ABAP Course

Chapter 5: Dynpros

Content
This chapter will show you the structure of more complex programs with integrated Dynpros. You will create a new program which selects and shows flight details from the SAP flight example. Therefore you will use the Dynpro technology.

Prerequisites
You should be familiar with the SAP flight example and SQL statements.

Motivation
As the major part of the SAP transactions are still developed using ABAP and Dynpros this chapter is one of the most important ones to understand the program logic of SAP.

Lecture notes
Students should be familiar with SQL statements and the SAP flight example from the previous chapter.
Students can go on with their account from chapter 1.

- **Product**: All
- **Level**: Beginner
- **Focus**: Programming
- **Version**: 1.0
- **Author**: UCC Technische Universität München

Task 1: Login into the SAP system

Short description: Use SAPGui to login into the SAP system with your username and password

Login

Start the SAPGui and login into the development system using the provided account and password. Please refer to chapter 1 for your username and your password.

Task 2: Create your first program with Dynpros

Short description: Create a program which uses two Dynpros to display flight details from the SAP flight example

Please start the Object Navigator from the SAP Easy Access Menu by using the following path:

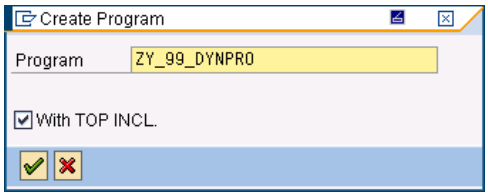
Menu path

Tools • ABAP Workbench • Overview • Object Navigator

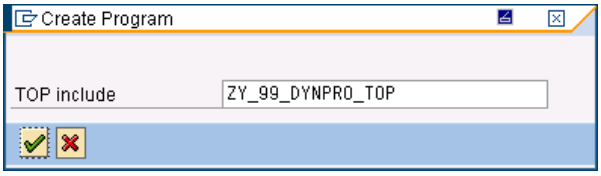
You may also use the transaction code **SE80** for direct access.

TOP INCL

Create a new program called '**ZY_##_DYNPRO**'. Please use '**TOP INCLUDE**'.



Modify the name of your top include to '**ZY_##_DYNPRO_TOP**'.



Hint

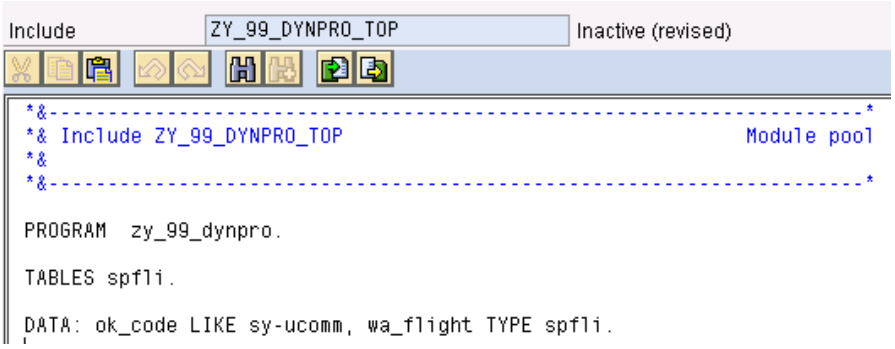
Hint:
Please note that the SAP system automatically proposed a name for your top include which did not start with the obligatory Z to mark it as a customer top include. Later on in the development you will often face such situations when the SAP system proposes a name which is not in the customer namespace. Pay attention to this!

Assign attributes

In the next pop-up you see all the programs attributes. You have to adjust this attributes. Select the status '**T Test Program**' and the application '*** Cross-Application**'. Continue and assign the new program to your existing package '**ZY_##**' and your transport request. The SAP system will ask you twice for a transport request: One for the new program and the other for the top include.

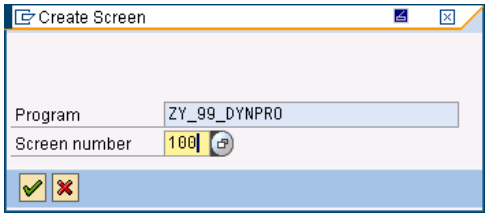
Maintain variables

If the system does not jump directly to your new top include, please open it manually by double clicking on it. In this step you have to define all global variables. Such variables are available in the entire program and every screen has access to them. These variables are the table **SPFLI**, the system variable **ok_code** and the work area **wa_flight**. Please define them in the top include:



