

Chapter 32: Modifying the Screen

Overview

There is a number of different ways in which screens can be modified at runtime:

- Field attributes can be set in the dynpro
Temporary changes to field properties (e.g. input/output field, mandatory field) can be made from a dialog program. Fields can also be temporarily suppressed. Using this technique to modify screens dynamically often means you avoid the need to define additional screens.
- Field attributes can be changed with the help of the function field selection
The function field selection supports you in changing the attributes of screens dynamically.
- Displaying subscreens at runtime
A subscreen can be displayed in order to enhance an existing screen at runtime. A subscreen is used to selectively display certain fields. For example, you could have two subscreens, one containing *Material name* and *Material number* fields and the other containing *Customer name* and *Customer number* fields. One of these two subscreens is selected according to the input made by the user in the previous screen.
- Selecting a cursor position
The cursor can be positioned in a particular field on the screen from a dialog program, according to the user's entries.

You can find further information in the following sections:

Setting Screen Field Attributes

Changing Screen Field Attributes with the Function Field Selection

Using Subscreens

Manipulating the Cursor

Contents

Setting Screen Field Attributes	32-3
Changing Screen Field Attributes with the Function <i>Field Selection</i>.....	32-6
Field Selection - Overview	32-6
Calling Field Selection	32-7
Combination Rules For Attributes.....	32-10
Screen Painter Attributes	32-12
Generating the Field Selection	32-12
Function Modules for Field Selection.....	32-13
Linking Fields	32-15
The Display Attribute 'Active'	32-17
Authorization for Field Selection.....	32-17

Setting Screen Field Attributes

Using Subscreens..... 32-18

Manipulating the Cursor..... 32-19

Setting Screen Field Attributes

Every screen field has attributes that you set in the Screen Painter when you define the screen. At runtime, you may want to change these attributes, depending on what functions the user has requested in the previous screen. At runtime, attributes for each screen field are stored in a memory table called SCREEN. You do not need to declare this table in your program. The system maintains the table for you internally and updates it with every screen change.

The memory table SCREEN contains the following fields:

Name	Length	Description
NAME	30	Name of the screen field
GROUP1	3	Field belongs to field group 1
GROUP2	3	Field belongs to field group 2
GROUP3	3	Field belongs to field group 3
GROUP4	3	Field belongs to field group 4
ACTIVE	1	Field is visible and ready for input
REQUIRED	1	Field input is mandatory
INPUT	1	Field is ready for input
OUTPUT	1	Field is for display only
INTENSIFIED	1	Field is highlighted
INVISIBLE	1	Field is suppressed
LENGTH	1	Field output length is reduced
DISPLAY_3D	1	Field is displayed with 3D frames
VALUE_HELP	1	Field is displayed with value help

To activate a field attribute, set its value to 1. To deactivate it, set it to 0. When you set the ACTIVE attribute to 0, the system suppresses the field and turns off the ready for input attribute. The user can neither see the field nor enter values into it.



Note

You can define values for each of these attributes in the *Attribs. for 1 field* section in the field list of the Screen Painter. If you need more information about attribute meanings, see *BC ABAP/4 Workbench Tools*.

Setting Screen Field Attributes

As an example of modifying the screen dynamically, start with transaction tz50 (development class SDWA).

The transaction consists of two screens. In the first screen the user can enter flight identifiers and either request flight details (by pressing a *Display* pushbutton) or press the *Change* pushbutton to change the data of screen 200.

The field attributes are now set dynamically, according to whether the *Display* button or the *Change* button was selected. In both cases the same screen is now called, but with different field attributes.

If the same attributes need to be changed for several fields at the same time, these fields can be grouped together. For example, in order to change the fields in screen 200 dynamically, we assign these fields in the Screen Painter to the group MOD. You can specify up to four modification groups for each field. The contents of the *Groups* field are stored in the SCREEN table.

