHERE');
    /*
    ** If we found it (at a positive index into
    ** the string), we extract the remainder of
    ** the text that follows the word 'WHERE' as
    ** the Where clause. This might include ORDER BY
    ** clause, too.
    */
    IF tmp_index > 0 THEN
      tmp_where := SUBSTR(tmp_lstqry, tmp_index + 6);
    END IF;
    RETURN (tmp_where);
EXCEPTION
  WHEN OTHERS THEN
    RETURN NULL;
END;
```

Example 2

```
PROCEDURE Run_Report_For_Last_Query
IS
   pl  ParamList;
```

```
   wc  VARCHAR2(2000); -- The Where Clause to Pass
BEGIN
   /*
   ** Create a parameter list for parameter passing
   */
   pl := Create_Parameter_List('tmp');
   /*
   ** Get the Where Clause from the Last Query
   ** using a user-defined function
   */
   wc := Last_Where_Clause;
   /*
   ** If there is a Non-NULL Last Where clause to
   ** pass, add a text parameter to the parameter
   ** list to specify the parameter name and its
   ** value. In this case the report definition has
   ** a parameter named 'the_Where_Clause' that
   ** it's expecting.
   */
   IF wc IS NOT NULL THEN
      Add_Parameter(pl,                  -- Handle to
                                         -- the ParamList
            'the_Where_Clause',          -- Name of Parameter
                                         -- in the Report
            TEXT_PARAMETER,              -- Type of Parameter
            wc                           -- String Value
                                         -- for Parameter
            );
   END IF;
   /*
   ** Launch the report, passing parameters in the
   ** parameter list.
   */
   Run_Product(REPORTS,              -- The Product to call
               'rep0058.rdf',        -- The name of the
                                     -- report definition
               SYNCHRONOUS,          -- The communications mode
               BATCH,                -- The Execution Mode
               FILESYSTEM,           -- The Location of the
                                     -- reports document
               pl                    -- The Handle to the
      );                                -- parameter list
   /*   Delete the parameter list  */
   Destroy_Parameter_List(pl);
END;
```

SYSTEM.LAST_RECORD system variable

Syntax

```
SYSTEM.LAST_RECORD
```

Description

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.

SYSTEM.LAST_RECORD examples

Assume that you want to create a user-defined procedure that displays a custom message when an operator navigates to the last record in a block's list of records. The following user-defined procedure performs the basic function.

```
PROCEDURE LAST_RECORD_MESSAGE IS
BEGIN
   IF :System.Last_Record = 'TRUE'
      THEN Message('You are on the last row');
   END IF;
END;
```

You can then redefine [Down], [Next Record], and [Scroll Down] to call this user-defined procedure in addition to their normal processing.

SYSTEM.MASTER_BLOCK system variable

Syntax

```
SYSTEM.MASTER_BLOCK
```

Description

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.

Please see the reference topic for SYSTEM.COORDINATION_OPERATION for more information.

SYSTEM.MESSAGE_LEVEL system variable

Syntax

```
SYSTEM.MESSAGE_LEVEL
```

Description

SYSTEM.MESSAGE_LEVEL stores one of the following message severity levels: 0, 5, 10, 15, 20, or 25. The default value is 0.

SYSTEM.MESSAGE_LEVEL can be set to either a character string or a number. The values assigned can be any value between 0 and 25, but values lower than 0 or higher than 25 will generate an error.

During a Runform session, Form Builder 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. Form Builder does not suppress prompts or vital error messages, no matter what severity level you select.

SYSTEM.MESSAGE_LEVEL examples

Assume that you want Form Builder to display only the most severe messages (level 25). The following Pre-Form trigger suppresses all messages at levels 20 and below.

```
:System.Message_Level := '20';
```

SYSTEM.MODE system variable

Syntax

```
SYSTEM.MODE
```

Description

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.

Usage Notes

When using SYSTEM.MODE to check whether the current block is in Enter Query mode, be aware that if testing from a When-Button-Pressed trigger in a control block, Enter Query mode will never be entered, because the control block is not the current block.

SYSTEM.MODE examples

Assume that you want Form Builder to display an LOV when the operator enters query mode and the input focus is in a particular text item. The following trigger accomplishes that operation.

```
/*
** When-New-Item-Instance trigger
*/
BEGIN
   IF :System.Cursor_Item = 'EMP.EMPNO' and
     :System.Mode = 'ENTER-QUERY'
   THEN
      IF NOT Show_Lov('my_lov') THEN
         RAISE Form_trigger_Failure;
   END IF;
END;
```

SYSTEM.MOUSE_BUTTON_MODIFIERS system variable

Syntax

```
SYSTEM.MOUSE_BUTTON_MODIFIERS
```

Description

SYSTEM.MOUSE_BUTTON_MODIFIERS indicates the keys that were pressed during the click, such as SHIFT, ALT, or CONTROL. The value is always a character string.

For example, if the operator holds down the control and shift keys while pressing the mouse button, SYSTEM.MOUSE_BUTTON_MODIFIERS contains the value "Shift+Control+".

The values returned by this variable will be invariant across all platforms, and will not change across languages. SYSTEM.MOUSE_BUTTON_MODIFIERS should be used in place of SYSTEM.MOUSE_BUTTON_SHIFT_STATE.

Possible values are: "Shift+", "Caps Lock+", "Control+", "Alt+", "Command+", "Super+", and "Hyper+".

SYSTEM.MOUSE_BUTTON_PRESSED system variable

Syntax

```
SYSTEM.MOUSE_BUTTON_PRESSED
```

Description

SYSTEM.MOUSE_BUTTON_PRESSED indicates the number of the button that was clicked, either 1, 2, or 3 (left, middle, or right). The value is always a character string.

Notes:

On Motif platforms pressing the right mouse button will not set the SYSTEM.MOUSE_BUTTON_PRESSED value.

SYSTEM.MOUSE_BUTTON_PRESSED examples

```
/*
**  trigger:  When-Mouse-Click
**  Example:  When mouse button 1 is pressed,
**            a help window appears.
*/
DECLARE
   the_button_pressed  VARCHAR(1);
BEGIN
   the_button_pressed := :System.Mouse_Button_Pressed;
   IF the_button_pressed = '1' THEN
      Show_Window('online_help',20,5);
   END IF;
END;
```

SYSTEM.MOUSE_BUTTON_SHIFT_STATE system variable

Syntax

```
SYSTEM.MOUSE_BUTTON_SHIFT_STATE
```

Description

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. The string itself may depend on the user's platform. For example, in Microsoft Windows, the strings returned are in the language of the operating system.

Key Pressed	*Value*
SHIFT	Shift+
CONTROL	Ctrl+
ALT	Alt+
SHIFT+CONTROL	Shift+Ctrl+

SYSTEM.MOUSE_BUTTON_SHIFT_STATE examples

```
/*
**  trigger:  When-Mouse-Click
**  Example:  If the operator presses down on the Shift key and
**            then clicks on a boilerplate image, a window
**            appears.
*/
DECLARE
   key_pressed  VARCHAR(30) := 'FALSE';
   x_position_clicked NUMBER(30);
   y_position_clicked NUMBER(30);

BEGIN
   key_pressed := :System.Mouse_Button_Shift_State;
   x_position_clicked := To_Number(:System.Mouse_X_Pos);
   y_position_clicked := To_Number(:System.Mouse_Y_Pos);
/*
**  If the operator shift-clicked within the x and y
**  coordinates of a boilerplate image, display a window.
*/
   IF  key_pressed = 'Shift+' AND x_position_clicked
      BETWEEN 10 AND 20 AND y_position_clicked BETWEEN 10
      AND 20 THEN
      Show_Window('boilerplate_image_window');
   END IF;
END;
```

SYSTEM.MOUSE_CANVAS system variable

Syntax

```
SYSTEM.MOUSE_CANVAS
```

Description

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 operator presses the left mouse button, then moves the mouse
- the platform is non-GUI

SYSTEM.MOUSE_CANVAS examples

```
/*
**  trigger:  When-Mouse-Move
**  Example:  When the mouse enters any one of several
overlapping
**            canvases, Form Builder brings that canvas to the
**            front.
*/
DECLARE
   canvas_to_front  VARCHAR(50);
BEGIN
   canvas_to_front := :System.Mouse_Canvas;
   Show_View(canvas_to_front);
END;
```

SYSTEM.MOUSE_FORM system variable

Syntax

```
SYSTEM.MOUSE_FORM
```

Description

If the mouse is in a form document, SYSTEM.MOUSE_FORM represents the name of that form document 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 system variable

Syntax

```
SYSTEM.MOUSE_ITEM
```

Description

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 operator presses the left mouse button, then moves the mouse
- the platform is not a GUI platform

SYSTEM.MOUSE_ITEM examples

```
/* trigger:  When-Mouse-Click
**   Example:  Dynamically repositions an item if:
**             1) the operator clicks mouse button 2
**                on an item and
**             2) the operator subsequently clicks mouse button
**                2 on an area of the canvas that is
**                not directly on top of another item.
*/
DECLARE
   item_to_move       VARCHAR(50);
   the_button_pressed VARCHAR(50);
   target_x_position  NUMBER(3);
   target_y_position  NUMBER(3);
   the_button_pressed VARCHAR(1);
BEGIN
   /* Get the name of the item that was clicked.
   */
   item_to_move := :System.Mouse_Item;
   the_button_pressed := :System.Mouse_Button_Pressed;
   /*
   **  If the mouse was clicked on an area of a canvas that is
   **  not directly on top of another item, move the item to
   **  the new mouse location.
   */
   IF item_to_move IS NOT NULL AND the_button_pressed =  '2'
THEN
      target_x_position := To_Number(:System.Mouse_X_Pos);
      target_y_position := To_Number(:System.Mouse_Y_Pos);
      Set_Item_Property(item_to_move,position,
          target_x_position,target_y_position);
      target_x_position := NULL;
      target_y_position := NULL;
      item_to_move := NULL;
   END IF;
```

```
END;
```

SYSTEM.MOUSE_RECORD system variable

Syntax

```
SYSTEM.MOUSE_RECORD
```

Description

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).

SYSTEM.MOUSE_RECORD examples

```
/*
**  trigger:  When-Mouse-Move
**  Example:  If the mouse is within a record, display a window
**            that contains an editing toolbar.
*/
DECLARE
   mouse_in_record  NUMBER(7);
BEGIN
   mouse_in_record := To_Number(:System.Mouse_Record);

   IF mouse_in_record > 0 THEN
      Show_Window('editing_toolbar');
   END IF;
END;
```

SYSTEM.MOUSE_RECORD_OFFSET system variable

Syntax

```
SYSTEM.MOUSE_RECORD_OFFSET
```

Description

If the mouse is in a record, SYSTEM.MOUSE_RECORD_OFFSET represents the offset from the first visible record as a CHAR value. SYSTEM.MOUSE_RECORD_OFFSET is only valid within mouse triggers. Its value represents which row within the visible rows the mouse has clicked.

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).

SYSTEM.MOUSE_X_POS system variable

Syntax

```
SYSTEM.MOUSE_X_POS
```

Description

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.

SYSTEM.MOUSE_X_POS examples

```
/*
**  Example:  See SYSTEM.MOUSE_ITEM  and
**            SYSTEM.MOUSE_BUTTON_SHIFT_STATE.
*/
```

SYSTEM.MOUSE_Y_POS system variable

Syntax

```
SYSTEM.MOUSE_Y_POS
```

Description

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.

SYSTEM.MOUSE_Y_POS examples

```
/*
**  Example:  See SYSTEM.MOUSE_ITEM and
**            SYSTEM.MOUSE_BUTTON_SHIFT_STATE.
*/
```

SYSTEM.RECORD_STATUS system variable

Syntax

```
SYSTEM.RECORD_STATUS
```

Description

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.

Usage Notes

Both SYSTEM.RECORD_STATUS and the GET_RECORD_PROPERTY built-in return the status of a record in a given block, and in most cases, they return the same status. However, there are specific cases in which the results may differ.

SYSTEM.RECORD_STATUS can in certain cases return a value of NULL, because SYSTEM.RECORD_STATUS is undefined when there is no current record in the system. For example, in a When-Clear-Block trigger, Form Builder is at the block level in its processing sequence, so there is no current record to report on, and the value of SYSTEM.RECORD_STATUS is NULL.

GET_RECORD_PROPERTY, on the other hand, always has a value of NEW, CHANGED, QUERY, or INSERT, because it returns the status of a specific record without regard to the processing sequence or whether the record is the current record.

SYSTEM.RECORD_STATUS examples

Assume that you want to create a trigger that performs a commit before clearing a Changed record. The following Key-CLRREC trigger performs this function.

```
IF :System.Record_Status IN ('CHANGED', 'INSERT') THEN
   Commit_Form;
END IF;
Clear_Record;
```

SYSTEM.SUPPRESS_WORKING system variable

Syntax

```
SYSTEM.SUPPRESS_WORKING
```

Description

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 Form Builder from issuing the "Working..." message.
FALSE	Allows Form Builder to continue to issue the "Working..." message.

SYSTEM.SUPPRESS_WORKING examples

Assume that you want to have the form filled with data when the operator enters the form. The following When-New-Form-Instance trigger will prevent the unwanted updates that would normally occur when you fill the blocks with data.