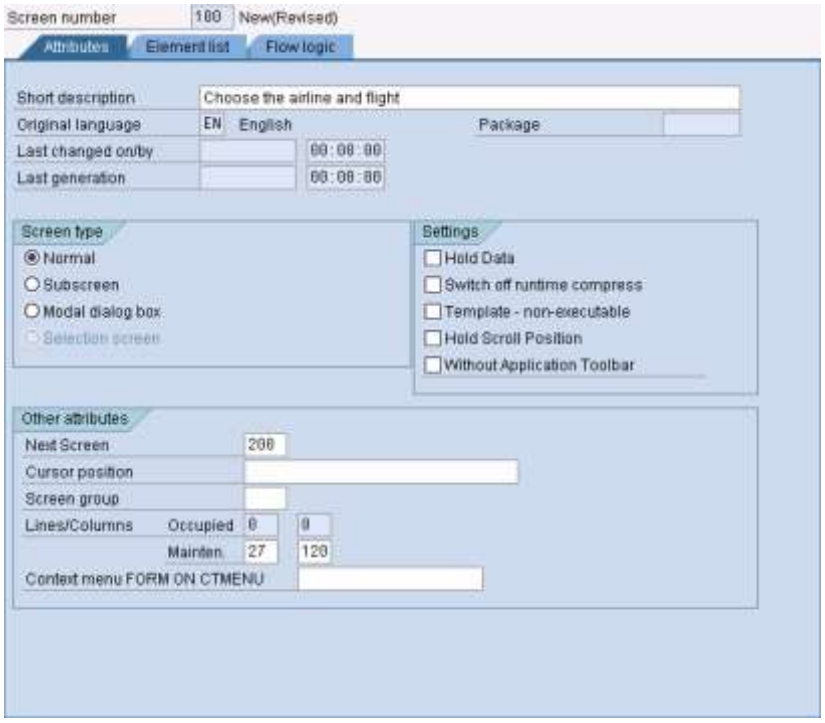
Create dynpro

The next step concerns the creation of the first Dynpro. Right click on your program name in the navigation tree and create a new Dynpro (screen) with the screen number 100.



Maintain attributes

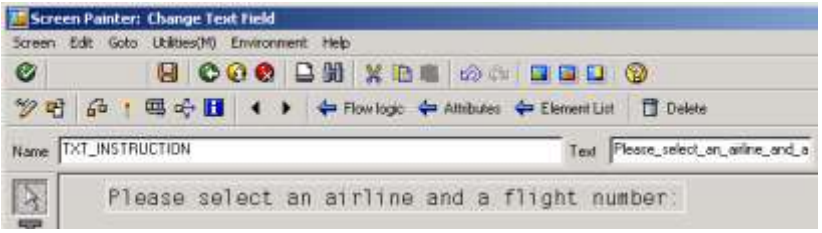
Now you have to specify the attributes of your new Dynpro. Maintain the short text and fill in Dynpro 200 as the 'N**ext Screen**'. This will let the Dynpro jump to Dynpro 200 after the PAI module of Dynpro 100 was processed. Of course, you have not created the Dynpro 200 yet, but you will do so later on in the exercise.



Call Graphical Screen Painter

Your next step is the creation of the layout of your first Dynpro. Therefore please call the Graphical Screen Painter by pressing the button Layout in the tool bar. As long as the Graphical Screen Painter is active you cannot access the SAPGui. You can return to the SAPGui by pressing the '**Back**' button in the Graphical Screen Painter.

Please select the text field button and draw a text field on your new screen. Use '**TXT_INSTRUCTION'** as the name of the text field and type in the text: '**Please select an airline and a flight number:**'. Your screen should now look similar to this:

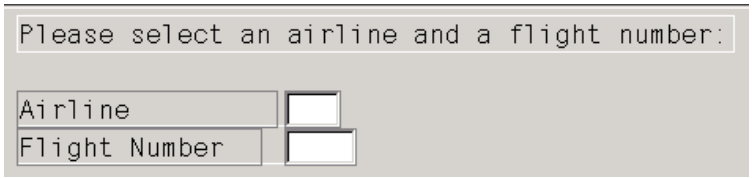


Call dictionary

The user of your new program should choose an airline and a flight number in this screen. This information is stored in the table '**SPFLI**' from the SAP flight example. To create two input fields now with an entry help you reference to the table '**SPFLI**' from the data dictionary. This can be done very easily by pressing the button '**Dictionary/Program fields window**' in the toolbar. Type in the name SPFLI and press the 'Get from Dictionary' button. You will notice that the SAP system reads the information from the dictionary.

Select CARRID und CONNID

As your application should provide the user with two input fields for the airline and the flight number, please select the '**CARRID**' and '**CONNID**' row from the pop-up and press enter. This changes your mouse cursor and lets you drop two new input fields on your screen. So far your screen should look similar to this:



In the last step we need a button to go on to our second Dynpro. So please add a button to your screen and name it '**BTN_SELECT**' and choose a button text on your own. After you entered the information you will notice that the button is still colored red. This means that the SAP system is missing information.