The changes to the attributes of the fields in this group can be implemented in a PBO module:

```
MODULE MODIFY_SCREEN OUTPUT.  
  CHECK MODE = CON_SHOW.  
  LOOP AT SCREEN.  
    CHECK SCREEN-GROUP1 = 'MOD'.  
    SCREEN-INPUT = '0'.  
    MODIFY SCREEN.  
  ENDLOOP.  
ENDMODULE.
```

The memory table SCREEN contains each field of the current screen together with its attributes.

The LOOP AT SCREEN statement puts this information in the header line of this system table.

In this example taken from transaction tz50, if the user chooses Display then SCREEN-INPUT is set to '0' and all fields belonging to the MOD group thus become display-only fields.

Because attributes have been changed, the MODIFY SCREEN statement is used to write the header line back to the table.

Changing Screen Field Attributes with the Function *Field Selection*

This topic describes how a special function *Field selection* (transaction SFAW and some function modules) support you in changing screen field attributes dynamically.

Field Selection - Overview

The function *Field selection* allows you to change the attributes of screen fields dynamically at runtime. However, you should only use this option if you often need to assign different field attributes to the same screen for technical reasons. In this case, the same rules apply for all fields, so any field modification is clear.

The following basic rules apply:

- All fields involved in the field selection process are grouped together in field selection tables and maintained using the *Field selection* function.
- Maintenance is always by module pool and screen group.
- On screens belonging to the screen group "blank" ('_'), there is no dynamic field selection.
- Since the screen field attribute SCREEN-GROUP1 is reserved for central field selection, you cannot use it for another purpose at the same time.
- If you are working with special predetermined rules where any change is tantamount to a program change, do not use this function. Instead, make any changes in the program itself.

With field selection, you can activate or deactivate the following attributes at runtime:

- Input
- Output
- Mandatory
- Active
- Highlighted
- Invisible

You can also determine the conditions and the type of changes involved. During the event PROCESS BEFORE OUTPUT, you call a function module which checks the conditions and, if necessary, changes the attributes.

Field selection distinguishes between influencing fields and modified fields. Modified fields must, of course, be screen fields. All fields should be defined in the Data Dictionary and you should declare the corresponding tables globally in the module pool with the TABLES statement. At runtime, a function module analyzes the field contents of the influencing fields and then sets the attributes of the modified fields accordingly.

Combining Screens in Screen Groups

Rather than maintaining field selection separately for each screen of a program, you can combine logically associated screens together in a screen group. To assign a screen to a screen group, enter the group in the field **Screen group** on the Screen Painter attributes screen.

Calling Field Selection

To call field selection, select *Tools → ABAP/4 Workbench → Development → Other tools → Field selection*. Maintenance is by program and screen group.

Module pool: ?

Screen group:

Selection:

- ☒ Influencing fields
- ☐ Modified fields
- ☐ Assign tables to screen group

Buttons: Display, Change

First, you must declare the table names of the fields involved. Choose *Assign tables to screen group* and enter the tables, for example:

Delete entry

Module pool: SAPMTXXX

Screen group: BILD

Tables for field selection

Table name	Text
SPFLI	Connection offers for flights

Changing Screen Field Attributes with the Function Field Selection

Save your entries and choose *Influencing fields* to enter the desired influencing fields into the list and optionally specify a NOT condition, a default value, and a field *Cust*, for example:

Infl. field	Op.	Non-condition	Default	Cust	Text
SPFLI-CARRID	NE	LH		<input type="checkbox"/>	Airline

The NOT condition is to be understood as a preselection. If one of the fields satisfies the NOT condition, it is not relevant for the following screen modification. Using the NOT condition may improve performance.

**Example**

Influencing field: SPFLI-CARRID

NOT condition: NE LH

SPFLI-CARRID is relevant for the field selection only if its contents are not equal to LH at runtime.

At runtime, the system uses the default value for the field modification if it cannot find the current value of the influencing field in the list of maintained values. You must define the default value yourself. This option allows you to maintain all the forms of an influencing field, which have the same influence, with a single entry.

