

Events

Activate Event

Applies To

OLE container control.

Description

This event handler section is performed when the corresponding OLE server application is about to be activated in place.

After-Any Event

Applies To

Dialog.

Description

This event handler section is performed after each event in an application and is therefore specified only once. You use it, for example, to modify variables uniformly across the application, or to perform checks. When using this type of event, make sure this does not interrupt the processing of the events that are performed afterwards.

You can use the system variables *CONTROL and *EVENT to determine which dialog element has received which event.

Note:

The after-any event will not occur after a close event.

After-Open Event

Applies To

Dialog.

Description

This event handler section is performed after the dialog with all child dialog elements has been created and displayed. You can use it, for example, to create dialog elements dynamically by using the ADD action of the PROCESS GUI statement in its event handler code. You can also use it to add items to list-box controls or selection-box controls with the corresponding PROCESS GUI statement action.

Before-Any Event

Applies To

Dialog.

Description

This event handler section is performed before each event in an application and is therefore specified only once. You use it, for example, to modify variables uniformly across the application, or to perform checks. When using this type of event, make sure this does not interrupt the processing of the event handlers that are performed afterwards.

Before-Open Event

Applies To

Dialog, context menu, submenu control.

Description

Occurs before the dialog is created and displayed. You can use it, for example, to load profile information from a database. For context menus and submenus, the event occurs immediately before the menu is displayed, allowing menu items to be checked, disabled, etc., according to the current context.

Note:

As the dialog elements are created after this dialog event, all handle variables except #DLG\$PARENT are NULL. Note also that in the case of dialogs, this event cannot be suppressed, although for context menus and submenus it can be.

Change Event

Applies To

Edit area control, input-field control, OLE container control, scroll bar control, selection-box control, table control.

Description

Occurs whenever the `STRING` attribute value of a dialog element has changed, for example, as a result of keyboard input or as a result of a dynamic change at runtime. Occurs for a scroll bar control when the `SLIDER` attribute value changes. Occurs for an OLE container control when the object's data have been changed. The event handler associated with this event would typically be used to enable other dialog elements when an input-field was filled (such as in an input-field control or in a selection-box control).

If this event occurs in a cell of a table control, the `ROW` and `COLUMN` attributes are modified so that they contain the position of the changed cell.

If this event occurs for an input-field control or a selection-box control with a linked variable, the linked variable must be updated in the change event.

Click Event

Applies To

Bitmap control, list-box control, menu item, OLE container control, push button control, radio-button control, table control, toggle-button control, tool-bar item.

Description

Occurs whenever the end user selects the dialog element by a keyboard function or by a mouse click: a push-button control may be pressed, a list box item may be selected, or a toggle-button control may be checked. It is the most important event because the end user may trigger business logic with it (press a push-button control to trigger a calculation or to validate data that have been entered, and so on).

Clicking a dialog element always selects the dialog element, but it does not necessarily cause an action to occur. A click event handler section is therefore only relevant if you want an action to occur.

If this event occurs in a cell of a table control, the `ROW` and `COLUMN` attributes are modified so that they contain the position of the cell that has been clicked on.

Client-Size Event

Applies To

Dialog.

Description

Occurs whenever the interior size of the dialog window changes, for example, if a tool-bar control is re-docked at a different location. The dialog interior is the part of the dialog which does not include the menu bar (if any), tool bars (if any) or status-bar (if any). You can use the client-size event handler to size controls in the dialog window relative to the dialog window's size.

Close Event

Applies To

Dialog, OLE container control.

Description

Occurs whenever the end user selects the Close option on the Natural menu to close a dialog window or whenever the dialog is closed using the CLOSE DIALOG statement or whenever an OLE container application is closed. It can be used to make the associated event handler check if all work has been saved and ask the user to save and close, exit and close, or cancel. When a dialog is closed, all child dialogs are also closed. In this case, the close event handlers of the child dialogs are not triggered. If you want to close the parent dialog and trigger the close event handlers in the child dialogs as well, you use the subprogram NGU-DIALOG-CLOSE-ALL.

Command-Status Event

Applies To

Dialog.

Description

This event occurs whenever one or more commands (signals, menu items or tool bar items without a SAME-AS attribute, or status bar panes) need to be updated (i.e., disabled, checked, etc.) by a dialog during idle processing or in response to an explicit UPDATE-COMMAND-STATUS action. The commands do not necessarily belong to the dialog receiving the command-status event if their SHARED attribute is set to TRUE, which causes MDI frame commands to be automatically redirected to the active MDI child dialog (if any). This is precisely the case where the command-status event is most useful, because it allows multiple instances of an MDI child dialog (or even completely different MDI child dialogs) to share the same tool bar and status bar controls without interfering with each other, as would be the case if each MDI child would attempt to update the commands directly.

The intended usage of the command-status event is to only update state variables (and not the commands themselves) when the program state changes, and then to do a bulk update of the commands according to the current values of the state variables within the command-status event handler.

