Editing Dialogs

The following topics are covered below:

- Editing a Dialog’s Source Code
- Editing a Dialog’s Attributes
- Editing a Dialog’s Event Handlers
- Defining a Dialog’s Menu Bar
- Defining a Dialog’s Toolbar
- Creating and Maintaining Timers for a Dialog
- Creating and Maintaining Signals for a Dialog
- Creating and Maintaining Context Menus
- Creating and Maintaining Wallpapers
- Adding a Comment Section to a Dialog
- Defining a Parameter or Local Data Area for a Dialog
- Selecting a Global Data Area for a Dialog
- Defining an Inline Subroutine for a Dialog
- Defining and Maintaining Help Text for a Dialog
- Defining the Control Sequence in a Dialog

Editing a Dialog’s Source Code

To edit a dialog’s source code

1. Load the dialog into the editor.
2. From the **Dialog** menu, choose **Source Code**.

   Or:
   Press CTRL+ALT+C.

   The dialog’s source code window appears and the program editor is loaded. This editor enables you to scan for text strings, replace them, and so on. For more information on how to use the program editor, refer to the *Program Editor*.

   You can switch between the dialog editor and the program editor by selecting the source code window or the dialog window. If you edit in either window, you need to synchronize your updates: (graphically) modifying the dialog locks the source code window and you may not make changes

there. Correspondingly, if you change the source code, you may not make changes in the dialog window, which is locked. If your editor is locked, its status bar displays "Locked".

If a source code window is open, but not active, you can activate it by choosing **Source Code** from the **Dialog** menu.

When you issue a command from the program editor window that affects the source code, such as **Save** or **Run**, the dialog editor updates itself automatically by scanning the source code, displaying the modified dialog, and then regenerating the source code.When you issue a command from the dialog editor window after you have modified the code in the source code window, you are prompted whether you want to update the source code or not.

3. To stow any modified source code: from the program editor’s **Object** menu, choose **Stow**.

Whenever you want to save a dialog under a new name, select **Save as** from the **Object** menu, where for the Save dialog as dialog box appears.

Editing a Dialog’s Attributes

To edit a dialog’s attributes

1. Load the dialog into the editor.
2. From the **Dialog** menu or from the dialog’s context menu, choose **Attributes**.

   Or:
   Double-click on the dialog.

   Or:
   Press ENTER.

   The dialog’s attributes window appears. To find out what the entries in the attributes window mean, choose "Help". For context-sensitive help, select the attribute entry and press F1.
3. Enter the desired attribute values.
4. Choose **OK** to confirm your changes.

Editing a Dialog’s Event Handlers

To edit a dialog’s event handlers

1. Load the dialog into the editor.
2. From the **Dialog** menu or from the dialog’s context menu, choose **Event handlers**.

   Or:
   Press CTRL+ALT+E or SHIFT+ENTER.

   The dialog’s event handler section appears.

3. Select the type of event (such as BEFORE-OPEN or ERROR).

   Or:
   Choose "New" to enter a user-defined event.

   Or:
   Choose "Rename" to save a user-defined event with a new name.

4. Enter the desired event code in free form either in the edit window in the Dialog Event Handler window itself, or using the Program Editor. To use the Program Editor, select the **Editor** push button, then close the Dialog Event Handler window using the **OK** push button. This code will be executed when the event occurs for the dialog. Note that if you have specified code in the before-any and after-any event sections, this will be triggered before and after the code entered here. So if you need common event code, you only have to enter it once in your dialog's before-any and after-any event section.

5. Choose **OK** to save your code, if using the Dialog Event Handler window. If using the Program Editor, the code can be saved by choosing **Save** from the **Object** menu, or by closing the Program Editor and selecting to save the changes when prompted.

Defining a Dialog's Menu Bar

To define a dialog's menu bar

1. Load the dialog into the editor.

2. From the **Dialog** menu, choose **Menu bar**.

   Or:

   Press CTRL+ALT+M.

   A dialog box appears asking you whether you want to turn the dialog's menu bar setting on.

3. Choose **Yes**.

   A blank default menu bar is added to the dialog and the menu bar's attributes window appears. For more information on the attributes window, see the section *Menu Editor Window*.

4. Choose **OK** to confirm your changes.

Defining a Dialog's Toolbar

To define a dialog's toolbar

1. Load the dialog into the editor.

2. From the **Dialog** menu, choose **Toolbar**.

   Or:
   Press CTRL+ALT+T.