By setting the field *Cust* accordingly, you can decide whether to allow customers to use the corresponding field for field selection or not. This field selection option is based on the predefined SAP field selection and allows customers to set screen attributes to suit their own requirements within a framework determined by application development. Many application areas have their own special Customizing transactions for this purpose (these are parameter transactions related to the Transaction SFAC; refer here to the documentation on the function module FIELD_SELECTION_CUSTOMIZE)

Then, choose *Modified fields* to enter all modifiable fields into a list, for example:

Changing Screen Field Attributes with the Function Field Selection

Modified field	Field	Cust	Text
SPFLI-AIRPFROM	2	<input type="checkbox"/>	Dep. airport

You can again set the field *Cust* accordingly if you want to allow customers to use these modifiable fields in special Customizing transactions. If *Cust* is selected, customers can also modify the field.

Each of these influencing and modifiable fields has an internal number which is unique for each program. When you generate by pressing *F16*, the number is automatically placed in SCREEN-GROUP1 of the appropriate screens and cannot be changed in Screen Painter. This enables the system to establish a one-to-one relationship between the field name and SCREEN-GROUP1.

Finally, you create links between influencing and modifiable fields from the two lists: Specify which contents of an influencing field influences the modifiable field in which way.

To link fields, select the fields from both lists with *Choose* or double-click. If you select an influencing field, the list of modifiable fields appears and vice versa. From this list, select the desired link. A list appears in which you can enter the relevant conditions, for example:

Contents	Input	Output	Active	Oblig.	Highl.	Invisible
LH	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

The entry above results in suppressing the display of field SPFLI-AIRPFROM on those screens, in whose PBO the corresponding field selection function modules are called and if SPFLI-CARRID then contains 'LH' (see *Function Modules for Field Selection*).

The function of the *Active* attribute is described in *The Display Attribute 'Active'*.

Combination Rules For Attributes

If several influencing fields influence the same modified field, there must be a combination rule to determine how these influences are linked. You use the tables to below establish how a single field attribute is set, if it is activated and/or deactivated by different influences. The screen processor controls the combination of several attributes.

Input			
		Field 1	
		' _ '	'X'
Field 2	' _ '	' _ '	' _ '
	'X'	' _ '	'X'

Changing Screen Field Attributes with the Function Field Selection

Output			
		Field 1	
		‘ ’ —	'X'
Field 2	‘ ’ —	‘ ’ —	‘ ’ —
	'X'	‘ ’ —	'X'

Active			
		Field 1	
		‘ ’ —	'X'
Field 2	‘ ’ —	‘ ’ —	‘ ’ —
	'X'	‘ ’ —	'X'

Mandatory			
		Field 1	
		‘ ’ —	'X'
Field 2	‘ ’ —	‘ ’ —	'X'
	'X'	'X'	'X'

Highlighted			
		Field 1	
		‘ ’ —	'X'
Field 2	‘ ’ —	‘ ’ —	'X'
	'X'	'X'	'X'

Invisible			
		Field 1	
		‘ ’ —	'X'
Field 2	‘ ’ —	‘ ’ —	'X'
	'X'	'X'	'X'

Description of characters:

_ = switched off (off), X = switched on (on)



Example

If Field 1 makes a screen field invisible (X), Field 2 cannot change this.

Screen Painter Attributes

With screen modification, the system takes into account not only the entries you make during field selection, but also any entries made in Screen Painter. This means that the result of the above combination is linked to the screen field attributes according to the same linking rules as described in *Combination Rules For Attributes*.

To take advantage of the full dynamic modification range, you should use the following attributes in Screen Painter:

```
Input = 'X'  
Output = 'X'  
Mandatory = '_'  
Invisible = '_'  
Highlighted = '_'.
```

Conversely, you cannot change the values defined on the screen in the following manner:

```
Input = '_'  
Output = '_'  
Mandatory = 'X'  
Invisible = 'X'  
Highlighted = 'X'
```

If you enter the following combination of influences, it is not really a valid combination, since the combination rules stipulate that the specified display attributes cannot be changed by another influencing field (or the screen).

```
Input = 'X'  
Output = 'X'  
Active = 'X'  
Mandatory = '_'  
Highlighted = '_'  
Invisible = '_'
```

When you go into field selection again, any such ineffective influence are not displayed, except in the case where you have defined a default value for the influencing field; here, display and maintenance of such an influence can be useful.

