

Chapter 28: Introduction to Dialog Programming

Overview

This section gives you an introduction into dialog programming. The following topics are described:

Transactions

A Sample Transaction

Dynpro

ABAP/4 Module Pool

Interaction between Dynpro and ABAP/4 Module Pool

Contents

Transactions	28-2
Dynpro	28-5
ABAP/4 Module Pool.....	28-7
Interaction between Dynpro and ABAP/4 Module Pool	28-10

Transactions

A transaction is a program that conducts a dialog with the user. In a typical dialog, the system displays a screen on which the user can enter or request information. As a reaction on the the user input or request, the program executes the appropriate actions: it branches to the next screen, displays an output, or changes the database.



Example

A travel agent wants to book a flight. The agent enters the corresponding data on the screen. The system either confirms the desired request, that is, the agent can book the flight and the customer travels on the desired day on the reserved seat to the chosen destination, or the system displays the information that the flight is already booked up.

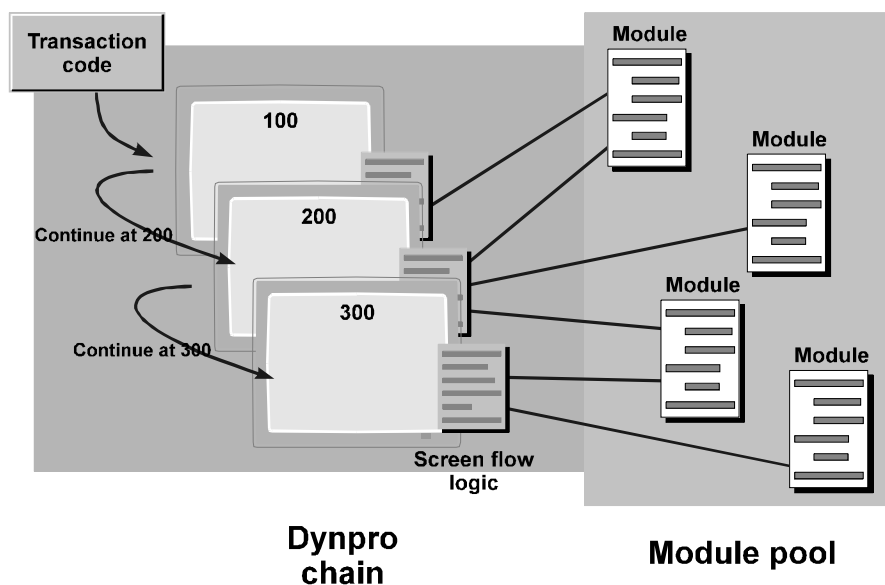
To fulfil such requirements, a dialog program must offer:

- a user-friendly user interface
- format and consistency checks for the data entered by the user
- easy correction of input errors
- access to data by storing it in the database.

ABAP/4 offers a variety of tools and language elements to meet the requirements stated above in the dialog programs.

Structure of a Dialog Program

A dialog program consists of the following basic components:



- Screens (dynpros)

Each dialog in an SAP system is controlled by dynpros. A dynpro (DYNAMIC PROGRAM) consists of a screen and its flow logic and controls exactly one dialog step. The flow logic determines which processing takes place before displaying the screen (PBO-Process Before Output) and after receiving the entries the user made on the screen (PAI-Process After Input).

The screen layout fixed in the Screen Painter determines the positions of input/output fields, text fields, and graphical elements such as radio buttons and checkboxes. In addition, the Menu Painter allows to store menus, icons, pushbuttons, and function keys in one or more GUI statuses. Dynpros and GUI statuses refer to the ABAP/4 program that control the sequence of the dynpros and GUI statuses at runtime.

- ABAP/4 module pool

Each dynpro refers to exactly one ABAP/4 dialog program. Such a dialog program is also called a module pool, since it consists of interactive modules. The flow logic of a dynpro contains calls of modules from the corresponding module pool. Interactive modules called at the PBO event are used to prepare the screen template in accordance to the context, for example by setting field contents or by suppressing fields from the display that are not needed. Interactive modules called at the PAI event are used to check the user input and to trigger appropriate dialog steps, such as the update task.