A dialog box appears asking you whether you want to turn the dialog's toolbar setting on.

3. Choose **Yes**.

   A blank default toolbar is added to the dialog and the toolbar's attributes window appears. For more information on the attributes window, see the section *Toolbar Control Attributes Window*. In this section you will also find information on new toolbar control features.

4. Choose **OK** to confirm your changes.

Creating and Maintaining Timers for a Dialog

You use timers to trigger a dialog event periodically.

To create and maintain a timer for a dialog

1. Load the dialog into the editor.
2. From the **Dialog** menu, choose **Timers**.

   Or:
   Press CTRL+ALT+I.

   The timer's attributes window appears. For more information on the attributes window, see the section *Timer Attributes Window*.

3. Choose **OK** to confirm your changes.

Creating and Maintaining Signals for a Dialog

You use signals to efficiently implement user commands, allowing their re-use by multiple user interface elements (such as menu or toolbar items).

To create and maintain signals for a dialog

1. Load the dialog into the editor.
2. From the **Dialog** menu, choose **Signals**.

   Or:
   Press CTRL+ALT+N.

   The signal attributes window appears. For more information on the attributes window, see the section *Signal Attributes Window*.

3. Choose **OK** to confirm your changes.
4. To associate a menu or toolbar item with the signal, select the signal from the list presented in the **Same as** field for the item in the menu editor or toolbar attributes window. The menu or toolbar item will then inherit the attributes from the signal. Furthermore, when the menu or toolbar item is clicked on running the dialog, the signal's `CLICK` event is invoked.

Creating and Maintaining Context Menus

You use context menus to define context-specific menus that can be associated with the dialog and/or any number of dialog elements within it. For more information, see the section *Defining and Using Context Menus*.

To create and maintain context menus

1. Load the dialog into the editor.
2. From the **Dialog** menu, choose **Context Menus**.

   Or:
   Press CTRL+ALT+X.

   The context menus window appears, which displays a list of context menus currently defined for the dialog. For more information on this window, see the section *Dialog Context Menus Window*. To edit the menu items for a context menu listed in this window, select the respective context menu in the list, then select the **Edit** pushbutton to invoke the Menu Editor Window.
3. Choose **OK** to confirm your changes.
4. To associate the context menu with the dialog or a dialog element within the dialog, open the corresponding attributes window (e.g. by double-clicking on the dialog or dialog element) and select the context menu from the list presented in the **Context Menu** field. Note, however, that not all dialog element types support context menus.

Creating and Maintaining Wallpapers

You use wallpapers to define background images that can be associated with the dialog and/or any number of dialog elements within it.

To create and maintain wallpapers

1. Load the dialog into the editor.
2. From the **Dialog** menu, choose **Dialog Wallpapers**.

   Or:
   Press CTRL+ALT+W.

   The wallpaper attributes window appears. For more information on the attributes window, see the section *Wallpaper Attributes Window*.
3. Choose **OK** to confirm your changes.
4. To associate the wallpaper with the dialog or a dialog element within the dialog, open the corresponding attributes window (e.g. by double-clicking on the dialog or dialog element) and select the wallpaper from the list presented in the **Wallpaper** field. Note, however, that not all dialog element types support wallpapers.

Adding a Comment Section to a Dialog

To add a comment section to a dialog

1. From the **Dialog** menu, choose **Dialog comment**.

   Or:
   Press CTRL+ALT+O.

   The dialog comment section appears where you can enter your comments in free form. Please note that you do not have to use the "/*" notation when entering comments in the text area. If you are listing your dialog code, you will find your comment at the beginning.

2. Choose **OK** to save your comment.

Defining a Parameter or Local Data Area for a Dialog

To define a local data area for a dialog

1. From the **Dialog** menu or from the dialog's context menu, choose **Local Data Area**.

   Or:
   Press CTRL+ALT+L.

   The definition section for the local data area appears. In a local data area, you must include all the user-defined variables or other variables that you want to use in an event handler code section or a subroutine of the current dialog. Note that the dialog editor automatically generates the data definitions for the dialog elements.

   The **Using** button opens a dialog box that enables you to include existing inline data definitions.

2. Choose **OK** to save your data definition.

To define a parameter data area for a dialog

1. From the **Dialog** menu, choose **Parameter Data Area**.

   Or:
   Press CTRL+ALT+P.