Note that the command-status event handler is called frequently, and therefore should return as quickly as possible. For this reason, database access (for example) should be avoided in this event.

If there are no commands which need to be serviced by a particular dialog, no command-status event is raised, even if there is code available for it and the event is not suppressed. This is the case for an active MDI child if no commands have been marked as shared, for example.

For performance reasons, it is not possible for the application to find out which commands caused a particular command-status event to be triggered. Instead, a dialog should update all commands for which it is the current target, and which require a non-default state to be set. The default state is disabled and unchecked for signals, menu items and tool bar items and invisible for status bar panes. Any commands not explicitly enabled or disabled (or any status bar panes not explicitly shown or hidden) by the program in the command-status event will be automatically reset to the default state by the system. This allows a particular MDI child dialog, for example, to only set the status of commands it "knows" about, and to let the system implicitly reset the commands intended only for different child dialogs. Thus, if a command is introduced for a specific MDI child dialog only, the command-status events of the all other MDI child dialogs do not need to be modified. This automatic command resetting is not performed if the corresponding dialog's command-status event is suppressed.

The command-status event for a particular dialog is raised before the idle event (if any) for that dialog, in order that the effects of the idle event code can be taken into account by the command status updating process.

DDE-Client Event

Applies To

Dialog.

Description

Occurs when your dialog is acting as a client of a DDE conversation and receives a DDE message from the DDE server application, for example Microsoft Excel. The field `DDE-VIEW.MESSAGE` (or its equivalent) contains one of the following values:

- DISCONNECT
- DATA
- NOTIFY
- TIMEOUT

DDE-Server Event

Applies To

Dialog.

Description

Occurs when your dialog is acting as a client of a DDE conversation and receives a DDE message from a DDE client application, for example Microsoft Excel. The field DDE-VIEW.MESSAGE (or its equivalent) contains one of the following values:

- CONNECT
- DISCONNECT
- REQUEST
- ADVISE
- UNADVISE
- POKE
- EXECUTE
- TIMEOUT

Default Event

Applies To

Any dynamically created dialog element.

Description

Occurs whenever a non-suppressed event occurs for which no event handler section is specified. Also occurs whenever the end user triggers an event in a dynamically created dialog element. The default event handler section can be used to DECIDE which event (value of *EVENT) occurred for which dialog element (value of *CONTROL). For each particular dialog element and event, you can then specify the individual event handler code section.

A default event occurs, for example, if you issue SEND EVENT 'HUGO', but no such 'HUGO' event has been defined. Another example is that the event is not suppressed, but the event handler section is empty.

Delete-Row Event

Applies To

Table control.

Description

Occurs after the end user has deleted a row by selecting it and by pressing DEL or by calling "PROCESS GUI ACTION TABLE-DELETE-ROW ...". The table control's STYLE attribute must have had the value "e" (extendable, allows end users to delete rows).

If this event occurs in a cell, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

Double-Click Event

Applies To

Bitmap control, canvas control, list-box control, OLE container control, table control.

Description

Occurs whenever the end user double-clicks on the dialog element. For list-box controls and radio-button controls, the double-click event normally triggers the event handler with the business logic, whereas the click event would normally trigger an event handler with a simple display function.

The first click of a double-click triggers a click event, except for when the click event is suppressed. In the context of list-box controls and radio-button controls, the first click of the double-click selects a list box item or a radio-button control. The second click should be equivalent to pressing the default button in the corresponding window.

If this event occurs in a cell of a table control, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

If this event occurs inside the rectangle of an OLE container control, the OLE server application is started. Double-clicking inside an OLE container control is equivalent to clicking with the right mouse button. (Clicking the right mouse button inside the OLE container control's rectangle invokes a pop-up menu. By default, releasing the mouse button without further selection executes the first command verb of the OLE server application.)

Drag-Drop Event

Applies To

Bitmap control.

Description

Whenever the end user points to a bitmap control with the mouse, holds the mouse button down, drags it onto a target bitmap control, this event occurs for the target bitmap control on releasing the mouse button (dropping the dragged bitmap control). In the event handler code of this event, you can use the PROCESS GUI statement action INQ-DRAG-DROP to find out where the bitmap control was dropped.

Enter Event

Applies To

Dialog, edit area control, input-field control, selection-box control, table control.

Description

Occurs whenever the end user changes the focus by clicking on the dialog element or by using TAB.

For dialogs, this event occurs when a dialog becomes active. This happens when

- the end user activates it by a mouse click or by a keyboard operation; or
- the PROCESS GUI statement action SET-FOCUS is applied to this dialog.

For dialog elements, this event occurs when the dialog element receives the focus by end user action or if the SET-FOCUS action is applied.

For table controls, this event occurs if the table receives the focus or if the end user enters a new *row*. If the end user enters a new *cell*, the enter-cell event occurs.