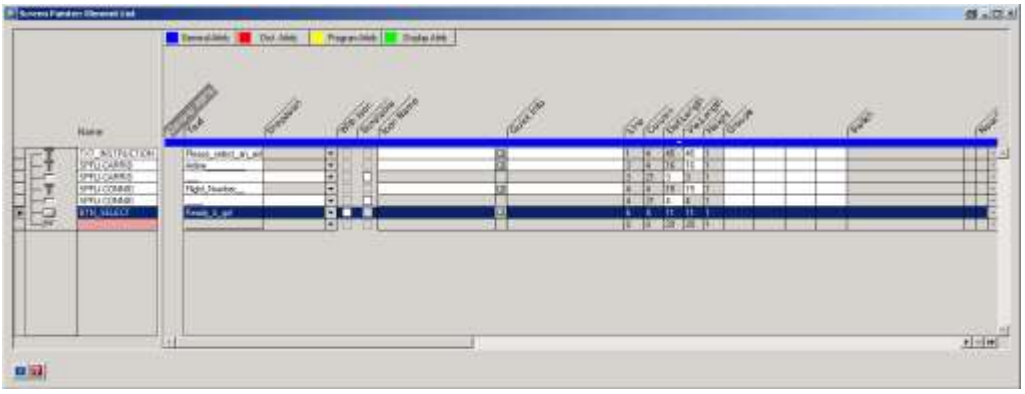
Hint

Hint:
If an element in the Graphical Screen Painter is colored red, this indicates that the Graphical Screen Painter is missing information. By double clicking on the red element you can get further information.

As your new button is still colored red you have to add some needed information. The needed information is the function code. Please double click on your button. This opens the attributes view and you discover the red input field where you enter now the function code '**SELECT**' and press enter. Later in your program pressing this button should automatically assign the value '**SELECT**' to your variable '**ok_code**'. The variable is

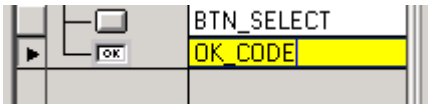
available everywhere in your program as this variable was declared in the top include. But until yet the link between the function code and the variable '**ok_code**' is still missing. To establish the link click on the '**Element List**' button in the toolbar. This will open a new window with all the elements from your current screen.

Element list



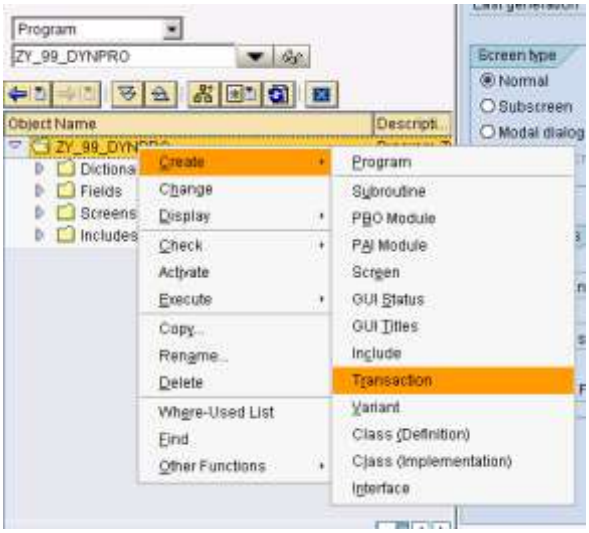
Once again you see a red field indicating that there is still something missing. Next to the red field there is a little OK-symbol which represents the variable '**ok_code**'. Please type in '**OK_CODE**' which will enable the link to the global variable '**ok_code**'.

OK_CODE

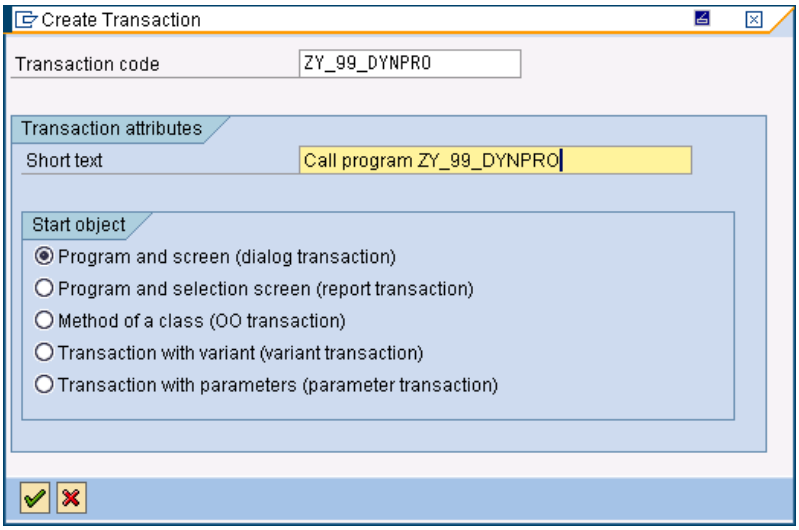


Please save your new Dynpro and return to the Object Navigator. To simplify the life of your user you create a transaction code for your new program. This is done by right clicking on your program in the navigation tree and choosing the menu point '**Create • Transaction**':