Generating the Field Selection

If the list of modified fields has changed at all, you must generate the field selection. This produces consecutive numbers for the modified SCREEN-GROUP1 fields in the screens of the relevant module pool.

To do so, choose *Generate* in the transaction SFAW.

Function Modules for Field Selection

To activate field selection for a screen, at the PROCESS BEFORE OUTPUT event, you can call either FIELD_SELECTION_MODIFY_ALL or FIELD_SELECTION_MODIFY_SINGLE. Both these function modules determine the contents of the influencing fields, refer if necessary to the combination rules and execute the screen modification. FIELD_SELECTION_MODIFY_ALL executes the LOOP AT SCREEN statement itself. However, with FIELD_SELECTION_MODIFY_SINGLE, you must code this yourself and call the function module within this loop. You can thus perform your own additional screen modifications within the LOOP.

Examples of calling the function modules in the event PBO:

```
CALL FUNCTION 'FIELD_SELECTION_MODIFY_ALL'
  EXPORTING MODULEPOOL = MODULEPOOL
            SCREENGROUP = SCRGRP.
```

or

```
LOOP AT SCREEN.
  IF SCREEN_GROUP1 NE SPACE AND
     SCREEN-GROUP1 NE '000'.
    CALL FUNCTION 'FIELD_SELECTION_MODIFY_SINGLE'
      EXPORTING MODULEPOOL = MODULEPOOL
                SCREENGROUP = SCRGRP.
  * Separate special rules
  MODIFY SCREEN.
ENDIF.
ENDLOOP.
```

or

as a), but includes your own LOOP AT SCREEN for special rules.

You must decide in each individual case which of the options b) or c) elicits the best performance.

Since the *Module pool* and *Screen group* parameters determine the field selection, you must maintain influences for these.

The *Module pool* parameter defines, in main memory, which loaded module pool you use to search for the current values of the influencing fields.



Note

When you call a function module, you must not directly include the system fields SY-REPID and SY-DYNGR. Instead, transfer their contents to other fields at a suitable code position, for example:

```
MODULEPOOL = SY-REPID.
SCRGRP     = SY-DYNGR
```

Sometimes, the *Module pool* values may differ from the current SY_REPID value.

If the *Screen group* parameter is blank, the system uses the current contents of SY-DYNGR. This is not possible for the *Module pool* parameter because the value '_' (blank) prevents any field modification.

**Example**

Regard a transaction resembling TZ50 in the development class SDWA (see *Setting Screen Field Attributes*).

Suppose the dynpro of the second screen contains the following module call in the PBO event:

```
PROCESS BEFORE OUTPUT.
```

```
...
```

```
MODULE MODIFY_SCREEN.
```

Suppose the module MODIFY_SCREEN contains the following function call:

```
MODULE MODIFY_SCREEN OUTPUT.
```

```
CALL FUNCTION 'FIELD_SELECTION_MODIFY_ALL'
  EXPORTING
    SCREENGROUP = 'SCREEN'
    MODULEPOOL   = 'SAPMTXXX'
  EXCEPTIONS
    OTHERS       = 1.
```

Suppose that for the screen group SCREEN and the module pool SAPMTXXX the influences in Transaction SFAW are maintained as shown in the figures of *Calling Field Selection*.

After calling the transaction, suppose these entries are made:

After choosing *Change*, the following screen appears:

Changing Screen Field Attributes with the Function Field Selection

From city	FRANKFURT
Destination	NEW YORK
Dest. airport	JFK
Flight time	08:24:00
Departure	10:10:00
Arrival	11:34:00
Distance	6.162
Dist. in	KM

However, if instead of 'LH' as airline carrier 'AA' is entered, the following screen appears:

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

When entering 'LH', the field SPFLI-AIRPFROM is invisible. When entering 'AA', it is visible as *Dep. airport*.