```
:System.Suppress_Working := 'TRUE';
Go_Block ('DEPT');
Execute_Query;
Go_Block ('EMP');
Execute_Query;
Go_Block ('DEPT');
:System.Suppress_Working := 'FALSE';
```

SYSTEM.TAB_NEW_PAGE system variable

Syntax

```
SYSTEM.TAB_NEW_PAGE
```

Description

The system variable SYSTEM.TAB_NEW_PAGE specifies the name of the tab page to which navigation occurred. Use it inside a When-Tab-Page-Changed trigger.

SYSTEM.TAB_NEW_PAGE examples

```
/* Use system variable SYSTEM.TAB_NEW_PAGE inside a
** When-Tab-Page-Changed trigger to change the label of
** the tab page to UPPERCASE when an end user navigates
** into the tab page:
*/
DECLARE
  tp_nm   VARCHAR2(30);
  tp_id   TAB_PAGE;
  tp_lbl  VARCHAR2(30);
BEGIN
  tp_nm  := :SYSTEM.TAB_NEW_PAGE;
  tp_id  := FIND_TAB_PAGE(tp_nm);
  tp_lbl := GET_TAB_PAGE_PROPERTY(tp_id, label);
  IF tp_nm LIKE 'ORD%' THEN
    SET_TAB_PAGE_PROPERTY(tp_id, label, 'ORDERS');
  END IF;
END;
```

SYSTEM.TAB_PREVIOUS_PAGE system variable

Syntax

```
SYSTEM.TAB_PREVIOUS_PAGE
```

Description

The system variable SYSTEM.TAB_PREVIOUS_PAGE specifies the name of the tab page from which navigation occurred. Use it inside a When-Tab-Page-Changed trigger.

SYSTEM.TAB_PREVIOUS_PAGE examples

```
/* Use system variable SYSTEM.TAB_PREVIOUS_PAGE inside a
** When-Tab-Page-Changed trigger to change the label of the
** tab page to initial-cap after an end user navigates out
** of the tab page:
*/
DECLARE
  tp_nm   VARCHAR2(30);
  tp_id   TAB_PAGE;
  tp_lbl  VARCHAR2(30);
BEGIN
  tp_nm  := :SYSTEM.TAB_PREVIOUS_PAGE;
  tp_id  := FIND_TAB_PAGE(tp_nm);
  tp_lbl := GET_TAB_PAGE_PROPERTY(tp_id, label);
  IF tp_nm LIKE 'ORD%' THEN
    SET_TAB_PAGE_PROPERTY(tp_id, label, 'Orders');
  END IF;
END;
```

SYSTEM.TRIGGER_BLOCK system variable

Syntax

```
SYSTEM.TRIGGER_BLOCK
```

Description

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.

SYSTEM.TRIGGER_BLOCK examples

Assume that you want to write a form-level procedure that navigates to the block where the cursor was when the current trigger initially fired. The following statement performs this function.

```
Go_Block(Name_In('System.trigger_Block'));
```

SYSTEM.TRIGGER_ITEM system variable

Syntax

```
SYSTEM.TRIGGER_ITEM
```

Description

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.

Usage Notes

SYSTEM.TRIGGER_ITEM remains the same from the beginning to the end of given trigger. This differs from SYSTEM.CURSOR_ITEM, which may change within a given trigger when navigation takes place.

SYSTEM.TRIGGER_ITEM restrictions

Avoid using SYSTEM.TRIGGER_ITEM in triggers where the current navigation unit is *not* the item, such as Pre- and Post-Record, Block, and Form triggers. In these triggers, the value of SYSTEM.TRIGGER_ITEM is NULL.

SYSTEM.TRIGGER_ITEM examples

Assume that you want to write a user-defined procedure that navigates to the item where the cursor was when the current trigger initially fired. The following statement performs this function.

```
Go_Item(:System.trigger_Item);
```

SYSTEM.TRIGGER_NODE_SELECTED system variable

Syntax

```
SYSTEM.TRIGGER_NODE_SELECTED
```

Description

SYSTEM.TRIGGER_NODE_SELECTED contains a valid value only during the WHEN-TREE-NODE-SELECTED trigger, indicating whether the trigger is firing for a selection or a deselection. The values are either TRUE or FALSE.

SYSTEM.TRIGGER_RECORD system variable

Syntax

```
SYSTEM.TRIGGER_RECORD
```

Description

SYSTEM.TRIGGER_RECORD represents the number of the record that Form Builder 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.

SYSTEM.TRIGGER_RECORD examples

In the following anonymous block, the IF statement uses SYSTEM.TRIGGER_RECORD to identify the current record before processing continues.

```
IF :System.trigger_Record = '1'
   THEN Message('First item in this order.');
END IF;
```

Triggers

Overview of trigger categories

This topic provides an overview of commonly used triggers, grouped into the following functional categories:

- block-processing triggers
- interface event triggers
- master-detail triggers
- message-handling triggers
- navigational triggers
- query-time triggers
- transactional triggers
- validation triggers

This topic does not list all of the triggers available in Form Builder, many of which have highly specialized uses. For a complete list of triggers, or for detailed information on a specific trigger, refer to the Triggers Category Page.

Block processing triggers

Block processing triggers fire in response to events related to record management in a block.

Trigger	***Typical Usage***
When-Create-Record	Perform an action whenever Form Builder attempts to create a new record in a block. For example, to set complex, calculated, or data-driven default values that must be specified at runtime, rather than design time.
When-Clear-Block	Perform an action whenever Form Builder flushes the current block; that is, removes all records from the block.
When-Database-Record	Perform an action whenever Form Builder changes a record's status to Insert or Update, thus indicating that the record should be processed by the next COMMIT_FORM operation
When-Remove-Record	Perform an action whenever a record is cleared or deleted. For example, to adjust a running total that is being calculated for all of the records displayed

in a block.

Interface event triggers

Interface event triggers fire in response to events that occur in the form interface. Some of these triggers, such as When-Button-Pressed, fire only in response to operator input or manipulation. Others, like When-Window-Activated, can fire in response to both operator input and programmatic control.

trigger	*Typical Usage*
When-Button-Pressed	Initiate an action when an operator selects a button, either with the mouse or through keyboard selection.
When-Checkbox-Changed	Initiate an action when the operator toggles the state of a check box, either with the mouse or through keyboard selection
When-Image-Activated	Initiate an action whenever the operator double-clicks an image item.
When-Image-Pressed	Initiate an action whenever an operator clicks on an image item.
Key- [all]	Replace the default function associated with a function key. For example, you can define a Key-EXIT trigger to replace the default functionality of the [Help] key.
When-Radio-Changed	Initiate an action when an operator changes the current radio button selected in a radio group item.
When-Timer-Expired	Initiate an action when a programmatic timer expires.
When-Window-Activated	Initiate an action whenever an operator or the application activates a window.
When-Window-Closed	Initiate an action whenever an operator closes a window with the window manager's Close command.
When-Window-Deactivated	Initiate an action whenever a window is deactivated as a result of another window becoming the active window.
When-Window-Resized	Initiate an action whenever a window is resized, either by the operator or programmatically.

Master/Detail triggers

Form Builder generates master/detail triggers automatically when a master/detail relation is defined between blocks. The default master/detail triggers enforce coordination between records in a detail block and the master record in a master block. Unless developing custom block-coordination schemes, you do not need to define these triggers. Instead, simply create a relation object, and let Form Builder generate the triggers required to manage coordination between the master and detail blocks in the relation.

trigger	*Typical Usage*
On-Check-Delete-Master	Fires when Form Builder attempts to delete a record in a block that is a master block in a master/detail relation.
On-Clear-Details	Fires when Form Builder needs to clear records in a block that is a detail block in a master/detail relation because those records no longer correspond to the current record in the master block.
On-Populate-Details	Fires when Form Builder needs to fetch records into a block that is the detail block in a master/detail relation so that detail records are synchronized with the current record in the master block.

Message-handling triggers

Form Builder automatically issues appropriate error and informational messages in response to runtime events. Message handling triggers fire in response to these default messaging events.

trigger	*Typical Usage*
On-Error	Replace a default error message with a custom error message, or to trap and recover from an error.
On-Message	To trap and respond to a message; for example, to replace a default message issued by Form Builder with a custom message.

Navigational triggers

Navigational triggers fire in response to navigational events. For instance, when the operator clicks on a text item in another block, navigational events occur as Form Builder moves the input focus from the current item to the target item.

Navigational events occur at different levels of the Form Builder object hierarchy (Form, Block, Record, Item). Navigational triggers can be further sub-divided into two categories: Pre- and Post-triggers, and When-New-Instance triggers.

Pre- and Post- Triggers fire as Form Builder navigates internally through different levels of the object hierarchy. As you might expect, these triggers fire in response to navigation initiated by an operator, such as pressing the [Next Item] key. However, be aware that these triggers also fire in response to internal navigation that Form Builder performs during default processing. To avoid unexpected results, you must consider such internal navigation when you use these triggers.

trigger	*Typical Usage*
Pre-Form	Perform an action just before Form Builder navigates to the form from "outside" the form, such as at form startup.
Pre-Block	Perform an action before Form Builder navigates to the block level from the form level.
Pre-Record	Perform an action before Form Builder navigates to the record level from the block level.
Pre-Text-Item	Perform an action before Form Builder navigates to a text item from the record level.
Post-Text-Item	Manipulate an item when Form Builder leaves a text item and navigates to the record level.
Post-Record	Manipulate a record when Form Builder leaves a record and navigates to the block level.
Post-Block	Manipulate the current record when Form Builder leaves a block and navigates to the form level.
Post-Form	Perform an action before Form Builder navigates to "outside" the form, such as when exiting the form.

When-New-Instance-Triggers fire at the end of a navigational sequence that places the input focus on a different item. Specifically, these triggers fire just after Form Builder moves the input focus to a different item, when the form returns to a quiet state to wait for operator input.

Unlike the Pre- and Post- navigational triggers, the When-New-Instance triggers do not fire in response to internal navigational events that occur during default form processing.

trigger	*Typical Usage*
When-New-Form-Instance	Perform an action at form start-up. (Occurs after the Pre-Form trigger fires).
When-New-Block-Instance	Perform an action immediately after the input focus moves to an item in a block other than the block that previously had input focus.
When-New-Record-Instance	Perform an action immediately after the input focus moves to an item in a different record. If the new

	record is in a different block, fires after the When-New-Block-Instance trigger, but before the When-New-Item-Instance trigger.
When-New-Item-Instance	Perform an action immediately after the input focus moves to a different item. If the new item is in a different block, fires after the When-New-Block-Instance trigger.

Query-time triggers

Query-time triggers fire just before and just after the operator or the application executes a query in a block.

trigger	*Typical Usage*
Pre-Query	Validate the current query criteria or provide additional query criteria programmatically, just before sending the SELECT statement to the database.
Post-Query	Perform an action after fetching a record, such as looking up values in other tables based on a value in the current record. Fires once for each record fetched into the block.

Transactional triggers

Transactional triggers fire in response to a wide variety of events that occur as a form interacts with the data source.

trigger	*Typical Usage*
On-Delete	Replace the default Form Builder processing for handling deleted records during transaction posting.
On-Insert	Replace the default Form Builder processing for handling inserted records during transaction posting.
On-Lock	Replace the default Form Builder processing for locking rows in the database.
On-Logon	Replace the default Form Builder processing for connecting to ORACLE, especially for a form that does not require a database connection or for connecting to a non-ORACLE data source.

On-Logout	Replace the default Form Builder processing for logout from ORACLE.
On-Update	Replace the default Form Builder processing for handling updated records during transaction posting.
Post-Database-Commit	Augment default Form Builder processing following a database commit.
Post-Delete	Audit transactions following the deletion of a row from the database.
Post-Forms-Commit	Augment the default Form Builder commit statement prior to committing a transaction.
Post-Insert	Audit transactions following the insertion of a row in the database.
Post-Update	Audit transactions following the updating of a row in the database.
Pre-Commit	Perform an action immediately before the Post and Commit Transactions process, when Form Builder determines that there are changes in the form to post or to commit.
Pre-Delete	Manipulate a record prior to its being deleted from the database during the default Post and Commit Transactions process; for example, to prevent deletion of the record if certain conditions exist.
Pre-Insert	Manipulate a record prior to its being inserted in the database during the default Post and Commit Transactions process.
Pre-Update	Validate or modify a record prior to its being updated in the database during the default Post and Commit Transactions process.

Note: This is only a partial list of the transactional triggers available. Many of the triggers not shown here are On-event triggers that exist primarily for applications that will run against a non-ORACLE data source.

Validation triggers

Validation triggers fire when Form Builder validates data in an item or record. Form Builder performs validation checks during navigation that occurs in response to operator input, programmatic control, or default processing, such as a Commit operation.

trigger	***Typical Usage***

When-Validate-Item	Augment default validation of an item.
When-Validate-Record	Augment default validation of a record.

Other trigger categories

The previous section listed triggers in groups according to their functions. Triggers can also be categorized by name. There are five such categories, each relating to a particular type of event that occurs during runtime processing.

When-Event Triggers signal a point at which Form Builder default processing may be *augmented* with additional tasks or operations. For example, the When-Validate-Item trigger fires immediately after Form Builder validates data in an item.

To augment the default validation checks that Form Builder performs, code additional validation in a When-Validate-Item trigger. Most When-event triggers can include calls to restricted built-in subprograms.

On-Event Triggers signal a point at which Form Builder default processing may be *replaced*. For example, the On-Logon trigger fires when Form Builder readies to log on to an ORACLE data source. If the application requires a connection to a non-ORACLE data source, code an On-Logon trigger to pass the appropriate logon parameters to the non-ORACLE data source. This completely replaces the default logon to ORACLE. On-event triggers can include calls to unrestricted built-in subprograms.

Pre-Event Triggers signal a point just prior to the occurrence of either a When-event or an On-event. Use triggers for these events to prepare objects or data for the upcoming event. Pre-event triggers can include calls to unrestricted built-in subprograms.

Post-Event Triggers signal a point just following the occurrence of either a When-event or an On-event. Write triggers for these events that validate objects or data, or that perform some auditing tasks based on the prior event. Post-event triggers can include calls to unrestricted built-in subprograms.

Key Triggers have a one-to-one relationship with specific keys. That is, the trigger fires when the operator presses a specific key or key-sequence.

Remember that most GUI applications offer operators more than one way to execute commands. For instance, an operator might be able to execute a query by clicking a button, selecting a menu command, or pressing the [Execute Query] key.

In such situations, it would be a mistake to place all of your application logic in a key trigger that might never fire. Similarly, in any mouse-driven application, you cannot rely entirely on key triggers for navigational keys like [Next Item] and [Next Block]. Because operators can navigate with a mouse, they may choose not to use these keys for navigation, and the associated triggers would not be fired.

Delete-Procedure trigger

Description

Automatically created by Form Builder when the delete data source is a stored procedure. This trigger is called when a delete operation is necessary. Think of this as an ON-DELETE trigger that is called by the system instead of doing default delete operations.

Do not modify this trigger.

Enter Query Mode Not applicable.

On Failure

No effect

Function Key triggers

Description

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 defined in a function key trigger replace the default action that the function key would normally perform.

The following table shows all function key triggers and the corresponding Runform function keys.

Key trigger	*Associated Function Key*
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 Form Builder.

[Clear Item]	[First Line]	[Scroll Left]
[Copy]	[Insert Line]	[Scroll Right]
[Cut]	[Last Line]	[Search]
[Delete Character]	[Left]	[Select]
[Delete Line]	[Paste]	[Show Keys]
[Display Error]	[Refresh]	[Toggle Insert/Replace]
[End of Line]	[Right]	[Transmit]

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

On Failure

no effect

Enter Query Mode yes

Usage Notes

The default functionality performed by the following keys is not allowed in Enter Query mode:

[Clear Block]	[Down]	[Next Record]
[Clear Form]	[Duplicate Item]	[Next Set of Records]
[Clear Record]	[Duplicate Record]	[Previous Block]
[Accept]	[Block Menu]	[Previous Record]
[Insert Record]	[Next Block]	[Up]
[Delete Record]	[Next Primary Key]	[Lock Record]

Common Uses

Use function key triggers to perform the following tasks:

- Disable function keys dynamically.
- Replace the default behavior of function keys.
- Dynamically remap function keys.
- Perform complex or multiple functions with a single key or key sequence.

Function Key Triggers restrictions

- Form Builder ignores the Key-Commit trigger when an operator presses [Commit/Accept] in a dialog box.

Insert-Procedure trigger

Description

Automatically created by Form Builder when the insert data source is a stored procedure. This trigger is called when a insert operation is necessary. Think of this as an ON-INSERT trigger that is called by the system instead of doing default insert operations.

Definition Level

Do not modify this trigger.

Enter Query Mode Not applicable.

On Failure

No effect

Key-Fn trigger

Description

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 Form Builder 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.

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

- Use Key-Fn triggers to create additional function keys for custom functions.
- The key description shown in the default menu's **Help->Keys** menu will always be for the form-level trigger defined for that key. If there are any lower-level triggers (e.g., block-level triggers) that are also defined for the key, their descriptions will be shown when focus is in the lower level (e.g., the block) and [Show Keys] is pressed, but they will not be displayed in the default menu's **Help->Keys** menu.
- Not all keys can be remapped on certain operating systems. For example, the Microsoft Windows operating system always displays the Windows Help System when F1 is pressed, and attempts to close the application window when Alt-F4 is pressed.

Key-Fn trigger restrictions

Form Builder ignores Key-Fn triggers in Edit mode.

Key-Others trigger

Description

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, Form Builder still displays the function key's default entry in the Keys screen.

Trigger Type key

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use Key-Others triggers to limit an operator's possible actions. Specifically, use Key-Others triggers to perform the following tasks:

- Disable all keys that are not relevant in a particular situation.
- Perform one specific action whenever an operator presses any key.

Also note:

- The key description shown in the default menu's **Help->Keys** menu will always be for the form-level trigger defined for that key. If there are any lower-level triggers (e.g., block-level triggers) that are also defined for the key, their descriptions will be shown when focus is in the lower level (e.g., the block) and [Show Keys] is pressed, but they will not be displayed in the default menu's **Help->Keys** menu.

Key-Others trigger restrictions

Form Builder ignores a Key-Others trigger under the following conditions:

- The form is in Enter Query mode and Fire in Enter-Query Mode is Off.
- A list of values, the Keys screen, a help screen, or an error screen is displayed.
- The operator is responding to a Runform prompt.

- The operator presses a static function key.

Lock-Procedure trigger

Description

Automatically created by Form Builder when the lock data source is a stored procedure. This trigger is called when a lock operation is necessary. Think of this as an ON-LOCK trigger that is called by the system instead of doing default lock operations.

Do not modify this trigger.

Enter Query Mode Not applicable.

On Failure

No effect

On-Check-Delete-Master trigger

Description

Form Builder creates this trigger automatically when you define a master/detail relation and set the Delete Record Behavior property to Non-Isolated. It fires when there is an attempt to delete a record in the master block of a master/detail relation.

Definition Level form or block

Legal Commands

Any command, unrestricted built-ins, restricted built-ins

Enter Query Mode no

On Failure

Prevents the deletion of the current master record

Fires In

Master/Detail Coordination

On-Check-Delete-Master trigger examples

Example

The following example replaces the On-Check-Delete-Master that is generated by default for a master/detail relation with a trigger that will fail if the sum of the distributions does not equal the purchase order total.