Create transaction

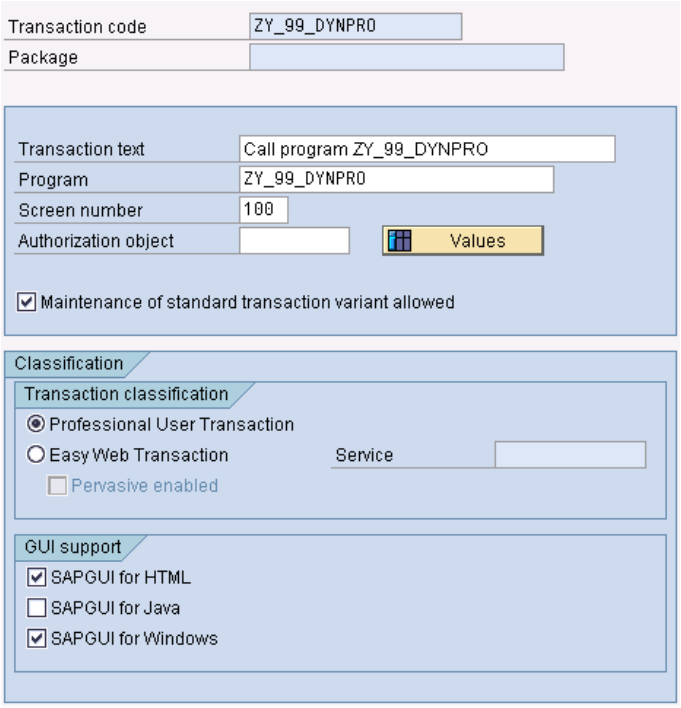


The SAP system comes up with a new pop-up where you can specify the transaction code as well as a description.



Please note that the transaction has nothing to do with the called program. The link between the transaction code and the program is done later when specifying the transaction attributes. This means: you may call a program with a completely different name than the transaction code. The little example of the Object Navigator with transaction code SE80 is a very good example of this independency.

Specify attributes



In the attributes of the transaction code you can now specify the called program. Please type in the name of your Dynpro program '**ZY_##_DYNPRO**' and enter the screen number **100**. Please check and save your transaction and activate all of your developments if they are not activated already.

Test and get an ABAP dump

To test your new transaction code and your new program, create a new session and call the program using your transaction code. You will see your text field and two input fields. The first one is equipped with an entry help and the second one is not. When hitting the button you will get an ABAP dump. This absolutely makes sense as you defined the screen 200 as the next screen of screen 100 and as you have not created the screen 200 yet your program runs into a problem.

Go back to your Dynpro program and switch to the '**Flow Logic**' tab. Here you see the flow logic of your first Dynpro, separated into a process before output module and a

process after input module. Please insert the following source code **after** the process after input line:

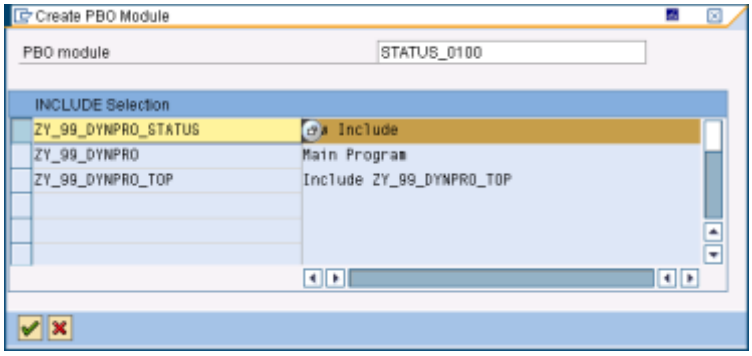
Insert

```
PROCESS BEFORE OUTPUT.
* MODULE STATUS_0100.
*
PROCESS AFTER INPUT.
  FIELD spfli-connid SELECT * FROM spfli WHERE carrid = spfli-carrid AND
  connid = spfli-connid.

* MODULE USER_COMMAND_0100.
```

This code fragment ensures that the Dynpro fills in the input fields every time the Dynpro resolves a user input. Why that is important can be shown very easily in the Dynpro program. Please start your Dynpro program again and test the behavior of your program. You will notice that both input fields are equipped with an entry help now. The second entry help is based on the first entry help but it is only useful when the user selected an airline before, because the second input field should only show the flights which are performed by the selected airline.

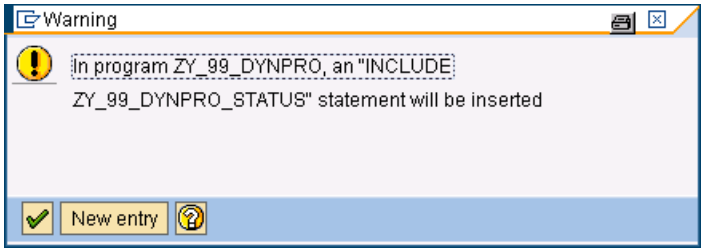
Uncomment & double click

Before you complete the flow logic, you will focus on the status bar and the title of your new program now. Therefore please uncomment the `MODULE STATUS_0100` line in the flow logic by removing the little * and double click on '**STATUS_0100**'. You will be asked if you want to create the new module. Please select yes and go on with the creation.