If this event occurs in a cell of a table control, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

When an end user leaves an MDI application, for example, by clicking into another application outside Natural, and clicks into the MDI application again, an enter event only occurs for the MDI Frame dialog, not for the MDI Child dialogs.

Enter-Cell Event

Applies To

Table control.

Description

Occurs whenever the end user changes the focus by clicking on a cell in a table control, by using TAB to enter a cell, or by using the arrow keys to enter a cell.

If this event occurs, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

Error Event

Applies To

Dialog.

Description

This event handler section is performed whenever a runtime error occurs while a dialog is active. You can specify event handler code to be executed whenever this error occurs. If no error event handler code is specified, Natural will terminate with an error message and all dialogs will be closed. If an error event handler code section was specified, the dialog will remain open and handle the error.

You can use the system variables *CONTROL and *EVENT to determine the event handler in which the error occurred. If you want the dialog to continue after processing the error, code an ESCAPE ROUTINE statement at the end of the error event handler.

Fill Event

Applies To

List-box control, table control.

Description

Occurs for list-box controls whenever the end user scrolls to the end of a list box control that has scroll bars. Occurs for table controls whenever the end user navigates to the last row in the table control or scrolls to the end of the table control with the scroll bar. If all items in the list-box control are visible without scrolling, there is no scroll bar and therefore no fill event will occur. In this event, you would usually append additional items to the end of the list-box control with the PROCESS GUI statement action ADD-ITEMS.

If this event occurs in a cell of a table control, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

Idle Event

Applies To

Dialog.

Description

Occurs during idle-time processing in situations where the user interface may need to be updated, which is after each time a key or mouse button is pressed or released. Note that mere mouse movement does not normally cause idle events to be raised, because this would be too slow. An exception is when the mouse pointer is moved away from a tool bar, in order to allow the program to update the status-bar text.

In addition to allowing the program to update the status-bar with an idle message (e.g., "Ready"), the idle event may also be used for monitoring user interface changes. For example, the idle event of an MDI frame dialog can query the active MDI child dialog and show or hide individual tool bars appropriately if necessary.

Note that the idle event handler is called frequently, and therefore should return as quickly as possible. For this reason, database access (for example) should be avoided in this event.

Insert-Row Event

Applies To

Table control.

Description

Occurs after the end user has inserted a row by pressing INS or by calling "PROCESS GUI ACTION TABLE-INSERT-ROW ...". The table control's STYLE attribute must have had the value "e" (extendable, allows end users to insert rows).

If this event occurs in a cell, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

Leave Event

Applies To

Dialog, edit area control, input-field control, selection-box control, table control.

Description

Occurs whenever the dialog or dialog element loses the focus. You can use the leave event handler to validate field entries or to dynamically modify other dialog elements whose status depends on that of the dialog element just left. Does not occur on closing a dialog.

For table controls, this event occurs if the end user leaves a *row* or if the table loses the focus. If the end user leaves a *cell*, the leave-cell event occurs.

If this event occurs in a cell of a table control, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

Before this event handler code is processed, end user input in (linked variable) input-field controls and selection-box controls is checked against any EDIT-MASK value or applicable Natural data type. However, in the following situations the leave event is *not* triggered after end user input was checked and a linked variable was updated:

- An input-field control or a selection-box control has the focus and the end user clicks a menu item; or
- the end user presses ENTER, triggering the default button; or
- the end user presses ENTER, triggering the OK button; or
- the end user presses an accelerator key.

Note:

It is not recommended to execute the PROCESS GUI statement action SET-FOCUS from the leave event because the leave event implies that the focus is to be set to yet another part of the application.

Leave-Cell Event

Applies To

Table control.

Description

Occurs whenever the end user moves the focus away from a cell (for example, by clicking on another cell in the table control or by using TAB to enter another cell).

If this event occurs in a cell, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

Size Event

Applies To

Dialog.

Description

Occurs whenever the size of the dialog window changes, for example, if the window is minimized or maximized. You can use the size event handler to size controls in the dialog window relative to the dialog window's size.

Top Event

Applies To

Table control.

Description

Occurs whenever the end user navigates to the top of the table control (either by using the scroll bar or by entering a cell of the top row).

If this event occurs in a cell, the ROW and COLUMN attributes are modified so that they contain the position of the cell.

User-Defined Events

Apply To

Dialog.

Description

User-defined events occur whenever you have specified a SEND EVENT statement with the name of a previously undefined event in an event handler code of a dialog. Note that the user event must be defined in another dialog than the one where it occurs. For more details and examples of how to cause an action to occur in another dialog, see the section Event-driven Programming Techniques of your Natural User's Guide for Windows.

You can define user events for dialogs by pressing the "New" button in the "Events..." dialog box of the dialog's attributes window. You can then enter any name of your newly-defined user event and specify the corresponding event section. It is recommend that you use "#" as the first letter to distinguish user-defined events from previously defined events.

If no event handler is defined for an event generated by the SEND EVENT statement, the default event handler is executed with the value of the system variable *EVENT as the user event name.