```
DECLARE
  the_sum NUMBER;
BEGIN
  SELECT SUM(dollar_amt)
    INTO the_sum
    FROM po_distribution
   WHERE po_number = :purchase_order.number;

/* Check for errors */
  IF the_sum <> :purchase_order.total THEN
    Message('PO Distributions do not reconcile.');
    RAISE Form_trigger_Failure;
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  end if;
END;
```

On-Check-Unique trigger

Description

During a commit operation, the On-Check-Unique trigger fires when Form Builder 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.

Replaces the default processing for checking record uniqueness. When a block has the PRIMKEYB property set to Yes, Form Builder, by default, checks the uniqueness of a record by constructing and executing the appropriate SQL statement to select for rows that match the current record's primary key values. If a duplicate row is found, Form Builder displays message FRM-40600: Record has already been inserted.

For a record that has been marked for insert, Form Builder always checks for unique primary key values. In the case of an update, Form Builder checks for unique primary key values only if one or more items that have the Primary Key item property have been modified.

Definition Level

form, block

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

To perform the default processing from this trigger, call the CHECK_RECORD_UNIQUENESS built-in.

On Failure

no effect

Fires In

Check Record Uniqueness

Post and Commit Transactions

See Process Flowcharts

On-Check-Unique trigger examples

Example

The following example verifies that the current record in question does not already exist in the DEPT table.

```
DECLARE
  CURSOR chk_unique IS SELECT 'x'
                         FROM dept
                        WHERE deptno = :dept.deptno;
  tmp VARCHAR2(1);
BEGIN
  OPEN chk_unique;
  FETCH chk_unique INTO tmp;
  CLOSE chk_unique;
  IF tmp IS NOT NULL THEN
    Message('This department already exists.');
    RAISE Form_trigger_Failure;
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Clear-Details trigger

Description

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.

Definition Level form, block

Legal Commands

Any command, unrestricted built-ins, restricted built-ins

Enter Query Mode no

Usage Notes

Form Builder creates the On-Clear-Details trigger automatically when a Master/Detail block relation is defined.

On Failure

Causes the coordination-causing operation and any scheduled coordination triggers to abort.

Fires In

Master Detail Coordination

See Process Flowcharts

On-Close trigger

Description

Fires when an operator or the application causes a query to *close*. By default, Form Builder 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 Form Builder "close cursor" phase of a query.

Definition Level form

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Close trigger after using the On-Select or On-Fetch triggers, specifically, to close files, close cursors, and free memory.
- The On-Close trigger fires automatically when the ABORT_QUERY built-in is called from an On-Select trigger.

On Failure

no effect

Fires In

ABORT_QUERY

Close The Query

On-Close trigger examples

Example

The following example releases memory being used by a user-defined data access method via the transactional triggers.

```
BEGIN
  IF NOT my_data source_open('DX110_DEPT') THEN
    my_datasource_close('DX110_DEPT');
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Column-Security trigger

Description

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

By default, Form Builder enforces column security by querying 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, Form Builder makes the corresponding base table items in the form non-updateable by setting the Update Allowed item property Off dynamically. Form Builder performs this operation at form startup, processing each block in sequence.

Definition Level form, block

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

To perform the default processing from this trigger, call the ENFORCE_COLUMN_SECURITY built-in.

On Failure

no effect

On-Column-Security trigger examples

Example

The following example sets salary and commission text items in the current block to disabled and non-updateable, unless the SUPERUSER role is enabled. Only users with the user-defined SUPERUSER role can change these number fields.

```
DECLARE
  itm_id Item;
  on_or_off NUMBER;
BEGIN
  IF NOT role_is_set('SUPERUSER') THEN
     on_or_off := PROPERTY_OFF;
  ELSE
    on_or_off := PROPERTY_ON;
  END IF;
  itm_id := Find_Item('Emp.Sal');
  Set_Item_Property(itm_id,ENABLED,on_or_off);
  Set_Item_Property(itm_id,UPDATEABLE,on_or_off);
  itm_id := Find_Item('Emp.Comm');
  Set_Item_Property(itm_id,ENABLED,on_or_off);
  Set_Item_Property(itm_id,UPDATEABLE,on_or_off);
```

```
 IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
END IF;
END;
```

On-Commit trigger

Description

Fires whenever Form Builder 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.

The default COMMIT statement that Form Builder issues to finalize a transaction during the Post and Commit Transactions process.

Definition Level form

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Commit trigger to change the conditions of normal Form Builder commit processing to fit the particular requirements of a commit to a non-ORACLE database.
- To perform the default processing from this trigger, call to the built-in.

On Failure

Aborts Post and Commit processing

Fires In

Post and Commit Transactions

See Process Flowcharts

On-Commit trigger examples

Example

This example disables the commit operation when running against a datasource that does not support transaction control. If the application is running against ORACLE, the commit operation behaves normally.

```
BEGIN
  IF Get_Application_Property(DATA_SOURCE) = 'ORACLE' THEN
    Commit_Form;
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
  /*
  ** Otherwise, no action is performed
  */
```

```
END;
```

On-Count trigger

Description

Fires when Form Builder 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, Form Builder issues the standard query hits message: FRM-40355: Query will retrieve <n> records.

Definition Level form, block

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode yes

Usage Notes

- Use an On-Count trigger to replace default Count Query processing in an application running against a non-ORACLE data source.
- To perform the default Form Builder processing from this trigger, include a call to the built-in.
- If you are replacing default processing, you can set the value of the Query_Hits block property to indicate the number of records in the non-ORACLE data source that match the query criteria.
- Form Builder will display the query hits message (FRM-40355) even if the On-Count trigger fails to set the value of the Query_Hits block property. In such a case, the message reports 0 records identified.

On Failure

no effect

Fires In

See Process Flowcharts

On-Count trigger examples

Example

This example calls a user-named subprogram to count the number of records to be retrieved by the current query criteria, and sets the Query_Hits property appropriately.

