

Chapter 29: Processing User Requests

Overview

In a transaction, the user makes requests by selecting screen pushbuttons, menu functions, function keys, tool-bar pushbuttons, icons or the **ENTER** key.

How does a dialog program handle user requests?

When an action is performed, the system triggers the **PROCESS AFTER INPUT** event. The data passed includes field screen data entered by the user and a function code. A function code is a technical name that has been allocated in the Screen Painter or Menu Painter to a menu entry, a pushbutton, the **ENTER** key or a function key of a screen. An internal work field (ok-code) in the PAI module evaluates the function code, and the appropriate action is taken.

The following sections use the TZ20 transaction (development class SDWA) to show how you can use pushbuttons to control the way a transaction executes.

Flight data	
From city	NEW YORK
Dep. airport	JFK
Destination	SAN FRANCISCO
Dest. airport	SFO
Flight time	06:01:00
Departure	13:30:00
Arrival	16:31:00
Distance	2.572
Dist. in	MLS

The final two sections of this chapter explain how radio buttons and check boxes can be used.

The following topics explain how you can include user functions in your transaction:

Programming with Function codes

Programming with Radio Buttons

Programming with Check Boxes

Contents

Programming with Function codes	29–3
Setting up Function codes	29–3
Handling Function codes	29–5
Handling Field Selections	29–6
Sharing GUI-Statuses	29–8
Programming with Radio Buttons	29–8
Programming with Check Boxes	29–9

Programming with Function codes

The following topics guide you in using function codes:

Setting up Function codes
Handling Function codes
Handling Field Selections
Sharing GUI-Statuses

Setting up Function codes

To handle user requests in a dialog program, you must assign function codes to the relevant screen and window elements in the Screen Painter or the Menu Painter.

- In the Screen Painter:
 - screen pushbuttons
- In the Menu Painter:
 - menu functions
 - function keys
 - tool-bar pushbuttons and icons
 - the **ENTER** key

Screen Painter

In the Screen Painter, you can assign a function code by setting the *FctCode* attribute for a pushbutton field.

The function code (FctCode) FTCH has been set in the Screen Painter for the *Display* pushbutton in the TZ20 transaction.

Screen Field Attributes							
Field type	Pushbutton						
Field name	%_AUTOTEXT002						
Field text	Display						
<input type="checkbox"/> With icon		Icon name	ICON_DISPLAY				
<input type="checkbox"/> Scroll.		Quick info	Display				
Line	3	Column	30	Length	19	Vis.len.	17
Groups						Height	1
FctCode	FTCH	FctType		LoopType		LoopDisp	0