Rename include

The system automatically proposes a name for your new status module. The name of your new PBO module is fine, but the name for the PBO module-include is not. Please change the name and add a '**ZY_**' at the beginning of the include name. After you hit '**Enter**', the system comes up with a short message telling you that an additional include-statement is inserted into your program:



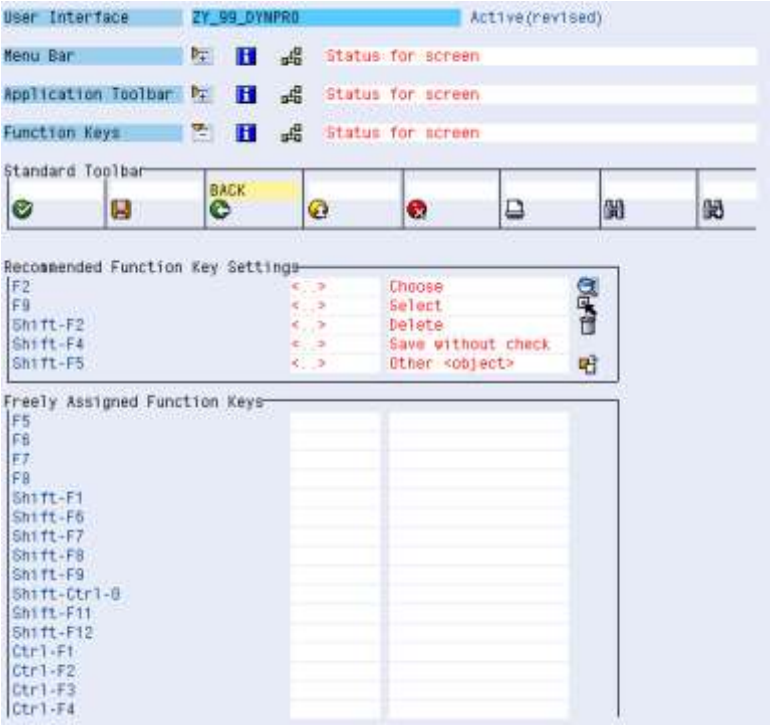
This is completely fine as you create a new include for the status bar and this include must be included in your Dynpro program. Double click on the new module STATUS_0100 to navigate forward to the source code of the module.

Uncomment

```
*---------------------------------------------------------------------*
***INCLUDE ZY_99_DYNPRO_STATUS .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Module  STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
module STATUS_0100 output.
  SET PF-STATUS '100'.
  SET TITLEBAR '100'.

endmodule.                    " STATUS_0100  OUTPUT
```

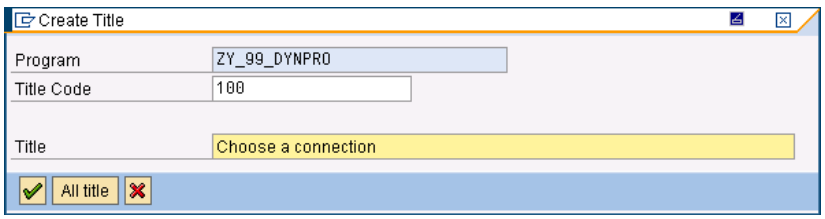
PF-STATUS

Uncomment both SET lines and replace the placeholder ‘xxx’ with the number ‘**100**’. Now you set up the status bar and then the title bar for your screen. This is done by double clicking on the ‘**100**’ after ‘**PF-STATUS**’. You are asked if you want to create the status. Please answer yes and maintain the short text in the next pop-up. After that pop-up the SAP system comes up with the following screen:

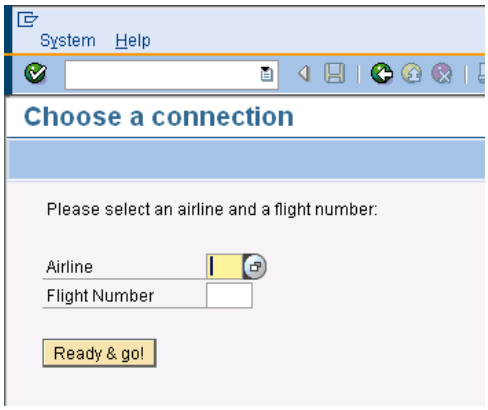


TITLEBAR

Here you can create, edit and delete the status bar for you screen. You may change the code which is parsed to the program for every button in the status bar. Those buttons, where no code is assigned are unavailable later on. Please maintain the code for the ‘**Back**’ button by adding ‘**BACK**’. Save your new status bar and return to the program code of your module. Double click on the ‘**100**’ after ‘**TITLEBAR**’.