```
DECLARE
  j NUMBER;
BEGIN
  j := Recs_Returned('DEPT',Name_In('DEPT.DNAME'));
  Set_Block_Property('DEPT',QUERY_HITS,j);
END;
```

On-Delete trigger

Description

Fires during the Post and Commit Transactions process and replaces the default Form Builder processing for handling deleted records during transaction posting. 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.

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Delete trigger to replace the default Form Builder processing for handling deleted records during transaction posting.
- To perform the default Form Builder processing from this trigger, that is, to delete a record from your form or from the database, include a call to the DELETE_RECORD built-in.

On Failure

Form Builder rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

On-Delete trigger examples

Example

This example updates the employee table to set the Termination_Date, rather than actually deleting the employee from the database.

```
BEGIN
  UPDATE emp
     SET termination_date = SYSDATE
   WHERE empno = :Emp.Empno;
IF form_fatal OR form_failure THEN
   raise form_trigger_failure;
END IF;
END;
```

On-Dispatch-Event trigger

Description

This trigger is called when an ActiveX control event occurs. You can call the DISPATCH_EVENT built-in from within this trigger to specify the dispatch mode as either restricted or unrestricted. For more information about working with ActiveX control events, see Responding to ActiveX Control Events.

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode No

On Failure

No effect

On-Dispatch Event examples

Example

```
/*
ON-DISPATCH-EVENT trigger
*/
BEGIN
   IF SYSTEM.CUSTOM_ITEM_EVENT = 4294966696 THEN
         /*when event occurs, allow it to apply to different
items */.
         FORMS4W.DISPATCH_EVENT(RESTRICTED_ALLOWED);
   ELSE
         /*run the default, that does not allow applying any
other                        item */
         FORMS4W.DISPATCH_EVENT(RESTRICTED_UNALLOWED);
   ENDIF;
  IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;

END;
```

On-Error trigger

Description

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

Replaces

The writing of an error message to the message line.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode yes

Usage Notes

- Use an On-Error trigger for the following purposes:
- to trap and recover from an error
- to replace a standard error message with a custom message

  Use the ERROR_CODE , ERROR_TEXT , ERROR_TYPE , DBMS_ERROR_TEXT , or DBMS_ERROR_CODE built-in function in an On-Error trigger to identify a specific error condition.
- In most cases, On-Error triggers should be attached to the form, rather than to a block or item. Trapping certain errors at the block or item level can be difficult if these errors occur while Form Builder is performing internal navigation, such as during a Commit process.

On Failure

no effect

On-Error trigger examples

Example

The following example checks specific error message codes and responds appropriately.

```
DECLARE
   lv_errcod  NUMBER       := ERROR_CODE;
   lv_errtyp  VARCHAR2(3)  := ERROR_TYPE;
   lv_errtxt  VARCHAR2(80) := ERROR_TEXT;
BEGIN
   IF (lv_errcod = 40nnn) THEN
   /*
   **  Perform some tasks here
   */
   ELSIF (lv_errcod = 40mmm) THEN
```

```
   /*
   **  More tasks here
   */
...
...
ELSIF (lv_errcod = 40zzz) THEN
   /*
   **  More tasks here
   */
ELSE
   Message(lv_errtyp||'-'||to_char(lv_errcod)||': '||lv_errtxt);
   RAISE Form_trigger_Failure;
END IF;
END;
```

On-Fetch trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

When you write an On-Fetch trigger to replace default fetch processing, the trigger must do the following:

Retrieve the appropriate number of records from the non-ORACLE data source, as indicated by the setting of the Records_To_Fetch property.

Create that number of queried records in the current block.

Populate the records with the retrieved data.

Create queried records from an On-Fetch trigger by calling the CREATE_QUERIED_RECORD built-in subprogram.

While the query remains open, the On-Fetch trigger continues to fire as more records are needed in the block. This behavior continues:

until no queried records are created in a single execution of the trigger. Failing to create any records signals an end-of-fetch to Form Builder, indicating that there are no more records to be retrieved.

until the query is closed, either by the operator or programmatically through a call to ABORT_QUERY.

until the trigger raises the built-in exception FORM_TRIGGER_FAILURE.

-

To perform the default Form Builder processing from this trigger, include a call to the FETCH_RECORDS built-in.

Do not use an ABORT_QUERY built-in in an On-Fetch trigger. ABORT_QUERY is not valid in an On-Fetch trigger, and produces inconsistent results.

On Failure

no effect

Fires In

Fetch Records

See Process Flowcharts

On-Fetch trigger examples

This example calls a client-side package function to retrieve the proper number of rows from a package cursor.

```
DECLARE
  j NUMBER := Get_Block_Property(blk_name, RECORDS_TO_FETCH);
  emprow emp%ROWTYPE;

BEGIN
  FOR ctr IN 1..j LOOP
    /*
    ** Try to get the next row.
    */
    EXIT WHEN NOT MyPackage.Get_Next_Row(emprow);
    Create_Queried_Record;
    :Emp.rowid := emprow.ROWID;
    :Emp.empno := emprow.EMPNO;
    :Emp.ename := emprow.ENAME;
       :
       :
  END LOOP;
    IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Insert trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Insert trigger to replace the default Form Builder processing for handling inserted records during transaction posting.
- To perform the default Form Builder processing from this trigger, include a call to the INSERT_RECORD built-in.

On Failure

Form Builder performs the following steps when the On-Insert trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

On-Lock trigger

Description

Fires whenever Form Builder 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.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Lock trigger to replace the default Form Builder processing for locking rows. For example, for an application intended for use on a single-user system, use the On-Lock trigger to speed processing by bypassing all lock processing. Also, use On-Lock when accessing a non-ORACLE data source directly, not by way of Open Gateway.
- When the On-Lock trigger fires as a result of an operator trying to modify data, the trigger fires only the first time the operator tries to modify an item in the record. The trigger does not fire during subsequent modifications to items in the same record. In other words, for every row that is to be locked, the trigger fires once.
- To perform the default Form Builder processing from this trigger, include a call to the LOCK_RECORD built-in.
- Use this trigger to lock underlying tables for non-updateable views.

Caution

In special circumstances, you may use the LOCK TABLE DML statement in an On-Lock trigger. However, as this could result in other users being locked out of the table, please exercise caution and refer to the *ORACLE RDMS Database Administrator's Guide* before using LOCK TABLE.

On Failure

When the On-Lock trigger fails, the target record is not locked and Form Builder attempts to put the input focus on the current item. If the current item cannot be entered for some reason, Form Builder attempts to put the input focus on the previous navigable item.

Fires In

Lock the Row

See Process Flowcharts

On-Logon trigger

Description

Fires once for each logon when Form Builder normally initiates the logon sequence.

Definition Level form

Legal Commands

unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Logon trigger to initiate a logon procedure to a non-ORACLE data source.
- Pre-Logon and Post-Logon triggers fire as part of the logon procedure.
- To create an application that does not require a data source, supply a NULL command to this trigger to bypass the connection to a data source.
- To perform the default Form Builder processing from this trigger, include a call to the LOGON built-in.

On Failure

Form Builder attempts to exit the form gracefully, and does not fire the Post-Logon trigger.

Fires In

LOGON

See Process Flowcharts

On-Logout trigger

Description

Fires when Form Builder normally initiates a logout procedure from Form Builder and from the RDBMS.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use an On-Logout trigger to replace the default logout processing either from the RDBMS or from a non-ORACLE data source.
- To perform the default Form Builder processing from this trigger, include a call to the LOGOUT built-in.
- If you call certain built-ins from within one of the Logout triggers, the results are undefined. For example, you cannot call the COPY built-in from a Pre-Logout trigger because Pre-Logout fires after the Leave the Form event. Because the form is no longer accessible, a COPY operation is not possible.

On Failure

If an exception is raised in an On-Logout trigger and the current Runform session is exited, Form Builder will not fire other triggers, such as Post-Logout .

Fires In

LOGOUT

See Process Flowcharts

On-Message trigger

Description

Fires whenever Form Builder would normally cause a message to display and pre-empts the message.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use an On-Message trigger for the following purposes:

- to trap and respond to an informative message
- to replace a standard informative message with a custom message
- to exclude an inappropriate message
- Use the MESSAGE_CODE, MESSAGE_TEXT, MESSAGE_TYPE built-ins in an On-Message trigger to identify the occurrence of a specific message condition.
- If you use the On-Message trigger to trap a message so that it does not display on the message line, the GET_MESSAGE built-in does not return a value. To display the current message from this trigger, you must trap the message and explicitly write it to the display device.
- In most cases, On-Message triggers should be attached to the form, rather than to a block or item. Trapping certain errors at the block or item level can be difficult if these errors occur while Form Builder is performing internal navigation, such as during a Commit process.

On Failure

no effect

On-Message trigger examples

Example

The following example responds to an error message by displaying an alert that gives the user a message and gives the user the choice to continue or to stop:

```
DECLARE
  alert_button NUMBER;
  lv_errtype VARCHAR2(3) := MESSAGE_TYPE;
  lv_errcod NUMBER := MESSAGE_CODE;
  lv_errtxt VARCHAR2(80) := MESSAGE_TEXT;
BEGIN
  IF lv_errcod = 40350 THEN
```

```
    alert_button := Show_Alert('continue_alert');
    IF alert_button = ALERT_BUTTON1 THEN
      ...
    ELSE
      ...
    END IF;
  ELSE
    Message(lv_errtyp||'-'||to_char(lv_errcod)||':
'||lv_errtxt);
    RAISE Form_trigger_Failure;
  END IF;
    IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Populate-Details trigger

Description

Form Builder creates this trigger automatically when a Master/Detail relation is defined. It fires when Form Builder would normally need to populate the detail block in a Master/Detail relation.

Definition Level form, block

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins, restricted built-ins

Enter Query Mode no

Usage Notes

Use an On-Populate-Details trigger when you have established a Master/Detail relationship and you want to replace the default populate phase of a query.

The On-Populate-Details trigger does not fire unless an On-Clear-Details trigger is present. If you are using the default Master/Detail functionality, Form Builder creates the necessary triggers automatically. However, if you are writing your own Master/Detail logic, be aware that the On-Clear-Details trigger must be present, even if it contains only the NULL statement.

When Immediate coordination is set, this causes the details of the instantiated master to be populated immediately. Immediate coordination is the default.

When Deferred coordination is set and this trigger fires, Form Builder marks the blocks as needing to be coordinated.

If you intend to manage block coordination yourself, you can call the SET_BLOCK_PROPERTY (COORDINATION_STATUS) built-in.

On Failure

Can cause an inconsistent state in the form.

Fires In

Master/Detail Coordination

See Process Flowcharts

On-Rollback trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use an On-Rollback trigger to replace standard Form Builder rollback processing.

To perform default Form Builder processing from this trigger, include a call to the ISSUE_ROLLBACK built-in.

Fires In

CLEAR_FORM

Post and Commit Transactions

ROLLBACK_FORM

See Process Flowcharts

On-Savepoint trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

To perform default Form Builder processing from this trigger, include a call to the ISSUE_SAVEPOINT built-in.

In an On-Savepoint trigger, the Savepoint_Name application property returns the name of the next savepoint that Form Builder would issue by default, if no On-Savepoint trigger were present. In an On-Rollback trigger , Savepoint_Name returns the name of the savepoint to which Form Builder would roll back.

Suppress default savepoint processing by setting the Savepoint Mode form document property to Off. When Savepoint Mode is Off, Form Builder does not issue savepoints and, consequently, the On-Savepoint trigger never fires.

On Failure

no effect

Fires In

CALL_FORM

Post and Commit Transactions

SAVEPOINT

See Process Flowcharts

On-Select trigger

Description

Fires when Form Builder would normally execute the open cursor, parse, and execute phases of a query, to identify the records in the database that match the current query criteria.

Definition Level form or block

Legal Commands

SELECT statements, PL/SQL, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use an On-Select trigger to open and execute the database cursor. Specifically, use this trigger when you are retrieving data from a non-ORACLE data source. The On-Select trigger can be used in conjunction with the On-Fetch trigger to replace the processing that normally occurs in the EXECUTE_QUERY built-in subprogram.

To perform the default Form Builder processing from this trigger, include a call to the SELECT_RECORDS built-in.

On Failure

no effect

Fires In

EXECUTE_QUERY

Open The Query

See Process Flowcharts

On-Select trigger examples

Example

In the following example, the On-Select trigger is used to call a user exit, 'Query,' and a built-in subprogram, SELECT_RECORDS, to perform a query against a database.