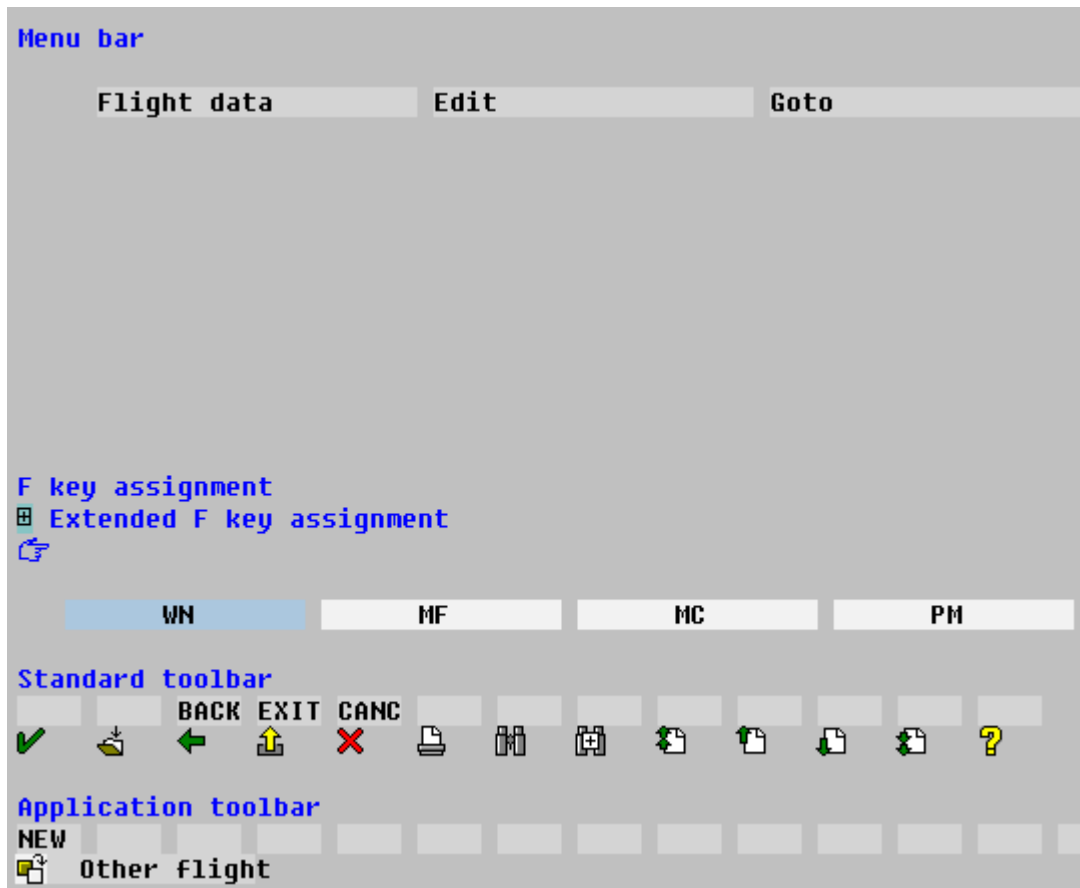
Menu Painter

In the SAP system, the user interface of a program consists of one or more GUI status and a GUI title that you define in the Menu Painter. A GUI status is a set of dynpro elements needed for the transaction at a given time. A GUI title is the title bar text that appears at the top of a Windows session.

Setting up Function codes

In order to be able to process the function code of a screen element that you define in the Menu Painter the program must specify the GUI status that should appear with the screen at PBO.

In the TZ20 transaction the *Other Flight* pushbutton is defined as a part of the menu bar in the TD0100 GUI status. The *Other Flight* pushbutton has the function code NEW in the Menu Painter.



You can set the GUI status in a dialog program with the key word SET PF-STATUS:

```
SET PF-STATUS <GUI_status>.
```

The <GUI_status> is a character string up to 8 characters and can be either a literal (in single quotes) or a variable.

You set the title of a screen or a dialog box with the key word SET TITLEBAR as follows:

```
SET TITLEBAR <title> WITH <p1> <p2> <p3> <p4>.
```

This statement can accept up to four parameter values. You enter the corresponding placeholders (using a &) in the title definition. For example, in a *Maintain Table* screen, the title might be defined as follows:

```
SET TITLEBAR 'ABC' WITH 'Customer'.
```

If the title 'ABC' is:

Maintain Table &

then the window title will be as follows:

Maintain Table Customer

The system fills & at runtime with the name of the specified table.

**Note**

If you do not set a user interface for a screen, the system displays that screen with the same GUI elements that were already set for the previous screen. If the transaction does not have a previous screen, or if you have not set any GUI status yet, the screen will be issued without a user interface.

For an example of how to program your PBO module in the flow logic of the corresponding screen, look at transaction TZ20. The single PBO module sets the GUI interface and title of the transaction. The TD0100 GUI status contains the definition of the *Other Flight* pushbutton.

```
MODULE STATUS_0100 OUTPUT.
  SET PF-STATUS 'TD0100'.
  SET TITLEBAR '100'.
ENDMODULE.
```

For more information about defining a GUI status and titles, see *BC ABAP/4 Workbench Tools*.

Handling Function codes

When the user selects a function in a transaction, the system copies the function code into a specially designated work field called OK_CODE. This field is global in the ABAP/4 module pool. The OK_CODE can then be evaluated in the corresponding PAI module.

The function code is always passed in exactly the same way, regardless of whether it comes from a screen's pushbutton, a menu option, function key or other GUI element.

In the Screen Painter, if you display the field list for a screen, the OK_CODE field is always the last field in the list. This field is initially nameless, and has field type *OK*. You can give this field any name, but it is traditionally called the OK_CODE. Whatever you name it, you must also include a definition for it as a global field in your module pool:

**Example**

```
PROGRAM SAPMTZ20.

...
DATA: OK_CODE(4).
```

The program declaration for an OK_CODE is *four* characters long.

For an example of how to handle the function code, look at transaction TZ20.

```
*-----*
*   PAI Modules MTZ20I01                               *
*&-----*
```

```

*&      Module  USER_COMMAND_0100  INPUT
*&-----*
MODULE USER_COMMAND_0100 INPUT.
  CASE OK_CODE.
    WHEN 'FTCH'.
      SELECT SINGLE * FROM SPFLI WHERE CARRID = SPFLI-CARRID
                        AND CONNID = SPFLI-CONNID.
      IF SY-SUBRC NE 0. CLEAR SPFLI. ENDIF.
      CLEAR OK_CODE.
    WHEN 'NEW'.
      CLEAR: SPFLI, OK_CODE.
    WHEN 'CANC'.
      CLEAR OK_CODE.
      SET SCREEN 0. LEAVE SCREEN.
    WHEN 'EXIT'.
      CLEAR OK_CODE.
      SET SCREEN 0. LEAVE SCREEN.
    WHEN 'BACK'.
      CLEAR OK_CODE.
      SET SCREEN 0. LEAVE SCREEN.
  ENDCASE.
ENDMODULE.

```

For example, if the user selects the *Display* pushbutton, the FTCH function code is copied into the OK_CODE internal work field. The OK_CODE is then checked in the PAI module and, if it contains FTCH, a SELECT is executed in order to fetch the data to be displayed.

The SET SCREEN and LEAVE SCREEN statements control screen flow. The specification SET SCREEN 0 tells the system to go back a whole call level. In this small program, this means exiting from the transaction altogether. Calling levels are described in *Controlling the Screen Flow*.