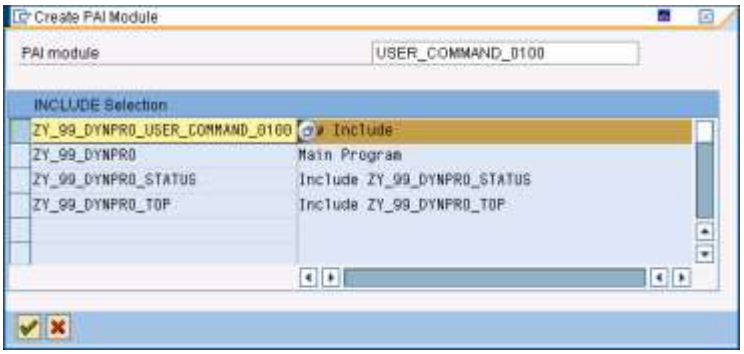


Maintain the title of your new title bar. Save, and activate **all** of your new objects, then test your new program. It should look similar to this:



Now as you have created the layout you should create the flow logic as well. So please go to your screen '**100**' and switch to the tab '**Flow Logic**'. Uncomment the line '**USER_COMMAND_0100**' and double click on it. You are asked by the system if you want to create the object. Please answer yes and maintain the name of the include: delete the last three digits of the name and add '**ZY_**' at the beginning.

USER_COMMAND_0100



Assign the new module to your existing package. In the next step you have to add the flow logic to your new module. The flow logic has to handle the '**ok_code**' in your program. Depending on the value of the '**ok_code**' the program either should leave the execution ('**BACK**') or read some data from table SPFLI ('**SELECT**'). Please insert a case instruction:

CASE instruction

```
*---------------------------------------------------------------------*
***INCLUDE ZY_99_DYNPRO_USER_COMMAND_0100 .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
MODULE user_command_0100 INPUT.

  CASE ok_code.
    WHEN 'BACK'.
      LEAVE PROGRAM.
    WHEN 'SELECT'.
      SELECT SINGLE * FROM spfli INTO wa_flight WHERE carrid = spfli-carrid
      AND connid = spfli-connid.
  ENDCASE.

  CLEAR ok_code.


ENDMODULE.                    " USER_COMMAND_0100  INPUT
```

Please save, check and activate all of your objects.

Hint:
Please note that the CASE instruction is case sensitive. This means that it makes a difference if you use 'back' or 'BACK' in your case instruction! This is a very common mistake.

Hint

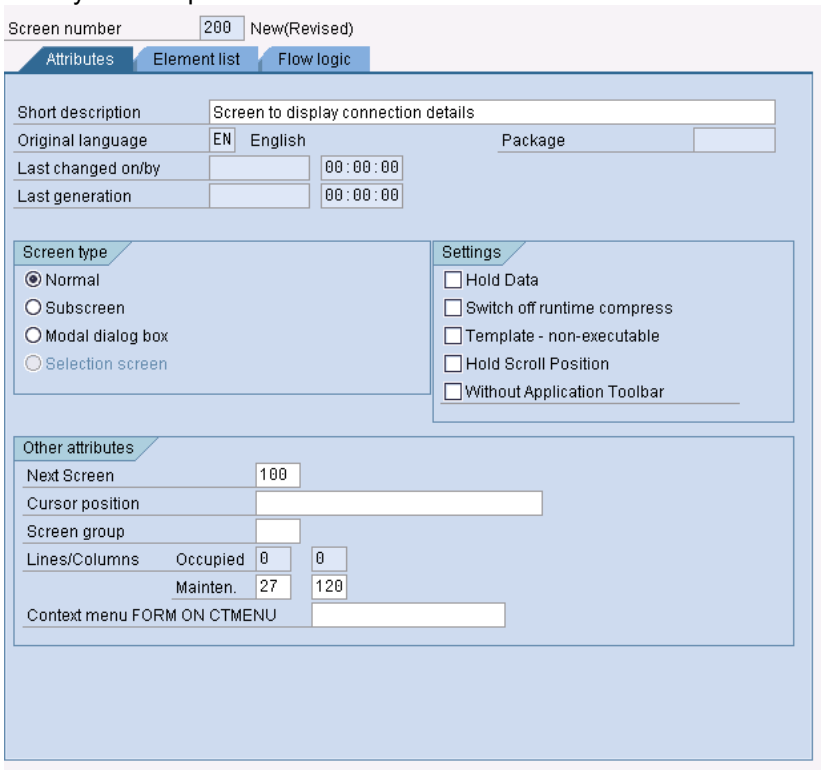
Task 3: Create the second screen

Short description: Create the second screen in your new program to display the flight details

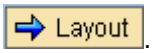
You have created the first screen in your program for the user to choose a connection, but you need a second screen to display the connection details. This is done in this task.

Screen 200

Please create a new screen in your program with the number '**200**'. Please refer to the second task if you have problems. Use screen '**100**' as the '**Next screen**'.



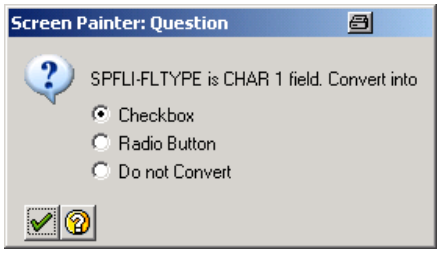
Edit Layout

Please switch to the layout of your new screen  . In the Graphical Screen Painter you are going to add some input fields to your screen as you want to display the connection details. But instead of creating the text fields step-by-step you refer to the data dictionary again  . Please enter the table name '**SPFLI**', hit '**ENTER**' and select every field from the table by using this little button:  . Then go on.