```
IF Get_Application_Property(DATASOURCE) = 'DB2' THEN
  User_Exit ( 'Query' );
  IF Form_Failure OR Form_Fatal THEN
    ABORT_QUERY;
  END IF;
ELSE
  /*
```

```
  **  Perform the default Form Builder task of opening the
query.
  */
  Select_Records;
END IF;
```

On-Sequence-Number trigger

Description

Fires when Form Builder would normally perform the default processing for generating sequence numbers for default item values. Replaces the default series of events that occurs when Form Builder interacts with the database to get the next value from a SEQUENCE object defined in the database.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

When a SEQUENCE is used as a default item value, Form Builder queries the database to get the next value from the SEQUENCE whenever the Create Record event occurs. Suppress or override this functionality with an On-Sequence-Number trigger.

To perform the default Form Builder processing from this trigger, call the GENERATE_SEQUENCE_NUMBER built-in.

On Failure

no effect

Fires In

GENERATE_SEQUENCE_NUMBER

See Process Flowcharts

On-Update trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use an On-Update trigger to replace the default Form Builder processing for handling updated records during transaction posting.

To perform the default Form Builder processing from this trigger, include a call to the UPDATE_RECORD built-in.

On Failure

Form Builder performs the following steps when the On-Update trigger fails:

- sets the error location
- rolls back to the most recently issued savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Post-Block trigger

Description

Fires during the Leave the Block process when the focus moves off the current block.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Block trigger to validate the block's current record; that is, the record that had input focus when the Leave the Block event occurred.

Use this trigger to test a condition and prevent the user from leaving a block based on that condition.

On Failure

If the trigger fails while trying to make the form the navigation unit, Form Builder tries to set the target to a particular block, record or item. Failing that, Form Builder attempts to put the cursor at a target location, but, if the target is outside of the current unit or if the operator indicates an end to the process, Form Builder aborts the form.

Fires In

Leave the Block

See Process Flowcharts

Post-Block trigger restrictions

A Post-Block trigger does not fire when the Validation Unit form document property is set to Form.

Post-Change trigger

Description

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.
- Form Builder 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.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

The Post-Change trigger is included only for compatibility with previous versions of Form Builder. Its use is not recommended in new applications.

The Post-Query trigger does not have the restrictions of the Post-Change trigger. You can use Post-Query to make changes to the fetched database values. Given such changes, Form Builder marks the corresponding items and records as changed.

On Failure

If fired as part of validation initiated by navigation, navigation fails, and the focus remains in the original item.

If there are V2-style triggers in the form and Form Builder is populating a record with fetched values, the following restrictions apply:

Form Builder ignores any attempt to change the value of a database item in the record.

Form Builder does not execute any SELECT statement that only affects database items in the record.

Form Builder does not execute a SELECT statement that does not have an INTO clause.

If Form Builder does not execute a SELECT statement in a V2-style trigger step, it treats the trigger step as though the step succeeded, even when the Reverse Return Code trigger step property is set.

During fetch processing, Post-Change triggers defined as PL/SQL triggers do not operate with these restrictions. Regardless of trigger style, during a record fetch, Form Builder does not perform validation checks, but marks the record and its items as Valid, after firing the Post-Change trigger for each item.

Fires In

Validate the Item

Fetch Records

See Process Flowcharts

Post-Change trigger restrictions

Note that it is possible to write a Post-Change trigger that changes the value of an item that Form Builder is validating. If validation succeeds, Form Builder marks the changed item as Valid and does not re-validate it. While this behavior is necessary to avoid validation loops, it does allow you to commit an invalid value to the database.

Post-Database-Commit trigger

Description

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 Form Builder issues the Commit to finalize the transaction.

Definition Level form

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Database-Commit trigger to perform an action anytime a database commit has occurred.

On Failure

There is no rollback, because at the point at which this trigger might fail, Form Builder has already moved past the point at which a successful rollback operation can be initiated as part of a failure response.

Fires In

Post and Commit Transactions

See Process Flowcharts

Post-Database-Commit trigger examples

Example

```
/*
** FUNCTION recs_posted_and_not_committed
**  RETURN BOOLEAN IS
** BEGIN
**  Default_Value('TRUE','Global.Did_DB_Commit');
**  RETURN (:System.Form_Status = 'QUERY'
**         AND :Global.Did_DB_Commit = 'FALSE');
** END;
*/
BEGIN
  :Global.Did_DB_Commit := 'FALSE';
END;
```

Post-Delete trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Delete trigger to audit transactions.

On Failure

Form Builder performs the following steps when the Post-Delete trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Post-Form trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Form trigger for the following tasks:

- To *clean up* the form before exiting. For example, use a Post-Form trigger to erase any global variables that the form no longer requires.
- To display a message to the operator upon form exit.

This trigger does not fire when the form is exited abnormally, for example, if validation fails in the form.

On Failure

processing halts

Fires In

Leave the Form

See Process Flowcharts

Post-Forms-Commit trigger

Description

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 Form Builder 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, Form Builder fires the Post-Forms-Commit trigger immediately, without posting changes to the database.

Definition Level form

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Forms-Commit trigger to perform an action, such as updating an audit trail, anytime a database commit is about to occur.

On Failure

Aborts post and commit processing: Form Builder issues a ROLLBACK and decrements the internal Savepoint counter.

Fires In

Post and Commit Transactions

See Process Flowcharts

Post-Forms-Commit trigger examples

Example

This example can be used in concert with the Post-Database-Commit trigger to detect if records have been posted but not yet committed.

```
/*
** FUNCTION recs_posted_and_not_committed
** RETURN BOOLEAN IS
** BEGIN
**    Default_Value('TRUE','Global.Did_DB_Commit');
**    RETURN (:System.Form_Status = 'QUERY'
**       AND :Global.Did_DB_Commit = 'FALSE');
** END;
*/
```

```
BEGIN
  :Global.Did_DB_Commit := 'FALSE';
END;
```

Post-Insert trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Insert trigger to audit transactions.

On Failure

Form Builder performs the following steps when the Post-Insert trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Post-Logon trigger

Description

Fires after either of the following events:

- The successful completion of Form Builder default logon processing.
- The successful execution of the On-Logon trigger.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Logon trigger to provide an event point for a task such as setting up a custom environment with special variables--to initialize on an application level rather than a form-by-form basis. You might accomplish this by initializing special global variables from this trigger.

On Failure

Varies based on the following conditions:

- If the trigger fails during the first logon process, Form Builder exits the form, and returns to the operating system.
- If the trigger fails after a successful logon, Form Builder raises the built-in exception FORM_TRIGGER_FAILURE .

Fires In

LOGON

See Process Flowcharts

Post-Logon trigger examples

Example

This example calls a user exit to log the current username and time to an encrypted audit trail file on the file system, which for security reasons is outside the database.

```
BEGIN
  User_Exit('LogCrypt '||
            USER||' '  ||
            TO_CHAR(SYSDATE,'YYYYMMDDHH24MISS'));
END;
```

Post-Logout trigger

Description

Fires after either of the following events:

- Form Builder successfully logs out of ORACLE.
- The successful execution of the On-Logout trigger.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Logout trigger to audit or to perform tasks on an Form Builder application that does not require or affect the RDBMS or other data source.

If you call certain built-ins from within one of the Logout triggers, the results are undefined. For example, you cannot call COPY from a Pre-Logout trigger because Pre-Logout fires after the Leave the Form event. Because the form is no longer accessible, a COPY operation is not possible.

On Failure

If this trigger fails while leaving the form, there is no effect.

If this trigger fails and you have initiated a call to the LOGOUT built-in from within the trigger, FORM_FAILURE is set to TRUE.

Fires In

LOGOUT

See Process Flowcharts

Post-Query trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Query trigger to perform the following tasks:

- populate control items or items in other blocks
- calculate statistics about the records retrieved by a query
- calculate a running total
- When you use a Post-Query trigger to SELECT non-base table values into control items, Form Builder marks each record as CHANGED, and so fires the When-Validate-Item trigger by default. You can avoid the execution of the When-Validate-Item trigger by explicitly setting the Status property of each record to QUERY in the Post-Query trigger. To set record status programmatically, use SET_RECORD_PROPERTY .

On Failure

Form Builder flushes the record from the block and attempts to fetch the next record from the database. If there are no other records in the database, Form Builder closes the query and waits for the next operator action.

Fires In

Fetch Records

See Process Flowcharts

Post-Query trigger examples

Example

This example retrieves descriptions for code fields, for display in non-database items in the current block.

```
DECLARE

  CURSOR lookup_payplan IS SELECT Payplan_Desc
                             FROM Payplan
```

```
                              WHERE Payplan_Id =
                                    :Employee.Payplan_Id;

  CURSOR lookup_area    IS SELECT Area_Name
                            FROM Zip_Code
                            WHERE Zip = :Employee.Zip;

BEGIN
  /*
  ** Lookup the Payment Plan Description given the
  ** Payplan_Id in the Employee Record just fetched.
  ** Use Explicit Cursor for highest efficiency.
  */
  OPEN lookup_payplan;
  FETCH lookup_payplan INTO :Employee.Payplan_Desc_Nondb;
  CLOSE lookup_payplan;

  /*
  ** Lookup Area Descript given the Zipcode in
  ** the Employee Record just fetched. Use Explicit
  ** Cursor for highest efficiency.
  */
  OPEN lookup_area;
  FETCH lookup_area INTO :Employee.Area_Desc_Nondb;
  CLOSE lookup_area;
END;
```

Post-Record trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Record trigger to perform an action whenever the operator or the application moves the input focus from one record to another. For example, to set a visual attribute for an item as the operator scrolls down through a set of records, put the code within this trigger.

On Failure

The input focus stays in the current record.

Fires In

Leave the Record

See Process Flowcharts

Post-Record trigger restrictions

A Post-Record trigger fires only when the form is run with a validation unit of the item or record, as specified by the Validation Unit form property.

Post-Select trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Note:

Use the Post-Select trigger to perform an action based on the outcome of the Select phase of query processing such as an action based on the number of records that match the query criteria.

On Failure

no effect

Fires In

Execute the Query

Open the Query

See Process Flowcharts

Post-Text-Item trigger

Description

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.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Text-Item trigger to calculate or change item values.

On Failure

Navigation fails and focus remains in the text item.

Fires In

Leave the Item

See Process Flowcharts

Post-Text-Item trigger restrictions

The Post-Text-Item trigger does not fire when the input focus is in a text item and the operator uses the mouse to click on a button, check box, list item, or radio group item that has the Mouse Navigate property Off. When Mouse Navigate is Off for these items, clicking them with the mouse is a non-navigational event, and the input focus remains in the current item (in this example, a text item).

Post-Update trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted function codes, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Post-Update trigger to audit transactions.

On Failure

Form Builder performs the following steps when the Post-Update trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Pre-Block trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Block trigger to:

- allow or disallow access to a block
- set variable values

On Failure

Navigation fails and focus remains in the source item.

Fires In

Enter the Block

See Process Flowcharts

Pre-Block trigger restrictions

A Pre-Block trigger fires only when the form is run with a validation unit of the item, record, or block, as specified by the Validation Unit form property.

Pre-Commit trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Commit trigger to perform an action, such as setting up special locking requirements, at any time a database commit is going to occur.

On Failure

The Post and Commit process fails: No records are written to the database and focus remains in the current item.

Note: If you perform DML in a Pre-Commit trigger and the trigger fails, you must perform a manual rollback, because Form Builder does not perform an automatic rollback. To prepare for a possible manual rollback, save the savepoint name in an On-Savepoint trigger, using GET_APPLICATION_PROPERTY (Savepoint_Name). Then you can roll back using ISSUE_ROLLBACK (Savepoint_Name).

Fires In

Post and Commit Transactions

See Process Flowcharts

Pre-Delete trigger

Description

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: Form Builder creates a Pre-Delete trigger automatically for any master-detail relation that has the Delete Record Behavior property set to Cascading

Definition Level form or block

Legal Commands

SELECT statements, Data Manipulation Language (DML) statements (i.e., DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Delete trigger to delete the detail record of a master record.

Use a Pre-Delete trigger to prevent the deletion of a record if that record is the master record for detail records that still exist.

On Failure

Form Builder performs the following steps when the Pre-Delete trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Pre-Form trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Form trigger to perform the following tasks:

- assign unique primary key from sequence
- restrict access to a form
- initialize global variables

On Failure

Form Builder leaves the current form and fires no other triggers.

Fires In

Enter the Form

See Process Flowcharts

Pre-Insert trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Insert trigger to perform the following tasks:

- change item values
- keep track of the date a record is created and store that in the record prior to committing

On Failure

Form Builder performs the following steps when the Pre-Insert trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Pre-Insert trigger examples

Example

This example assigns a primary key field based on a sequence number, and then writes a row into an auditing table, flagging creation of a new order.

