How To Open and Close Dialogs

This document covers the following topics:

Opening a Dialog

An event-driven application is started by executing the base dialog. Events triggered by the end user will then typically cause other dialogs to be started. The application ends when the base dialog is closed.

To open a dialog from anywhere within an event-driven application

- Use the statement `OPEN DIALOG`.

This statement causes the dialog to be loaded and the processing on its opening to be performed.

Control over processing returns from the opened dialog except for dialogs with the style "Dialog Box". For those dialog styles, control returns only after the dialog has ended.

The parameters passed are accessible only during the processing on the opening of a dialog (before-open and after-open events), except for when the parameters are declared as `BY VALUE` in the parameter data area of the opened dialog or when the dialog has the style "Dialog Box".

To open a dialog from anywhere within an event-driven Natural application, the following syntax is used:

```
OPEN DIALOG operand1    [USING]    [PARENT] operand2
                      [ [GIVING]   [DIALOG-ID] operand3 ]
                      [  WITH     { operand4...          } ]
                      [           { PARAMETERS-clause    } ]
```

Operands

Operand1 is the name of the dialog to be opened. If the *PARAMETERS-clause* is used, *operand1* must be a constant (the name of a cataloged dialog).

Operand2 is the handle name of the parent.

Operand3 is a unique dialog ID returned from the creation of the dialog. It must be defined with format/length I4.

Passing Parameters to the Dialog

When a dialog is opened, parameters may be passed to this dialog.

As *operand4* you specify the parameters that are passed to the dialog.

With the *PARAMETERS-clause*, parameters may be passed selectively:

PARAMETERS [*parameter-name=operand 4*] _ **END-PARAMETERS**

Anmerkung:
You may only use the *PARAMETERS-clause* if *operand1* is an alphanumeric constant and if the dialog is cataloged.

Parameter-name is the name of the parameter as defined in the parameter data area section of the dialog.

To avoid format/length conflicts between operands and parameters passed, see the `BY VALUE` option of the `DEFINE DATA` statement in the *Statements* documentation.

When passing parameters only with *operand4*, a dialog may be opened as follows:

```
/* The following parameters are defined in the calling dialog’s parameter
/* data area (not in the parameter data area of the dialog to be opened):
 1 #MYDIALOG-ID (I4)
 1 #MYPARM1 (A10)
/* Pass the operands #MYPARM1 and ’MYPARM2’ to the parameters #DLG-PARM1 and
/* #DLG-PARM2 defined in the dialog to be opened:
  OPEN DIALOG ’MYDIALOG’ USING
  #DLG$WINDOW GIVING
  #MYDIALOG-ID WITH
  #MYPARM1 ’MYPARM2’
```

When passing parameters selectively with the *PARAMETERS-clause*, a dialog may be opened as shown in the following example:

```
/* The following parameters are defined in the calling dialog’s parameter
/* data area (not in the parameter data area of the dialog to be opened):
 1 #MYDIALOG-ID (I4)
 1 #MYPARM1 (A10)
/* Pass the operands #MYPARM1 and ’MYPARM2’ to the parameters #DLG-PARM1 and
/* #DLG-PARM2 defined in the dialog to be opened:
  OPEN DIALOG ’MYDIALOG’ USING
  #DLG$WINDOW GIVING
  #MYDIALOG-ID WITH PARAMETERS
  #DLG-PARM1=#MYPARM1
  #DLG-PARM2=’MYPARM2’
END-PARAMETERS
```

Permanence in Creating, Passing and Checking Data

The term "permanence" is used in Natural to denote data defined in a base dialog’s local data area whose existence is guaranteed throughout the whole lifetime of the dialog. Data defined in the global data area are not kept permanent because the global data area can be exchanged while the application is executed.

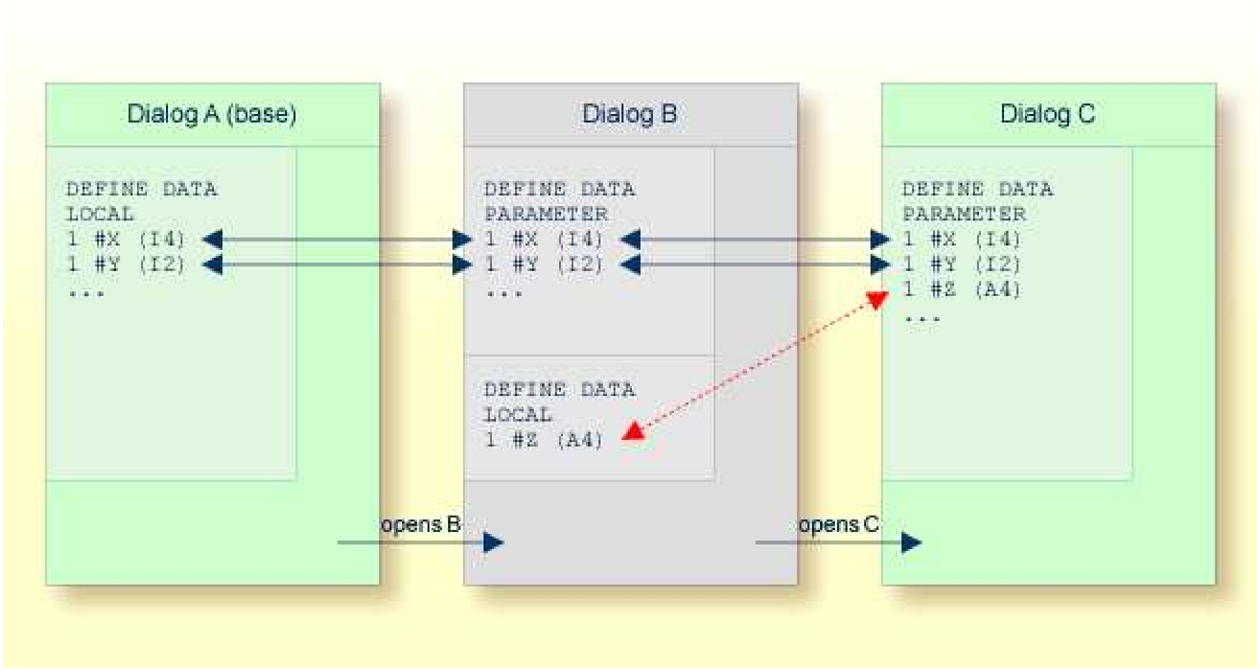
The reference to the permanent data is kept by saving the parameter data area internally during opening of the dialog. This reference is reused when