SPFLI-FLTYPE

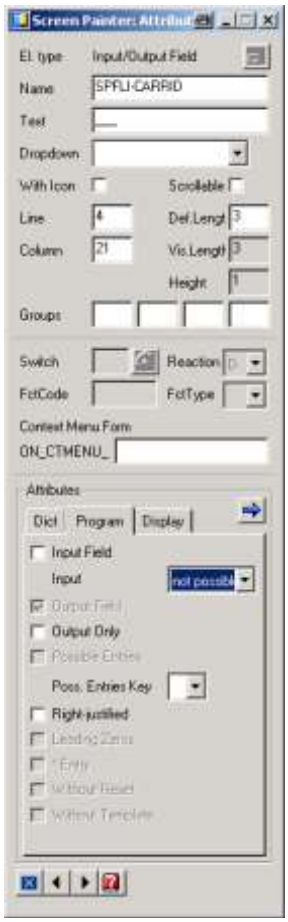
As you already know from the creation of your first screen your mouse cursor has changed now and you can just drop the new input fields onto your new screen. The SAP system comes up with a pop-up, asking you what to do with the '**SPFLI-FLTYPE**' table

field. The question arises as this table field is a '**Char 1**' field and indicates if the connection is either a charter flight or a regular flight. Please select checkbox, which means every charter flight will be indicated by an activate checkbox later on.

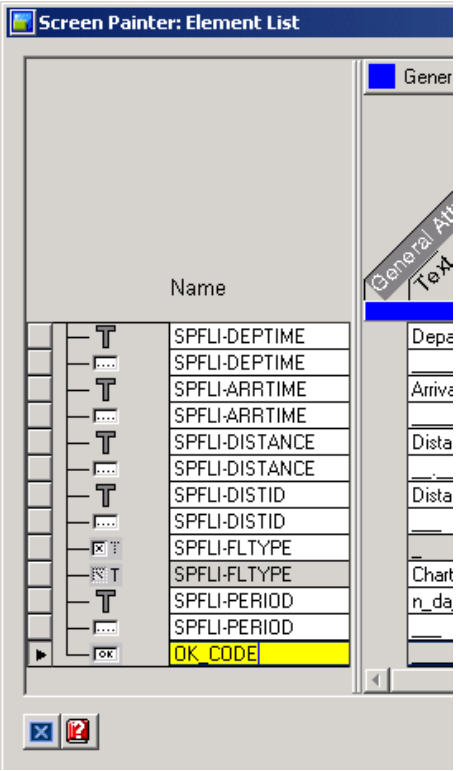


Input not possible

So far you see several input field in your new screen. Please note that the values in the client input field cannot be changed. Beside the client information we want to ensure nobody makes any changes to the airline and flight no. input fields too. This can be done very easily by changing the attributes of these input fields. Please double click on the airline input field (not the text field!) and change the attribute so that input is not possible:

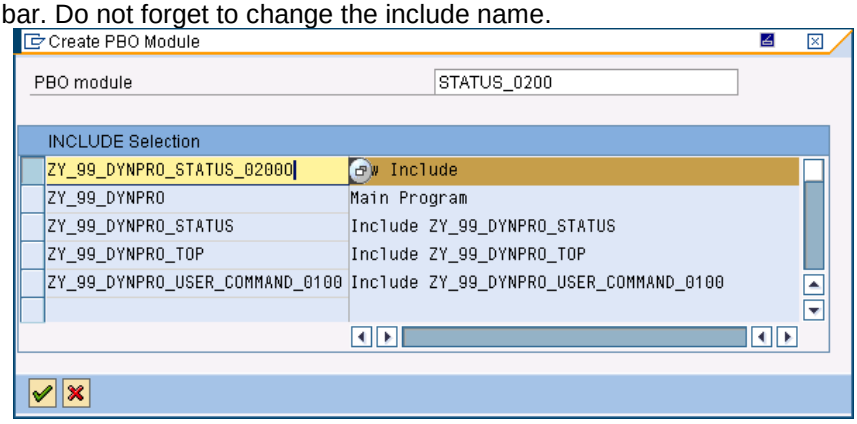


Do the same to the input field of the flight number. The last step is about handling the '**ok_code**'. Please assign the OK_CODE element to the variable '**ok_code**'.



PF-STATUS

Save and activate your new screen. Go back to the flow logic now and uncomment the line '**MODULE STATUS_0200**'. Double click on '**STATUS_0200**' to create the new status bar. Do not forget to change the include name.