```
DECLARE
  CURSOR next_ord IS SELECT orderid_seq.NEXTVAL FROM dual;
BEGIN

  /*
  ** Fetch the next sequence number from the
  ** explicit cursor directly into the item in
  ** the Order record. Could use SELECT...INTO,
  ** but explicit cursor is more efficient.
  */
  OPEN next_ord;
```

```
  FETCH next_ord INTO :Order.OrderId;
  CLOSE next_ord;

  /*
  ** Make sure we populated a new order id ok...
  */
  IF :Order.OrderId IS NULL THEN
    Message('Error Generating Next Order Id');
    RAISE Form_trigger_Failure;
  END IF;

  /*
  ** Insert a row into the audit table
  */
  INSERT INTO ord_audit( orderid, operation, username, timestamp
)
  VALUES ( :Order.OrderId,
           'New Order',
           USER,
           SYSDATE );

END;
```

Pre-Logon trigger

Description

Fires just before Form Builder initiates a logon procedure to the data source.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Logon trigger to prepare the form for the logon procedure, particularly to a non-ORACLE data source.

On Failure

The results of a failure depend on which of the following conditions applies:

- If Form Builder is entering the form for the first time and the trigger fails, the form is exited gracefully, but no other triggers are fired.
- If the trigger fails while Form Builder is attempting to execute the LOGON built-in from within the trigger, Form Builder raises the FORM_TRIGGER_FAILURE exception.

Fires In

LOGON

See Process Flowcharts

Pre-Logout trigger

Description

Fires once before Form Builder initiates a logout procedure.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Logout trigger to prepare the form for logging out from the data source, particularly a non-ORACLE data source.

If you call certain built-ins from within one of the Logout triggers, the results are undefined. For example, the COPY built-in cannot be called from a Pre-Logout trigger because Pre-Logout fires after the Leave the Form event. Because the form is no longer accessible at this point, the COPY operation is not possible.

On Failure

The results of a failure depend on which of the following conditions applies:

- If Form Builder is exiting the form and the trigger fails, the form is exited gracefully, but no other triggers are fired.
- If the trigger fails while Form Builder is attempting to execute the LOGOUT built-in from within the trigger, Form Builder raises the FORM_TRIGGER_FAILURE exception.

If an exception is raised in a Pre-Logout trigger, Form Builder does not fire other triggers, such as On-Logout and Post-Logout .

Fires In

LOGOUT

See Process Flowcharts

Pre-Popup-Menu trigger

Description

This trigger is called when a user causes a pop-up menu to be displayed. (In a Microsoft Windows environment, this occurs when a user presses the right mouse button.) Actions defined for this trigger are performed before the pop-up menu is displayed.

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use this trigger to enable or disable menu items on a pop-up menu before it is displayed.

On Failure

No effect

Pre-Query trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Query trigger to modify the example record that determines which rows will be identified by the query.

On Failure

The query is canceled. If the operator or the application had placed the form in Enter Query mode, the form remains in Enter Query mode.

Fires In

COUNT_QUERY

EXECUTE_QUERY

Open the Query

Prepare the Query

See Process Flowcharts

Pre-Query trigger examples

Example

This example validates or modifies query criteria for a database block query.

```
BEGIN
  /*
  ** Set the ORDER BY clause for the current block
  ** being queried, based on a radio group
  ** called 'Sort_Column' in a control block named
  ** 'Switches'. The Radio Group has three buttons
  ** with character values giving the names of
  ** three different columns in the table this
  ** block is based on:
  **
  **     SAL
```

```
  **     MGR,ENAME
  **     ENAME
  */
  Set_Block_Property('EMP',ORDER_BY, :Switches.Sort_Column);
  /*
  ** Make sure the user has given one of the two
  ** Columns which we have indexed in their search
  ** criteria, otherwise fail the query with a helpful
  ** message
  */
  IF :Employee.Ename IS NULL AND :Employee.Mgr IS NULL THEN
    Message('Supply Employee Name and/or Manager Id '||
            'for Query.');
    RAISE Form_trigger_Failure;
  END IF;

  /*
  ** Change the default where clause to either show "Current
  ** Employees Only" or "Terminated Employees" based on the
  ** setting of a check box named 'Show_Term' in a control
  ** block named 'Switches'.
  */
  IF Check box_Checked('Switches.Show_Term') THEN
    Set_Block_Property('EMP',DEFAULT_WHERE,'TERM_DATE IS NOT
NULL');
  ELSE
    Set_Block_Property('EMP',DEFAULT_WHERE,'TERM_DATE IS NULL');
  END IF;
END;
```

Pre-Record trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Record trigger to keep a running total.

On Failure

Navigation fails and focus remains in the current item.

Fires In

Enter the Record

See Process Flowcharts

Pre-Record trigger restrictions

A Pre-Record trigger fires only when the form is run with a validation unit of the item or record, as specified by the Validation Unit form property.

Pre-Record trigger examples

Example

The following trigger prevents the user from entering a new record given some dynamic condition and the status of SYSTEM.RECORD_STATUS evaluating to NEW.

```
IF (( dynamic-condition)
   AND :System.Record_Status = 'NEW') THEN
   RAISE Form_trigger_Failure;
END IF;
```

Pre-Select trigger

Description

Fires during Execute Query and Count Query processing, after Form Builder 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 .

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Select trigger to prepare a query prior to execution against a non-ORACLE data source.

On Failure

No effect. The current query fetched no records from the table. The table is empty, or it contains no records that meet the query's search criteria.

Fires In

EXECUTE_QUERY

Open the Query

Prepare the Query

See Process Flowcharts

Pre-Text-Item trigger

Description

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

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Text-Item trigger to perform the following types of tasks:

- derive a complex default value, based on other items previously entered into the same record.
- record the current value of the text item for future reference, and store that value in a global variable or form parameter.

On Failure

Navigation fails and focus remains in the current item.

Fires In

Enter the Item

See Process Flowcharts

Pre-Text-Item trigger restrictions

A Pre-Text-Item trigger fires only when the form is run with a validation unit of the item, as specified by the Validation Unit form property.

Pre-Update trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, DML statements (DELETE, INSERT, UPDATE), unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a Pre-Update trigger to audit transactions.

On Failure

Form Builder performs the following steps when the Pre-Update trigger fails:

- sets the error location
- rolls back to the most recent savepoint

Fires In

Post and Commit Transactions

See Process Flowcharts

Pre-Update trigger examples

Example

The following example writes a row into an Audit Table showing old discount and new discount for a given customer, including timestamp and username making the change.

```
DECLARE
  old_discount NUMBER;
  new_discount NUMBER := :Customer.Discount_Pct;
  oper_desc    VARCHAR2(80);
  CURSOR old_value IS SELECT discount_pct FROM customer
                      WHERE CustId = :Customer.CustId;
BEGIN
  /*
  ** Fetch the old value of discount percentage from the
  ** database by CustomerId. We need to do this since the
  ** value of :Customer.Discount_Pct will be the *new* value
  ** we're getting ready to commit and we want to record for
  ** posterity the old and new values. We could use
  ** SELECT...INTO but choose an explicit cursor for
```

```
  ** efficiency.
  */
  OPEN old_value;
  FETCH old_value INTO old_discount;
  CLOSE old_value;

  /*
  ** If the old and current values are different, then
  ** we need to write out an audit record
  */
  IF old_discount <> new_discount THEN
    /*
    ** Construct a string that shows the operation of
    ** Changing the old value to the new value. e.g.
    **
    **   'Changed Discount from 13.5% to 20%'
    */
    oper_desc := 'Changed Discount from '||
                 TO_CHAR(old_discount)||'% to '||
                 TO_CHAR(new_discount)||'%';

    /*
    ** Insert the audit record with timestamp and user
    */
    INSERT INTO cust_audit( custid, operation, username,
       timestamp )
    VALUES ( :Customer.CustId,
             oper_desc,
             USER,
             SYSDATE );
  END IF;
END;
```

Query-Procedure trigger

Description

Automatically created by Form Builder when the query data source is a stored procedure. This trigger is called when a query operation is necessary. Think of this as an On-Query trigger that is called by the system instead of doing default query operations.

Do not modify this trigger.

Enter Query Mode See Usage Notes

Usage Notes

When constructing a query, any of the items may be used, but the Query Data Source Columns property must be set so that those items can be passed to the query stored procedure. Then, the query stored procedure has to use those values to filter the data. This means that the enter query mode does not happen automatically unless you specify it.

On Failure

No effect

Update-Procedure trigger

Description

Automatically created by Form Builder when the update data source is a stored procedure. This trigger is called when a update operation is necessary. Think of this as an On-Update trigger that is called by the system instead of doing default update operations.

Do not modify this trigger.

Enter Query Mode Not applicable.

On Failure

No effect

User-Named trigger

Description

A user-named trigger is a trigger defined in a form by the developer. User-Named triggers do not automatically fire in response to a Form Builder event, and must be called 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');
```

Definition Level form, block, or item

Legal Commands

Any commands that are legal in the parent trigger from which the user-named trigger was called.

Enter Query Mode no

Usage Notes

User-named PL/SQL subprograms can be written to perform almost any task for which one might use a user-named trigger.

As with all triggers, the scope of a user-named trigger is the definition level and below. When more than one user-named trigger has the same name, the trigger defined at the lowest level has precedence.

It is most practical to define user-named triggers at the form level.

Create a user-named trigger to execute user-named subprograms defined in a form document from menu PL/SQL commands and user-named subprograms. (User-named subprograms defined in a form cannot be called directly from menu PL/SQL, which is defined in a different document.) In the menu PL/SQL, call the EXECUTE_TRIGGER built-in to execute a user-named trigger, which in turn calls the user-named subprogram defined in the current form.

On Failure

Sets the FORM_FAILURE built-in to TRUE. Because the user-named trigger is always called by the EXECUTE_TRIGGER built-in, you can test the outcome of a user-named trigger the same way you test the outcome of a built-in subprogram; that is, by testing for errors with the built-in functions FORM_FAILURE, FORM_SUCCESS, FORM_FATAL .

When-Button-Pressed trigger

Description

Fires when an operator selects a button, by clicking with a mouse, or using the keyboard.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Button-Pressed trigger to perform navigation, to calculate text item values, or for other item, block, or form level functionality.

On Failure

no effect

When-Button-Pressed trigger examples

Example

This example executes a COMMIT_FORM if there are changes in the form.

```
BEGIN
  IF :System.Form_Status = 'CHANGED' THEN
    Commit_Form;
    /*
    ** If the Form_Status is not back to 'QUERY'
    ** following a commit, then the commit was
    ** not successful.
    */
    IF :System.Form_Status <> 'QUERY' THEN
      Message('Unable to commit order to database...');
      RAISE Form_trigger_Failure;
    END IF;
  END IF;
END;
```

When-Checkbox-Changed trigger

Description

Fires when an operator changes the state of a check box, either by clicking with the mouse, or using the keyboard.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Checkbox-Changed trigger to initiate a task dependent upon the state of a check box.

When an operator clicks in a check box, the internal value of that item does not change until navigation is completed successfully. Thus, the When-Checkbox-Changed trigger is the first trigger to register the changed value of a check box item. So for all navigation triggers that fire before the When-Checkbox-Changed trigger, the value of the check box item remains as it was before the operator navigated to it.

On Failure

no effect

When-Clear-Block trigger

Description

Fires just before Form Builder clears the data from the current block.

Note that the When-Clear-Block trigger does not fire when Form Builder clears the current block during the CLEAR_FORM event.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode yes

Usage Notes

- Use a When-Clear-Block trigger to perform an action every time Form Builder flushes the current block. For example, you might want to perform an automatic commit whenever this condition occurs.
- In a When-Clear-Block trigger, the value of SYSTEM.RECORD_STATUS is unreliable because there is no current record. An alternative is to use GET_RECORD_PROPERTY to obtain the record status. Because GET_RECORD_PROPERTY requires reference to a specific record, its value is always accurate.

On Failure

no effect on the clearing of the block

Fires In

CLEAR_BLOCK

COUNT_QUERY

ENTER_QUERY

Open the Query

See Process Flowcharts

When-Create-Record trigger

Description

Fires when Form Builder 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, Form Builder fires this trigger.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a When-Create-Record trigger to perform an action every time Form Builder attempts to create a new record. This trigger also is useful for setting complex, calculated, or data-driven default values that must be specified at runtime, rather than at design-time.

On Failure

Prevents the new record from being created. Returns to the previous location, if possible.

Fires In

CREATE_RECORD

See Process Flowcharts

When-Create-Record trigger examples

Example

This example assigns data-driven or calculated default values without marking the record as changed.

```
DECLARE
  CURSOR ship_dflt IS SELECT val
                        FROM cust_pref
                       WHERE Custid = :Customer.Custid
                         AND pref   = 'SHIP';
BEGIN
  /*
  ** Default Invoice Due Date based on Customer's
  ** Net Days Allowed value from the Customer block.
  */
  :Invoice.Due_Date := SYSDATE + :Customer.Net_Days_Allowed;
  /*
  ** Default the shipping method based on this customers
  ** preference, stored in a preference table. We could
  ** use SELECT...INTO, but explicit cursor is more
  ** efficient.
```

```
  */
  OPEN ship_dflt;
  FETCH ship_dflt INTO :Invoice.Ship_Method;
  CLOSE ship_dflt;
END;
```

When-Custom-Item-Event trigger

Description

Fires whenever a JavaBean or ActiveX (on 32-bit Windows) or VBX (on 16-bit Microsoft Windows 3.x) custom component in the form causes the occurrence of an event.

Definition Level:

form, block, item

Legal Commands:

unrestricted built-ins, restricted built-ins

Enter Query Mode:

yes

Usage Notes

Use a When-Custom-Item-Event trigger to respond to a selection or change of value of a custom component. The system variable SYSTEM.CUSTOM_ITEM_EVENT stores the name of the event that occurred, and the system variable SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS stores a parameter name that contains the supplementary arguments for an event that is fired by a custom control.