   The definition section for the parameter data area appears. In a parameter data area, you must include all the parameters that you want to be passed on to the current dialog in an `OPEN DIALOG` or `SEND EVENT` statement.

   The **Using** button opens a dialog box that enables you to include existing inline data definitions.

2. Choose **OK** to save your data definition.

Selecting a Global Data Area for a Dialog

To select a global data area for a dialog

1. From the **Dialog** menu, choose **Global Data Area**.

   Or:
   Press CTRL+ALT+G.

   A dialog box appears where you can select a global data area for the dialog.

2. Select an entry in the **Available Global Data Areas** list box.

3. Choose **OK**.

Defining an Inline Subroutine for a Dialog

To define an inline subroutine for a dialog

1. Load the dialog into the editor.

2. From the **Dialog** menu or from the dialog's context menu, choose **Inline Subroutines**.

   Or:
   Press CTRL+ALT+S.

   The "Dialog inline subroutines" code section appears.

3. Choose "New" to enter a new subroutine.

   Or:
   Select the name of an existing subroutine you want to edit.

   If you have chosen "New", a dialog box prompts you for a name.

4. Enter the name of the new subroutine.

5. Choose **OK**.

6. Enter the desired subroutine code in free form, either directly in the window itself or using the Program Editor by selecting the **Editor** push button, then close the window using the **OK** push button.

7. Choose **OK** to save your code.

Defining and Maintaining Help Text for a Dialog

Natural contains an integrated Help Organizer, which allows the definition of simple (e.g., pop-up) help text for a dialog. In order to create more elaborate and/or HTML-based help content, a third-party help authoring tool will be required. For more information on the Help Organizer, see *Organizing an Application's Help File*.

To define help text for a dialog

1. Load the dialog into the editor.
2. From the **Dialog** menu or from the dialog's context menu, choose **Help Organizer**.

   Or:
   Press CTRL+ALT+H.

   The help organizer's main window appears. For more information, see *Using the Help Organizer's Main Dialog*.
3. Choose **OK** or **Apply** to confirm your changes.
4. To generate the help, load the generated help project (.hpj) file into the Microsoft Help Compiler Workbench (HCW.EXE) and compile the help project. The resulting help (.hlp) file can be deployed in the RES subdirectory of the logon library, the RES directory of one of the steplibs, or the directory assigned to the environment variable `NATGUI_BMP`.
5. To associate the compiled help file with the dialog, select the file in the **Help file** field in the dialog attributes window. Select the **Help button** option in the dialog attributes window if you wish a help (question mark) button to appear in the window's title bar (cannot be used in conjunction with the minimize or maximize buttons). Select the **Popup help** option in the dialog attributes window if you wish the help to be displayed in a small tooltip-style window instead of a full-blown help window.
6. To invoke the help when the dialog is running, either press F1 to display the help corresponding to the dialog element with the focus, or select the help button (if available) then select the desired dialog element (or the dialog itself). Note that if the dialog element has a help ID of zero, the help ID of its parent is used to identify the help topic to be displayed.

Defining the Control Sequence in a Dialog

The control sequence is the keyboard navigation sequence in which the end user will go through the dialog elements.

To define the control sequence of a dialog

1. From the **Dialog** menu, choose **Control sequence**.

   Or:
   Press CTRL+ALT+Q.

   The control sequence is displayed as a number at the top left corner of each dialog element. The editor is now in navigation sequence definition mode.
2. Use the mouse to select the dialog elements in the desired sequence.

If you *do not* select a dialog element before enabling navigation sequence definition mode, the next dialog element that you select will be the first in the navigation sequence. Its number is greyed and you can select the next dialog element in the sequence, and so on.

If you *do* select a dialog element, you can redefine the sequence from this element onwards.You can also select a dialog element when in control sequence editing mode *without* resequencing it by holding down the SHIFT key whilst making the selection. This is especially useful if the selected dialog element is one of the last elements in the sequence - you do not have to redefine the sequence of all preceding dialog elements. Note that instead of selecting each dialog element with the mouse, you can also select them from the selection box in the status bar of the dialog editor. This selection box always shows the dialog elements in their control sequence.

You can exit control sequence editing mode implicitly, by selecting another command (e.g. ’Insert Push Button’) or explicitly by selecting the ’control sequence’ menu of this again, or by pressing ESC.

Note:
The control sequence also decides the order in which the dialog elements overlap.