All dynpros to be called from within one transaction refer to a common module pool. The dynpros of a module pool are numbered. By default, the system stores for each dynpro the dynpro to be displayed next. This dynpro sequence or chain can be linear as well as cyclic. From within a dynpro chain, you can even call another dynpro chain and, after processing it, return to the original chain.

Transferring Field Data

How do I display fields known in an ABAP/4 module on the screen? How do I transfer user entries on the screen to the module? In contrast to report programming, you **cannot** write field data to the screen using the WRITE statement. The system instead transfers data by comparing screen field names with ABAP/4 variable names. If both names are the same, it transfers screen field values to ABAP/4 program fields and vice versa. This happens immediately before and immediately after displaying the screen.

Field Attributes

For all screen fields of a dynpro, field attributes are defined in the Screen Painter. If a field name in the screen corresponds to the name of an ABAP/4 Dictionary field, the system automatically establishes a reference between these two fields. Thus, a large number of field attributes in the dynpro is automatically copied from the ABAP/4 Dictionary. The field attributes together with data element and domain of the assigned Dictionary field form the basis for the standard functions the dynpro executes in a dialog (automatic format check for screen fields, automatic value range check, online help, and so on).

Error Dialogs

Another task of the dynpro processor is to conduct error dialogs. Checking the input data is carried out either automatically using check tables of the ABAP/4 Dictionary or by the ABAP/4 program itself. The dynpro processor includes the error message into the received screen and returns the screen to the user. The message may be context-sensitive, that is, the system replaces placeholders in the message text with current field contents. In addition, only fields whose contents is related to the error and for which a correction may solve the error can accept input. For more information on error handling, see *Handling Errors and Messages*.

Data Consistency

To keep data consistent within complex applications, ABAP/4 offers techniques for optimizing database updates that operate independent of the underlying database and correspond to the special requests of dialog programming. For more information on database updates, see *Programming Database Updates*.

To illustrate the concept and usage of transactions, a sample transaction follows.

A Sample Transaction

Transaction TZ10 (development class SDWA) is delivered with the system. This transaction consists of one dynpro only. The user can enter the ID of an airline company and a flight number to request flight information:

Display Flight Data	
Flight data	Edit Goto System Help
✓	[Search] [Download] [Upload] [Back] [Forward] [Cancel] [Print] [List]
[Find]	
Airline carrier	AA
Flight number	17
Flight data	
From city	[]
Dep. airport	[]
Destination	[]
Dest. airport	[]
Flight time	00:00:00
Departure	00:00:00
Arrival	00:00:00
Distance	0
Dist. in	[]

If the user chooses *Display*, the system retrieves the requested data from the database and displays it:

Display Flight Data	
Flight data Edit Goto System Help	
<input type="checkbox"/> <input type="text"/> <input type="button" value="↓"/> <input type="button" value="↶"/> <input type="button" value="↷"/> <input type="button" value="✖"/> <input type="button" value="🖨"/> <input type="button" value="📁"/> <input type="button" value="🔍"/>	
<input type="button" value="🔍"/>	
Airline carrier	AA
Flight number	17
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

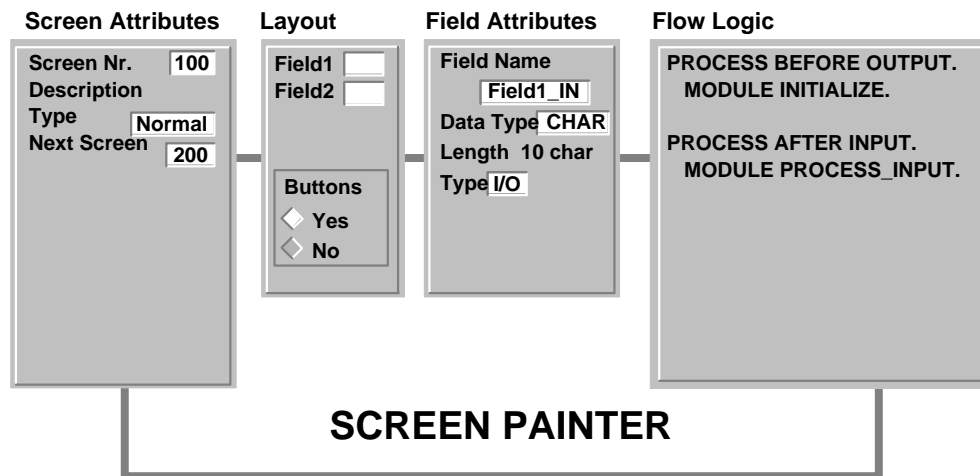
The structure of transaction TZ10 is described in the following topics:

Dynpro

Each screen contains fields used to display or request information. Fields can be text strings, input or output fields, radio buttons, checkboxes, or pushbuttons. The screen of Transaction TZ10 contains only texts and input/output fields.

An SAP dynpro consists of several components:

- **Flow logic:** Calls of the ABAP/4 modules for a screen.
- **Screen layout:** Positions of the texts, fields, pushbuttons, and so on for a screen.
- **Screen attributes:** Number of the screen, number of the subsequent screen, and others.
- **Field attributes:** Definition of the attributes of the individual fields on a screen.



You create and edit all components of a dynpro in the Screen Painter. To call the Screen Painter, create a dynpro in the Object Browser or double-click on an existing dynpro. The Object Browser then calls the Screen Painter. There, you can enter the flow logic of the new dynpro. By pressing the corresponding pushbutton you can maintain the *Screen attributes*, branch to the Full Screen-Editor or you choose the pushbutton *Field list* and change the attributes of fields. For more information on the Screen Painter, see the documentation *BC ABAP/4 Workbench Tools*.

Screen Attributes

From the user's point of view, a transaction is a sequence of screens, displayed one after another. How do I determine this sequence? The transactions's attributes determine the first screen to be displayed. The attributes of the individual dynpros determine which screen to display after the current screen. You can also set the number of the subsequent screen dynamically from within the ABAP/4 program.

For our example, the screen attributes need not be changed, since no subsequent screen is called.

Layout

Choose *Fullscreen* to go to the screen editor. Here you can determine the layout of the screen. For Transaction TZ10, the desired fields can be copied from Table SPFLI of the ABAP/4 Dictionary. For more information on the fullscreen editor, see *BC ABAP/4 Workbench Tools*.

Field Attributes

To display and modify the attributes of the individual fields (input/output fields, input required, possible entries button, invisible, and so on), use the *Field list*.

The fields *Company* (SPFLI-CARRID) and *Flight number* (SPFLI-CONNID) are defined as input/output fields. All other fields are used only for outputting the flight data.

Flow Logic

The flow control code of a dynpro consists of a few statements that syntactically resemble ABAP/4 statements. However, you cannot use flow control keywords in ABAP/4 and vice versa. You enter the flow control code in the Screen Painter as one component of the dynpro.

The flow control for the dynpro of Transaction TZ10 looks like this:

```
PROCESS BEFORE OUTPUT.
  MODULE SET_STATUS_0100.
*
PROCESS AFTER INPUT
  MODULE USER_COMMAND_0100.
```

The PROCESS statement names the event type for the dynpro and the MODULE statement tells the system which ABAP/4 routine to call for this event. In this example, there is only one MODULE for each event PBO and PAI. However, an event can contain several statements with several keywords. (The flow control language contains only few statement types. The most important are MODULE, FIELD, CHAIN, LOOP, CALL SUBSCREEN.)

To display information on the statement syntax in the flow logic, choose *Utilities → Help on...* in the flow logic editor. In the subsequent dialog window, mark *Flow logic keyword*, enter the name of the desired keyword, and press **ENTER**.