Control event names are case sensitive.

On Failure:

no effect

When-Custom-Item-Event trigger examples

JavaBeans Example

This is an example of a procedure called by a When-Custom-Item-Event trigger in a form that uses two JavaBeans. (For the context, see the complete example.)

The trigger is fired by the dispatching of a custom event in one of the JavaBeans, in response to a change in value.

```
CustomEvent ce = new CustomEvent(mHandler, VALUECHANGED);
dispatchCustomEvent(ce);
```

In the form, the When_Custom_Item_Event trigger that is attached to this JavaBean's Bean Area item is automatically fired in response to that custom event.

In the trigger code, it executes the following procedure. Note that this procedure picks up the new values set in the JavaBean container (a new animation rate, for example) by accessing the System.Custom_Item_Event_Parameters.

In this example, the procedure then uses the Set_Custom_Item_Property built-in to pass those values to the other JavaBean.

```
PROCEDURE Slider_Event_Trap IS
  BeanHdl          Item;
   BeanValListHdl   ParamList;
   paramType        Number;
   EventName        VarChar2(20);
   CurrentValue     Number(4);
   NewAnimationRate Number(4);
Begin
   -- Update data items and Display fields with current radius
information
   BeanValListHdl :=
get_parameter_list(:SYSTEM.Custom_Item_Event_Parameters);
   EventName      := :SYSTEM.Custom_Item_Event;
   :event_name    := EventName;
   if (EventName = 'ValueChanged') then
      get_parameter_attr(BeanValListHdl,'Value',ParamType,
CurrentValue);
      NewAnimationRate := (300 - CurrentValue);
      :Animation_Rate  := NewAnimationRate;
     set_custom_item_property('Juggler_Bean','SetAnimationRate',
NewAnimationRate);
   elsif (EventName = 'mouseReleased') then
      get_parameter_attr(BeanValListHdl,'Value',ParamType,
CurrentValue);
     set_custom_item_property('Juggler_Bean','SetAnimationRate',
CurrentValue);
  end if;
End;
```

VBX Example

This is an example of a procedure that can be called when Form Builder fires the When-Custom-Item-Event trigger.

```
DECLARE
 TabEvent  varchar2(80);
 TabNumber Number;
BEGIN
 TabEvent := :system.custom_item_event;
 /*
 ** After detecting a Click event, identify the
 ** tab selected, and use the user-defined Goto_Tab_Page
 ** procedure to navigate to the selected page.
 */
 IF (UPPER(TabEvent) = 'CLICK') THEN
```

```
  TabNumber := VBX.Get_Property('TABCONTROL','CurrTab');
  Goto_Tab_Page(TabNumber);
 END IF;
END;
```

When-Database-Record trigger

Description

Fires when Form Builder first marks a record as an insert or an update. That is, the trigger fires as soon as Form Builder 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.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a When-Database-Record trigger to perform an action every time a record is first marked as an insert or an update.

On Failure

no effect

When-Form-Navigate trigger

Description

Fires when navigation between forms takes place, such as when the user changes the focus to another loaded form.

Definition Level form

Legal Commands:

unrestricted built-ins, restricted built-ins

Enter Query Mode:

no

Usage Notes

Use a When-Form-Navigate trigger to perform actions when any cross form navigation takes place without relying on window activate and window deactivate events.

On Failure

no effect

When-Form-Navigate trigger examples

Example

This is an example of a procedure that can be called when Form Builder fires the When-Form-Navigate trigger.

```
DECLARE
 win_id WINDOW := FIND_WINDOW('WINDOW12');
BEGIN
 if (GET_WINDOW_PROPERTY(win_id,WINDOW_STATE) = 'MAXIMIZE' THEN
  SET_WINDOW_PROPERTY(win_id,WINDOW_STATE,MINIMIZE);
 else
  SET_WINDOW_PROPERTY(win_id,WINDOW_STATE,MAXIMIZE);
 end if;
END;
```

When-Image-Activated trigger

Description

Fires when an operator uses the mouse to:

- single-click on an image item

double-click on an image item

Note that When-Image-Pressed also fires on a double-click.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

On Failure

no effect

When-Image-Pressed trigger

Description

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.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Image-Pressed trigger to perform an action when an operator clicks or double-clicks on an image item.

On Failure

no effect

When-List-Activated trigger

Description

Fires when an operator double-clicks on an element in a list item that is displayed as a T-list.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

A When-List-Activated trigger fires only for T-list style list items, not for drop-down lists or combo box style list items. The display style of a list item is determined by the List Style property.

On Failure

no effect

When-List-Changed trigger

Description

Fires when an end user 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 end user enters or modifies entered text.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-List-Changed trigger to initiate an action when the value of the list is changed directly by the end user. The When-List-Changed trigger is not fired if the value of the list is changed programmatically such as by using the DUPLICATE_ITEM built-in, or if the end user causes a procedure to be invoked which changes the value. For example, the When-List-Changed trigger will not fire if an end user duplicates the item using a key mapped to the DUPLICATE_ITEM built-in.

On Failure

no effect

When-Mouse-Click trigger

Description

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 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.

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use the When-Mouse-Click trigger to perform an action every time the operator clicks the mouse within an item and/or canvas.

On Failure

no effect

When-Mouse-DoubleClick trigger

Description

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 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.

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Mouse-DoubleClick trigger to perform an action every time the operator double-clicks the mouse within an item and/or canvas.

On Failure

no effect

When-Mouse-DoubleClick trigger examples

Example

Assume that an application requires Behavior A when the operator clicks the mouse and Behavior B when the operator double-clicks the mouse. For example, if the operator clicks the mouse, a product information window must appear. If the operator double-clicks the mouse, an online help window must appear.

Three triggers are used in this example, a When-Mouse-Click trigger, a When-Timer-Expired trigger, and a When-Mouse-DoubleClick trigger.

```
/*
**  trigger:  When-Mouse-Click
**  Example:  When the operator clicks the mouse, create a timer
**            that will expire within .5 seconds.
*/

DECLARE
   timer_id        TIMER;
   timer_duration  NUMBER(5) := 500;
BEGIN
   timer_id := Create_Timer('doubleclick_timer', timer_duration,
     NO_REPEAT);
END;

/*
**  trigger:  When-Timer-Expired
**  Example:  When the timer expires display the online help
**            window if the operator has double-clicked the
mouse
**            within .5 seconds, otherwise display the product
**            information window.
*/
BEGIN
   IF :Global.double_click_flag = 'TRUE' THEN
     Show_Window('online_help');
     :Global.double_click := 'FALSE';
   ELSE
     Show_Window('product_information');
   END IF;
END;

/*
**  trigger:  When-Mouse-DoubleClick
**  Example:  If the operator double-clicks the mouse, set a
**            flag that indicates that a double-click event
**            occurred.
*/
BEGIN
   :Global.double_click_flag := 'TRUE';
END;
```

When-Mouse-Down trigger

Description

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 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

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Mouse-Down trigger to perform an action every time the operator presses down the mouse button within an item and/or canvas.

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

On Failure

no effect

When-Mouse-Down trigger restrictions

Depending on the window manager, navigational code within a When-Mouse-Down trigger may fail. For example on Microsoft Windows, if the operator clicks the mouse button within a field (Item_One), a When-Mouse-Down trigger that calls GO_ITEM ('item_two') will fail because Windows will return focus to Item_One, not Item_Two since the When-Mouse-Up event occurred within Item_Two.

When-Mouse-Enter trigger

Description

Fires when the mouse enters an item or canvas if one of the following events occurs:

- if attached to the form, when the mouse enters any canvas 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

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Mouse-Enter trigger to perform an action every time the mouse enters an item or canvas.

Do not use the When-Mouse-Enter trigger on a canvas that is larger than the window. Iconic buttons and items on the canvas below the initial window cannot be selected. The user is able to scroll the canvas to see the items. However, as soon as the mouse enters that area, the trigger fires and returns focus to the previous target, so the user is never able to click on those items.

Changing a tooltip's property in a When-Mouse-Enter trigger cancels the tooltip before it is ever shown.

Be careful when calling a modal window from a When-Mouse-Enter trigger. Doing so may cause the modal window to appear unnecessarily.

For example, assume that your When-Mouse-Enter trigger causes Alert_One to appear whenever the mouse enters Canvas_One. Assume also that your application contains two canvases, Canvas_One and Canvas_Two. Canvas_One and Canvas_Two do not overlap each other, but appear side by side on the screen. Further, assume that Alert_One displays within Canvas_Two's border.

Finally, assume that the mouse has entered Canvas_One causing the When-Mouse-Enter trigger to fire which in turn causes Alert_One to appear.

When the operator dismisses the message box, Alert_One will appear again unnecessarily *if* the operator subsequently enters Canvas_One with the mouse. In addition, when the operator moves the mouse out of Canvas_Two, any When-Mouse-Leave triggers associated with this event will fire. This may not be the desired behavior.

On Failure

no effect

When-Mouse-Leave trigger

Description

Fires after the mouse leaves an item or canvas if one of the following events occurs:

- if attached to the form, when the mouse leaves any canvas 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

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Mouse-Leave trigger to perform an action every time the mouse leaves an item and/or canvas.

On Failure

no effect

When-Mouse-Move trigger

Description

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 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

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use the When-Mouse-Move trigger to perform an action every time the operator moves the mouse.

The When-Mouse-Move trigger may have performance implications because of the number of times this trigger can potentially fire.

On Failure

no effect

When-Mouse-Up trigger

Description

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 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

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode yes

Usage Notes

Use the When-Mouse-Up trigger to perform an action every time the operator presses and releases the mouse.

The mouse up event is always associated with the item that received the mouse down event. For example, assume that there is a When-Mouse-Up trigger attached to Item_One. If the operator presses down the mouse on Item_One, but then releases the mouse on Item_Two, the mouse up trigger will fire for Item_One, rather than for Item_Two.

On Failure

no effect

When-New-Block-Instance trigger

Description

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

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode no

Usage Notes

Use a When-New-Block-Instance trigger to perform an action every time Form Builder instantiates a new block.

On Failure

no effect

Fires In

Return for Input

See Process Flowcharts

When-New-Form-Instance trigger

Description

At form start-up, Form Builder 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.

Definition Level form

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins

Enter Query Mode no

On Failure

no effect

Fires In

Run the Form

See Process Flowcharts

When-New-Form-Instance trigger restrictions

- When you call FORMS_OLE.GET_INTERFACE_POINTER from the When-New-Form-Instance trigger, an exception (ORA-305500) is raised unless you initialize the OLE item or the ActiveX control with the SYNCHRONIZE built-in.
- When a new form is called, it will appear in the default x-y position on the screen. If this is not the desired position, you can change the x-y coordinates. However, they cannot be changed in this When-New-Form-Instance trigger. (This trigger fires too late in the sequence.) To change the coordinates, use the Pre-Form trigger.

When-New-Form-Instance trigger examples

Example

This example calls routine to display dynamic images, starts a timer to refresh the on-screen clock, and queries the first block.

```
BEGIN
  Populate_Dynamic_Boilerplate;
  Start_OnScreen_Clock_Timer;
  Go_Block('Primary_Ord_Info');
```

```
  /*
  ** Query the block without showing
  ** the working message.
  */
  :System.Suppress_Working := 'TRUE';
  Execute_Query;
  :System.Suppress_Working := 'FALSE';
END;
```

When-New-Item-Instance trigger

Description

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

Definition Level form, block, or item

Legal Commands

SELECT statements, restricted built-ins, unrestricted built-ins.

Enter Query Mode yes

Usage Notes

Use a When-New-Item-Instance trigger to perform an action whenever an item gets input focus. The When-New-Item-Instance trigger is especially useful for calling restricted (navigational) built-ins.

On Failure

no effect

Fires In

Return for Input

See Process Flowcharts

When-New-Item-Instance trigger restrictions

The conditions for firing this trigger are *not* met under the following circumstances:

- Form Builder navigates through an item, without stopping to accept input
- the input focus moves to a field in an alert window, or to any part of an Form Builder menu

When-New-Record-Instance trigger

Description

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 Form Builder is ready to accept input in a record that is different than the record that previously had input focus.

Fires whenever Form Builder instantiates a new record.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-New-Record-Instance trigger to perform an action every time Form Builder instantiates a new record. For example, when an operator presses [Down] to scroll through a set of records, Form Builder fires this trigger each time the input focus moves to the next record, in other words, each time Form Builder instantiates a new record in the block.

On Failure

no effect

Fires In

Return for Input

See Process Flowcharts

When-Radio-Changed trigger

Description

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 using the keyboard.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use a When-Radio-Changed trigger to perform an action depending on the state of a radio group. (De-selecting a radio button in a radio group sets the radio group value to NULL; operators use this technique in Enter Query mode to exclude a radio group from a query.)