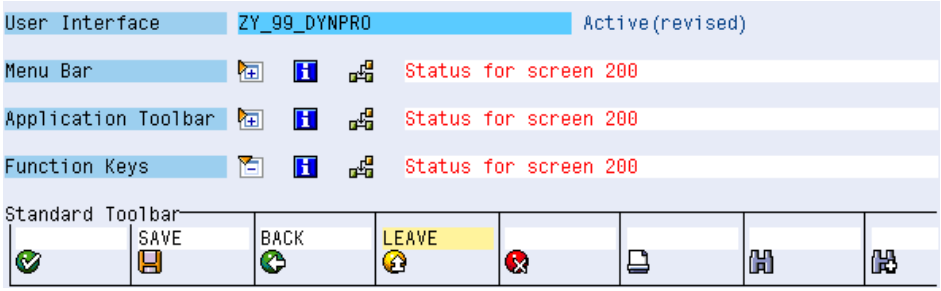


Title bar

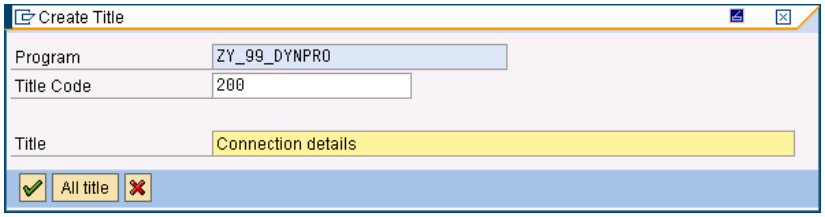
Uncomment the lines '**PF-STATUS**' and '**TITLEBAR**' in your module and replace the placeholder with '**200**'. Then double click on the first '**200**' to create the status bar. Maintain the short text.

```
*---------------------------------------------------------------------*
***INCLUDE ZY_99_DYNPRO_STATUS_0200 .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Module  STATUS_0200  OUTPUT
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
module STATUS_0200 output.
  SET PF-STATUS '200'.
  SET TITLEBAR '200'.

endmodule.                    " STATUS_0200  OUTPUT
```

Please add the following codes to the buttons:



You will create the case instruction for handling the parsed code later on. At first you have to create the title bar. So please save your status bar and go back to your module. Double click on the '**200**' after title bar and create a new one.



Before you may continue with the case instruction, add the following instruction to your module source code. This ensures that the input fields (named **spfli**) on your screen 200 contain the data from the work area **wa_flight**. The work area was filled in screen 100 and contains the connection details.

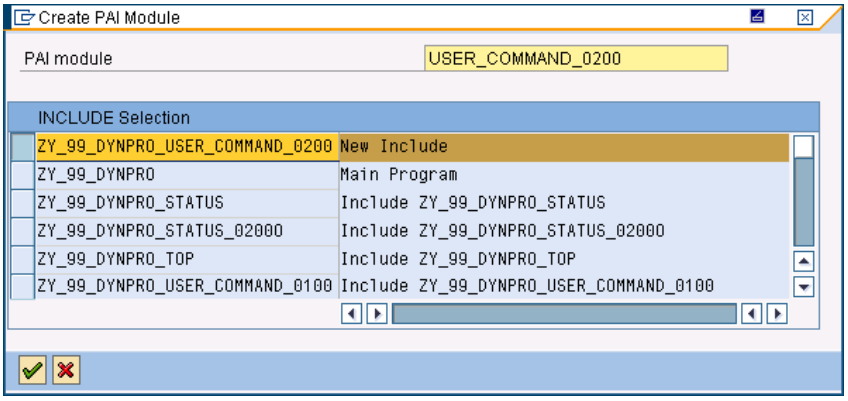
Spfli = wa_flight

```
*---------------------------------------------------------------------*
***INCLUDE ZY_99_DYNPRO_STATUS_02000 .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Module  STATUS_0200  OUTPUT
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
MODULE status_0200 OUTPUT.
  SET PF-STATUS '200'.
  SET TITLEBAR '200'.

  spfli = wa_flight.

ENDMODULE.                    " STATUS_0200  OUTPUT
```

Now save, check and activate your new screen and go back to the flow logic tab of your screen 200.

As you already did for the first screen you have to implement the flow logic for the second screen too. In the '**Flow Logic**' tab uncomment '**USER_COMMAND_0200**' and double click on it. Now implement a new case instruction, which handles the parsed code from your status bar.

USER_COMMAND_0200



Rename include

Do not forget to change the name of your include by adding a '**ZY_**' as prefix. After you have saved the include file add the following source code:

```
*---------------------------------------------------------------------*
***INCLUDE ZY_99_DYNPRO_USER_COMMAND_0200 .
*---------------------------------------------------------------------*
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0200  INPUT
*&---------------------------------------------------------------------*
*       text
*---------------------------------------------------------------------*
MODULE user_command_0200 INPUT.

  CASE ok_code.
    WHEN 'LEAVE'.
      LEAVE PROGRAM.
    WHEN 'BACK'.
    WHEN 'SAVE'.
      IF spfli <> wa_flight.
        MODIFY spfli FROM spfli.
      ENDIF.
  ENDCASE.

  CLEAR ok_code.

ENDMODULE.                 " USER_COMMAND_0200  INPUT
```

The source code does the following: in case of a '**LEAVE**', the program should be ended immediately. In case of a '**SAVE**' changed data from the structure '**SPFLI**' should be written to the database table '**SPFLI**'.
Save, check and activate your new program and test it.