After processing the function code, delete the contents of the field OK_CODE to avoid any unwanted function selection.



Note

The function code currently active in a program can also be ascertained from the SY-UCOMM variable.

Handling Field Selections

Transactions often let users select a field to request a function. In some cases, field selection alone can trigger a function. At others, the user clicks on a field and then selects a menu option, function key or other button.

To get notification of field selection for your program:

- Field selection alone

When you provide the F2 key with a function code, the system passes the code to your program when the user selects a field. In this respect, double-clicking a field and single-clicking it with F2 are equivalent. The system responds to both selections by triggering PAI for the F2 function code.
- Field selection plus function request

Provide the screen elements with function codes, as usual. When the user selects or presses the element, the function code is passed in to the program.

When your program receives the relevant function code, it first needs to know what field has been selected. Use the GET CURSOR command:

```
GET CURSOR FIELD <field name>.
```

So you can find out what field the cursor is sitting in. The system sets the variable <field name> to the name of the screen field where the cursor is currently positioned.

Note that GET CURSOR returns an empty field-name parameter, if the user double-clicks on an area that is not a field.

When processing step-loops, use the LINE parameter to find out which loop block line contains the cursor.

```
GET CURSOR FIELD <field name> LINE <field2>.
```

The variable <field2> contains the number of the loop block line containing the cursor.



Example

Field selection plus function request:

```
GET CURSOR FIELD selffield.  
IF SELFIELD NOT SPACE.  
  CASE OK_CODE.  
    WHEN 'SELE'. PERFORM DISPLAY_FIELD_INFO USING SELFIELD.  
    WHEN 'CHNG'. PERFORM MODIFY_FIELD USING SELFIELD.  
    WHEN 'DELE'. PERFORM DELETE_FIELD USING SELFIELD.  
  ENDCASE.  
  CLEAR OK_CODE.  
ENDIF.
```

Sharing GUI-Statuses

In the ABAP/4 Development Workbench, screens and user interfaces are independent of one another. You can, for instance, use the same interface for several screens. Some screens, however, might have fewer active functions than other screens. Instead of creating a new GUI-status for these screens, you can use the same status and deactivate one or more of its functions.

The system offers you an additional keyword to deactivate specific menu functions in a GUI-status. The format of this statement is:

```
SET PF-STATUS <GUI status> EXCLUDING <function codes>.
```

If you want to deactivate a single function, the **<function codes>** parameter is a single type C field. Enter the name of the appropriate function in quotes. To deactivate several functions, place these function codes in an internal table. The internal table must have the following structure:

```
DATA: BEGIN OF INTTAB OCCURS 20,  
      FUNCTION (4),  
      END OF INTTAB.
```

Specifying functions with EXCLUDING deactivates all the relevant GUI-items (menu item, pushbutton and function code) assigned the given code. Deactivated pushbuttons do not appear at all. Deactivated menu functions appear (as do function key assignments in the list displayed with the right mouse button), but only in grayed-out form. If the user selects them, nothing happens.

Programming with Radio Buttons

Radio buttons are simple input fields. They do not have associated function codes, and thus do not themselves trigger a PAI event. However, there are some special things to know about using them to interpret user requests.

Setting Up Radio Buttons

When you add radio buttons to a screen, the Screen Painter automatically creates a one-character screen field for the button. You should declare a corresponding one-character variable for the radio buttons in your ABAP/4 module.

```
DATA: RADIO1, RADIO2, RADIO3.
```


By default, the system sets the first radio button in the group to on.

Checking the Radio Button Selection

Radio buttons are exclusive selection buttons that belong to a logical group. If the user clicks on one, all the other buttons in the group are automatically deselected by the system. If a radio button is set then it has the value 'X'. You do not have to program the deselection of radio buttons: the screen interface does it for you.

At the relevant PAI event, your check can assume that only one button is on at a time.

```
IF RADIO1 NE SPACE:
  PERFORM PROCESS_RADIO1.
ELSEIF RADIO2 NE SPACE:
  PERFORM PROCESS_RADIO2.
ELSE.
  PERFORM PROCESS_RADIO3.
ENDIF.
```

Setting Radio Button Values from the Program

When the user clicks on a radio button, the screen interface turns off all the other buttons in the group.

However, if you want to set radio buttons yourself (from the module pool), the screen interface is not available to deselect the rest. You must program this deselection yourself.

```
RADIO1 = SPACE.
RADIO2 = 'X'.
RADIO3 = SPACE.
```

Programming with Check Boxes

Check boxes are non-exclusive selection buttons. The user can turn on more than one at a time.

When you add check boxes to a screen, the Screen Painter automatically creates a one-character screen field for each box. You should declare a corresponding one-character variable for the boxes in your ABAP/4 module.

```
DATA: CHECK1, CHECK2.
```

By default, all check boxes are initialized to off. If you want to initialize them to on, assign them a value:

```
CHECK1 = 'X'.
CHECK2 = 'X'.
```

To check which boxes were selected by the user, you can query which boxes have a value not equal to space.