ABAP/4 Module Pool

In the Object Browser, the module pool code belongs to one of the following categories:

- **Global fields:** data declarations that can be used by all modules in the module pool
- **PBO modules:** modules that are called before displaying the screen
- **PAI modules:** modules that are called in response to the user input
- **Subroutines:** subroutines that can be called from any position within the module pool

By default, the system divides a module pool into one or several include programs. An include program can contain several modules of the same type (only PBO modules or only PAI modules). The main program then consists of a sequence of INCLUDE statements that link the modules to the module pool:

```
*&-----*
*& Module pool      SAPMTZ10                                *
*&                                                         *
*&-----*
*&                                                         *
*&      Display data of Table SPFLI                        *
*&                                                         *
*&-----*

* Global data
INCLUDE MTZ10TOP.

* PAI modules
INCLUDE MTZ10I01.
```

```
* PBO modules
INCLUDE MTZ10O01.
```

In the ABAP/4 editor, you can display the code hidden behind the INCLUDE statements by choosing *Edit* → *More functions* → *EXPAND include*. With all INCLUDE statements expanded, the module pool looks like this:

```
*&-----*
*& Module pool      SAPMTZ10                                *
*&                FUNCTION: Display data from Table SPFLI    *
*&                                                         *
*&-----*
*-----*
*  INCLUDE MTZ10TOP (This is the TOP include:                *
*                   the TOP module contains global data declarations) *
*-----*
PROGRAM  SAPMTZ10.
  TABLES: SPFLI.

  DATA OK_CODE(4).

*-----*
*  INCLUDE MTZ10I01  (This is a PAI include.)                *
*-----*
*&-----*
*&      Module      USER_COMMAND_0100  INPUT                *
*&-----*
*      Retrieve data from SPFLI or leave transaction          *
*-----*
MODULE USER_COMMAND_0100 INPUT.
  CASE OK_CODE.
    WHEN 'SHOW'.
      CLEAR OK_CODE.
      SELECT SINGLE * FROM SPFLI WHERE CARRID = SPFLI-CARRID
                                AND  CONNID = SPFLI-CONNID.
    WHEN SPACE.
    WHEN OTHERS.
      CLEAR OK_CODE.
      SET SCREEN 0. LEAVE SCREEN.
  ENDCASE.
ENDMODULE.

*-----*
*  INCLUDE MTZ10O01  (This is a PBO include.)                *
*-----*
*&-----*
*&      Module STATUS_0100                                *
*&-----*
*      Specify GUI status and title for screen 100          *
*-----*
MODULE STATUS_0100.
  SET PF-STATUS 'TZ0100'.
  SET TITLEBAR '100'.
ENDMODULE.
```

You use the ABAP/4 Dictionary to store frequently used data declarations centrally. Objects defined in the Dictionary are known throughout the system. Active Dictionary definitions can be accessed by any application. Data defined in the Dictionary can be included in a screen or used by an ABAP/4 program. You declare global data in the TOP module of the transaction, using the TABLES, STRUCTURE, LIKE statements and

others. Transaction TZ10 accesses the Dictionary definition of Table SPFLI to provide the desired flight data display. If the TOP include contains the TABLES: SPFLI declaration, all modules in the module pool can access the table fields of Table SPFLI.

The PAI module USER_COMMAND_0100 checks which pushbutton the user activated (**CASE OK_CODE**). The *Display* pushbutton in Transaction TZ10 has the function code 'SHOW'. (For more information on handling function codes, see *Processing User Requests*). The program then tries to select those records in the SPFLI database that correspond to the data the user entered. The WHERE condition determines matching records by comparing the fields **SPFLI-CARRID** and **SPFLI-CONNID** with the database key fields **CARRID** and **CONNID**. As soon as a matching record is found, the database transfers all accompanying SPFLI fields to the program table.

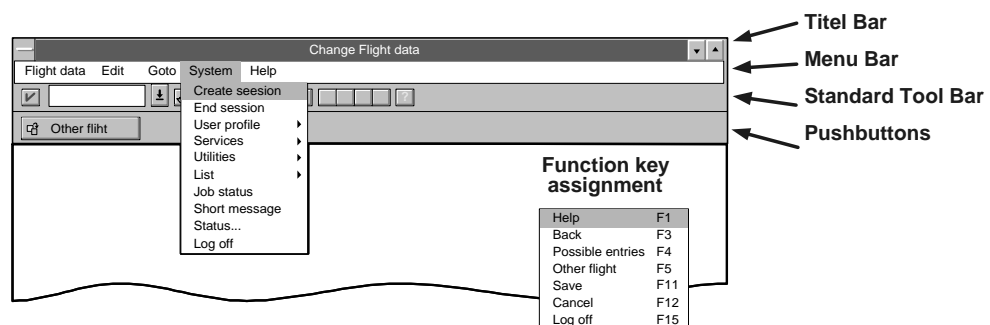
When the screen is displayed again, the complete information appears in the output fields of the screen. The system automatically displays these fields, since the ABAP/4 field names **SPFLI-CARRID** and **SPFLI-CONNID** are the same as the screen field names.