- a dialog element receives an event;
- all parameters passed from one dialog to another are permanent, provided they reference the base dialog’s local data area.

Parameters are accessible

- during the before-open and after-open event processing on opening of a dialog or
- if *all of them* reference the base dialog’s local data area.

The following example illustrates a case in which two parameters are kept permanently and one other is not. Assume the base dialog is dialog A. This base dialog now opens dialog B, passing parameters #X and #Y. After that, dialog B passes parameters #X and #Y on to dialog C. The #X and #Y parameters which are now in dialog C will be permanent, even if dialog B is closed. If, however, dialog B passes its own parameter #Z when opening dialog C, the parameter #Z is not permanent, because if dialog B is closed, the reference to its local data area is no longer valid. No parameter in dialog C is accessible (#Z does not reference the base dialog’s local data area).



Processing Steps When Opening a Dialog

This section describes what happens when a dialog is opening. You can open a dialog either by executing it, for example from the command line, or by invoking it with an `OPEN DIALOG` statement.

- The dialog object is loaded and starts executing.
- The `BEFORE-ANY` event-handler section is executed, the value of the system variable `*EVENT` being `OPEN`.
- The `BEFORE-OPEN` event-handler section is executed.
- The dialog window is created as specified in the dialog editor.
- The `BEFORE-ANY` event-handler section is executed. `*EVENT` = `AFTER-OPEN`.
- All dialog elements are created as specified in the dialog editor.
- The dialog window and all dialogs are made visible except those that are `VISIBLE = FALSE`.
- The `AFTER-OPEN` event-handler section is executed.
- The `AFTER-ANY` event-handler section is executed. `*EVENT` = `AFTER-OPEN`.
- The `AFTER-ANY` event-handler section is executed. `*EVENT` = `OPEN` (not if the dialog’s `STYLE` attribute value is "Dialog Box").

Closing Dialogs

To close a dialog dynamically, you specify the following:

```
CLOSE DIALOG [USING] [DIALOG-ID] { operand1   }
                                 { *DIALOG-ID }
```

Operand1 is the identifier of the dialog as returned in the `OPEN DIALOG` statement.

Example:

```
CLOSE DIALOG *DIALOG-ID /* Close the current Dialog
```

The dialog will then be erased from the screen and removed from memory. All local data associated with the dialog will be gone.

Anmerkung:
If a modal dialog is a child in a hierarchy of dialogs, the modal dialog should not close its parent(s) because this will result in a deadlock.

operand1

Operand1 is the name of the dialog to be closed.

To close the current dialog, you specify `*DIALOG-ID`.

Initializing Attribute Values

You can specify conditions for the opening and closing of a dialog: this applies to the before-open, after-open, and close events. These conditions can be used to initialize the attribute values in the dialog.

The following is an example of after-open event-handler code: Red foreground color is assigned to push buttons that the user must press after entering data in the associated input fields.

```
DEFINE DATA LOCAL
...
  1  #OK-BUTTON HANDLE OF PUSHBUTTON
  1 #CALC-BUTTON HANDLE OF PUSHBUTTON
  1 #SAVE-BUTTON HANDLE OF PUSHBUTTON
  1 #CONVERT-BUTTON
HANDLE OF PUSHBUTTON
...
END-DEFINE
...
#OK-BUTTON.FOREGROUND-COLOUR-NAME := RED
#CALC-BUTTON.FOREGROUND-COLOUR-NAME := RED
#SAVE-BUTTON.FOREGROUND-COLOUR-NAME := RED
#CONVERT-BUTTON.FOREGROUND-COLOUR-NAME := RED
```

If you want to modify attribute values of dialog elements and of the dialog before the dialog is opened (and displayed to the end user), do not specify this in the "before open" event-handler code, because the dialog elements and the dialog window are not yet created. Instead, create the dialog with the dialog editor and set the attribute `VISIBLE` to `FALSE` in the Dialog Attributes window. Then modify all the attribute values in the after-open event-handler code (when the handles are available). Then make the dialog visible with `VISIBLE = TRUE`.

Example:

```
DEFINE DATA LOCAL
...
 1 #DIA-1 HANDLE OF DIALOG
 1 #OK-BUTTON HANDLE OF PUSHBUTTON
 1 #CALC-BUTTON HANDLE OF PUSHBUTTON
...
END-DEFINE
...
/* AFTER OPEN event-handler code section
...
#OK-BUTTON.FOREGROUND-COLOUR-NAME := RED
#CALC-BUTTON.FOREGROUND-COLOUR-NAME := RED
#DIA-1.VISIBLE := TRUE
```