When an operator clicks an item in a radio group, the internal value of that item does not change until navigation is completed successfully. Thus, the When-Radio-Changed trigger is the first trigger to register the changed value of a radio group. For all navigation triggers that fire before the When-Radio-Changed trigger, the value of the radio group remains as it was before the operator navigated to it.

On Failure

no effect

When-Remove-Record trigger

Description

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

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a When-Remove-Record trigger to perform an action every time Form Builder clears or deletes a record.

On Failure

Form Builder navigates to the block level with or without validation depending on the current operation, and puts the cursor at the target block.

Fires In

CLEAR_RECORD

DELETE_RECORD

See Process Flowcharts

When-Tab-Page-Changed trigger

Description

Fires whenever there is explicit item or mouse navigation from one tab page to another in a tab canvas.

Definition Level form

Legal Commands

unrestricted built-ins, restricted built-ins

Enter Query Mode no

Usage Notes

- Use a When-Tab-Page-Changed trigger to perform actions when any tab page is changed during item or mouse navigation.
- When-Tab-Page-Changed fires only when tab page navigation is explicit; it does not respond to implicit navigation. For example, the trigger will fire when the mouse or keyboard is used to navigate between tab pages, but the trigger will not fire if an end user presses [Next Item] (Tab) to navigate from one field to another field in the same block, but on different tab pages.
- When-Tab-Page-Changed does not fire when the tab page is changed programmatically.

On Failure

no effect

When-Tab-Page-Changed examples

Example

```
/* Use a When-Tab-Page-Changed trigger to dynamically
** change a tab page's label from lower- to upper-case
** (to indicate to end users if they already have
** navigated to the tab page):
*/
DECLARE
  tp_nm   VARCHAR2(30);
  tp_id   TAB_PAGE;
  tp_lb   VARCHAR2(30);

BEGIN
  tp_nm := GET_CANVAS_PROPERTY('emp_cvs', topmost_tab_page);
  tp_id := FIND_TAB_PAGE(tp_nm);
  tp_lb := GET_TAB_PAGE_PROPERTY(tp_id, label);

  IF tp_lb LIKE 'Sa%' THEN
    SET_TAB_PAGE_PROPERTY(tp_id, label, 'SALARY');
  ELSIF tp_lb LIKE 'Va%' THEN
    SET_TAB_PAGE_PROPERTY(tp_id, label, 'VACATION');
  ELSE null;
```

```
  END IF;
END;
```

When-Timer-Expired trigger

Description

Fires when a timer expires.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

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

- The When-Timer-Expired trigger can not fire during trigger, navigation, or transaction processing.
- Use a When-Timer-Expired trigger to initiate an event, update item values, or perform any task that should occur after a specified interval.
- You can call GET_APPLICATION_PROPERTY(TIMER_NAME) in a When-Timer-Expired trigger to determine the name of the most recently expired timer.

On Failure

no effect

Fires In

Process Expired Timer

See Process Flowcharts

When-Timer-Expired trigger restrictions

A When-Timer-Expired trigger will not fire when the user is currently navigating a menu.

When-Timer-Expired trigger examples

Example

The following example displays a message box each time a repeating timer expires. The following example is from a telemarketing application, in which sales calls are timed, and message boxes are displayed to prompt the salesperson through each stage of the call. The message box is displayed each time a repeating timer expires.

```
DECLARE
  timer_id  TIMER;
  alert_id  ALERT;
```

```
  call_status NUMBER;
  msg_1       VARCHAR2(80) := 'Wrap up the first phase of your
                              presentation';
  msg_2       VARCHAR2(80) := 'Move into your close.';
  msg_3       VARCHAR2(80) := 'Ask for the order or
                              repeat the close.'
  two_minutes NUMBER(6) := (120 * 1000);
  one_and_half NUMBER(5) := (90 * 1000);
BEGIN
  :GLOBAL.timer_count := 1
  BEGIN
    timer_id := FIND_TIMER('tele_timer');
    alert_id := FIND_ALERT('tele_alert');
  IF :GLOBAL.timer_count = 1 THEN
    Set_Alert_Property(alert_id, ALERT_MESSAGE_TEXT, msg_1);
  call_status := Show_Alert(alert_id);
  IF call_status = ALERT_BUTTON1 THEN
    Delete_Timer(timer_id);
    Next_Record;
  ELSIF
    call_status = ALERT_BUTTON2 THEN
    :GLOBAL.timer_count := 0;
  ELSE
    Set_Timer(timer_id, two_minutes, NO_CHANGE);
  END IF;
ELSIF :GLOBAL.timer_count = 2 THEN
    Change_Alert_Message(alert_id, msg_2);
    call_status := Show_Alert(alert_id);
    IF call_status = ALERT_BUTTON1 THEN
      Delete_Timer(timer_id);
      Next_Record;
    ELSIF
      call_status = ALERT_BUTTON2 THEN
      :GLOBAL.timer_count := 0;
    ELSE
      Set_Timer(timer_id, one_and_half, NO_CHANGE);
    END IF;
  ELSE
    Change_Alert_Message(alert_id, msg_3);
    call_status := Show_Alert(alert_id);
    IF call_status = ALERT_BUTTON1 THEN
      Delete_Timer(timer_id);
      Next_Record;
    ELSIF
     call_status = ALERT_BUTTON2 THEN
     :GLOBAL.timer_count := 0;
    ELSE
      Set_Timer(timer_id, NO_CHANGE, NO_REPEAT);
    END IF;
  END IF;
  :GLOBAL.timer_count = 2;
  END;
END;
```

When-Tree-Node-Activated trigger

Description

Fires when an operator double-clicks a node or presses Enter when a node is selected.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

- SYSTEM.TRIGGER_NODE is the node the user clicked on. SYSTEM.TRIGGER_NODE returns a value of type NODE.
- No programmatic action will cause the When-Tree-Node-Activated trigger to fire. Only end-user action will generate an event.

On Failure

no effect

When-Tree-Node-Expanded trigger

Description

Fires when a node is expanded or collapsed.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

- SYSTEM.TRIGGER_NODE is the node the user clicked on. SYSTEM.TRIGGER_NODE returns a value of type NODE.
- No programmatic action will cause the When-Tree-Node-Espanded trigger to fire. Only end-user action will generate an event.

On Failure

no effect

When-Tree-Node-Selected trigger

Description

Fires when a node is selected or deselected.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

- SYSTEM.TRIGGER_NODE is the node the user clicked on. SYSTEM.TRIGGER_NODE returns a value of type NODE.
- No programmatic action will cause the When-Tree-Node-Selected trigger to fire. Only end-user action will generate an event.

On Failure

no effect

When-Validate-Item trigger

Description

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.

Definition Level form, block, or item

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

- Use a When-Validate-Item trigger to supplement Form Builder default item validation processing.
- It is possible to write a When-Validate-Item trigger that changes the value of an item that Form Builder is validating. If validation succeeds, Form Builder marks the changed item as Valid and does not re-validate it. While this behavior is necessary to avoid validation loops, it does make it possible for your application to commit an invalid value to the database.
- The setting you choose for the Defer Required Enforcement property can affect the When-Validate-Item trigger. See Defer_Required_Enforcement for details.

On Failure

If fired as part of validation initiated by navigation, navigation fails, and the focus remains on the original item.

Fires In

Validate the Item

See Process Flowcharts

When-Validate-Item trigger examples

Example

```
The following example finds the commission plan in the COMMPLAN
table, based on the current value of the commcode item in the
EMPLOYEE block in the form, to verify that the code is valid.
If the code in the COMMPLAN table is located, the description of
the COMMPLAN is obtained and deposited in the non-database
Description item. Otherwise, an error is raised.
** Method 1: Using a SELECT...INTO statement, the trigger
**           looks more readable but can be less efficient
**           than Method 2 because for ANSI Standard
**           compliance, the SELECT...INTO statement must
**           return an error if more than one row is
```

```
**           retrieved that matches the criteria. This
**           implies PL/SQL may attempt to fetch data twice
**           from the table in question to insure that there
**           aren't two matching rows.
*/
BEGIN
  SELECT description
    INTO :Employee.Commplan_Desc
    FROM commplan
   WHERE commcode = :Employee.Commcode;
EXCEPTION
  WHEN No.Data_Found THEN
    Message('Invalid Commission Plan, Use <List> for help');
    RAISE Form_trigger_Failure;
  WHEN Too_Many_Rows THEN
    Message('Error. Duplicate entries in COMMPLAN table!');
    RAISE Form_trigger_Failure;
END;

/*
** Method 2: Using an Explicit Cursor looks a bit more
**           daunting but is actually quite simple. The
**           SELECT statement is declared as a named cursor
**           in the DECLARE section and then is OPENed,
**           FETCHed, and CLOSEd in the code explicitly
**           (hence the name). Here we guarantee that only a
**           single FETCH will be performed against the
**           database.
*/
DECLARE
  noneFound BOOLEAN;
  CURSOR cp IS SELECT description
                 FROM commplan
                WHERE commcode = :Employee.Commcode;
BEGIN
  OPEN cp;
  FETCH cp INTO :Employee.Commplan_Desc;
  noneFound := cp%NOTFOUND;
  CLOSE cp;
  IF noneFound THEN
    Message('Invalid Commission Plan, Use <List> for help');
    RAISE Form_trigger_Failure;
  END IF;
END;
```

When-Validate-Record trigger

Description

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.

Definition Level form or block

Legal Commands

SELECT statements, unrestricted built-ins

Enter Query Mode no

Usage Notes

Use a When-Validate-Record trigger to supplement Form Builder default record validation processing.

Note that it is possible to write a When-Validate-Record trigger that changes the value of an item in the record that Form Builder is validating. If validation succeeds, Form Builder marks the record and all of the fields as Valid and does not re-validate. While this behavior is necessary to avoid validation loops, it does make it possible for your application to commit an invalid value to the database.

On Failure

If fired as part of validation initiated by navigation, navigation fails, and the focus remains on the original item.

Fires In

Validate the Record

See Process Flowcharts

When-Validate-Record trigger examples

Example

The following example verifies that Start_Date is less than End_Date. Since these two text items have values that are related, it's more convenient to check the combination of them once at the record level, rather than check each item separately. This code presumes both date items are mandatory and that neither will be NULL.

```
/* Method 1: Hardcode the item names into the trigger.
**           Structured this way, the chance this code will
**           be reusable in other forms we write is pretty
**           low because of dependency on block and item
**           names.
*/
BEGIN
  IF :Experiment.Start_Date > :Experiment.End_Date THEN
    Message('Your date range ends before it starts!');
```

```
    RAISE Form_trigger_Failure;
  END IF;
END;

/* Method 2: Call a generic procedure to check the date
**           range. This way our date check can be used in
**           any validation trigger where we want to check
**           that a starting date in a range comes before
**           the ending date. Another bonus is that with the
**           error message in one standard place, i.e. the
**           procedure, the user will always get a
**           consistent failure message, regardless of the
**           form they're currently in.
*/
BEGIN
  Check_Date_Range(:Experiment.Start_Date,:Experiment.End_Date);
END;

/*
** The procedure looks like this
*/
PROCEDURE Check_Date_Range( d1 DATE, d2 DATE ) IS
BEGIN
  IF d1 > d2 THEN
    Message('Your date range ends before it starts!');
    RAISE Form_trigger_Failure;
  END IF;
END;
```

When-Window-Activated trigger

Description

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 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.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use this trigger to perform the following types of tasks:

- Capture initial settings of window properties, by way of the GET_WINDOW_PROPERTY built-in.
- Enforce navigation to a particular item whenever a window is activated.
- Keep track of the most recently fired window trigger by assigning the value from SYSTEM.EVENT_WINDOW to a variable or global variable.

On Failure

no effect

When-Window-Closed trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use this trigger to programmatically close a window when the operator issues the window-manager Close command.

You can hide the window that contains the current item.

On Failure

no effect

When-Window-Closed trigger examples

Example

The following example of a call to SET_WINDOW_PROPERTY from this trigger closes a window whenever the operator closes it by way of the window manager operation:

```
Set_Window_Property('window_name', VISIBLE, PROPERTY_OFF);
```

When-Window-Deactivated trigger

Description

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

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use this trigger to audit the state of a window whenever the operator deactivates the window by setting the input focus in another window.

Note that if this form opens another form, this deactivate trigger does not immediately fire. Instead, it will fire later when control returns to this form. (Assuming this window also has an activate trigger, then when control returns to this form, first the deactivate trigger fires followed immediately by the activate trigger.)

On Failure

no effect

When-Window-Resized trigger

Description

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.

Definition Level form

Legal Commands

SELECT statements, unrestricted built-ins, restricted built-ins

Enter Query Mode yes

Usage Notes

Use this trigger to perform any one of the following types of tasks:

- Capture the changed window properties, such as width, height, x coordinate, or y coordinate.
- Audit the actions of an operator.
- Set the input focus in an item on the target window.
- Maintain certain visual standards by resetting window size if the window was improperly resized.

On Failure

no effect

Index

M

N

O

P

Q

R

S

T

U

V

W

X

Y