In the PBO module STATUS_0100 of Transaction TZ10, the screen 100 receives a GUI status (using SET PF-STATUS) and a GUI title (using SET TITLEBAR):

```
SET PF-STATUS 'TZ0100'.
SET TITLEBAR '100'.
```

A *GUI status* is a subset of the interface elements used for a certain screen. The status comprises those elements that are currently needed by the transaction. The GUI status for a transaction may be composed of the following elements:

GUI-Status Elements



The GUI title is the screen title displayed in the title bar of the window. In contrast to the GUI status that can be used for several screens, a GUI title belongs to one screen.

To create and edit GUI status and GUI title, you use the Menu Painter. To start the Menu Painter, create a GUI status or GUI title in an object list in the Object Browser (or double-click on an existing status or title).

For more information on the Menu Painter, see the documentation *BC ABAP/4 Workbench Tools*.

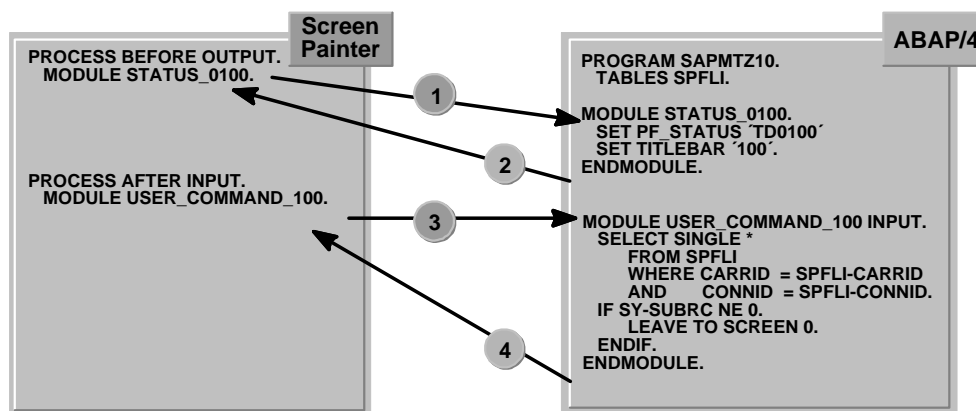
Interaction between Dynpro and ABAP/4 Module Pool

In its most simple form, a transaction is a collection of screens and ABAP/4 routines, controlled and executed by a dialog processor. The dialog processor processes screen after screen, thereby triggering the appropriate ABAP/4 processing for each screen.

For each screen, the system executes the flow logic that contains the corresponding ABAP/4 processing. The control passes from screen flow logic to ABAP/4 code and back.

The sequence of events for Transaction TZ10, for example, looks like this:

Interaction between Screens and ABAP/4 Modules



- 1 3 Control switches from screen processing to ABAP/4 processing
- 2 4 Control switches from ABAP/4 processing to screen processing

- In the PBO event, the statement **MODULE STATUS_0100** passes control to the corresponding ABAP/4 module.
In the ABAP/4 module pool, the screen to be displayed receives a menu interface.
- After processing the module **STATUS_0100**, control returns to the flow logic.
For the PBO event, no further processing is required. The system display the screen and receives entries from the user. The entries are:
 - the values for the fields *Company* and *Flight number*.
 - the four-character function code that tells which pushbutton the user activated.
- The user input triggers the PAI event. The first PAI statement passes control to the ABAP/4 module **USER_COMMAND_0100**.
Module **USER_COMMAND_0100** processes the requests of the user. In our example, only one request is possible: displaying the flight data for the specified

flight. The ABAP/4 statement SELECT retrieves the data from the database and displays it.

4. After processing **MODULE USER_COMMAND_0100**, control returns to PAI. This terminates the dialog.