Linking Fields

Every influencing field influences a field which can be modified regardless of other fields. A link of influencing fields is desired in some cases but then only possible via the definition of help fields which you must set in the application program before calling up the function module (see also *Combination Rules For Attributes*).

This restriction helps the transparency of the field selection.

Examples of Links

Assume the following fields:

Influencing fields: F4711, F4712

Field that can be modified: M4711

The following cases can only be implemented via a detour:

OR Condition and "Ready for Input"

If F4711 = 'A' OR F4712 = 'B',
then M4711 is ready for input.

Changing Screen Field Attributes with the Function Field Selection

Solution:

Define H4711 as an influencing field in SFAW;
define the following condition in SFAW:
if H4711 = 'AB'
then M4711 input on (that is, input = 'X')

In the application program, you must program the following before calling up the function module:

```
IF F4711 = 'A' OR F4712 = 'B'.
  H4711 = 'AB'.
ENDIF.
```

AND Condition and "Mandatory"

If F4711 = 'A' AND F4712 = 'B',
then M4711 obligatory and only then.

Solution:

Maintenance in the field selection:
If H4711 = 'AB',
then M4711 is a required-entry field
(H4711 = 'AB' only precisely with the above AND condition)

In the application program, you program:

```
IF F4711 = 'A' AND F4712 = 'B'
  H4711 = 'AB'
ELSE.
  H4711 = ....
ENDIF.
```

On the other hand, you can represent the following cases directly:

AND Condition and "Ready for Input"

If F4711 = 'A' AND F4712 = 'B',
then M4711 is ready for input.
thus: if F4711 <> 'A' OR F4712 <> 'B'

Solution:

Screen: M4711 ready for input

Field selection:

Influencing field	F4711	Value 'A'	Input = 'X'
		Value 'A1'	Input = ' '
		Value 'AX'	Input = ' '
Influencing field	F4712	Value 'B'	Input = 'X'
		Value 'B1'	Input = ' '
		Value 'BX'	Input = ' '

OR Condition and "Mandatory"

If F4711 = 'A' OR F4712 = 'B',
then M4711 is a required-entry field.

Solution:

Screen: Mandatory is switched off

Influencing field F4711 Value 'A',
mandatory = 'X'

Influencing field F4712 Value 'B',
mandatory = 'X'



Note

The possibility to define a NOT condition for an influencing field gives further variants of the field selection definition.

The Display Attribute 'Active'

At present, this display attribute has only one consequence. If SCREEN-ACTIVE = '0' in the ABAP/4 program, the following occurs at runtime with the MODIFY SCREEN statement:

```
SCREEN-INPUT = '0'  
SCREEN-OUTPUT = '0'  
SCREEN-INVISIBLE = '1'
```

If SCREEN-ACTIVE = '1', nothing at all occurs.

If, on the other hand, SCREEN-INPUT = '0', SCREEN-OUTPUT = '0' and SCREEN-INVISIBLE = '1', the internal result would be SCREEN-ACTIVE = '0' (without consequences).

Even when SCREEN-ACTIVE = '0', a module specified with the relevant FIELD statement in the screen flow logic always runs. However, a module with a deactivated field is not meant to run. If you do not intend this, you can separate the FIELD and MODULE statements with a period.

In the case of inactive fields with additional specifications like ON INPUT, ON REQUEST ..., the screen modules do not run, whereas modules without additional specifications are always processed.

Authorization for Field Selection

The authorization object for *Field selection* is "central field selection" (S-FIELDSEL). This object consists of an activity and a program authorization group. The latter is taken from the program authorizations.

The following activities are allowed:

- '02' = Change
- '03' = Display
- '14' = Generate field selection information on screens
- '15' = Assign relevant tables to field selection

Using Subscreens

A subscreen is an independent screen that is displayed in an area of another (“main”) screen. You might want to use a subscreen to vary certain fields in a main screen. For instance, you can define an area in a main screen where supplementary fields appear, depending on the input the user makes in a previous screen.

You create a subscreen as follows:

1. Adjust the frame of the subscreen within the “main” screen until it has the size you want. Name the subscreen in the *Field name* field.
2. Create a screen with screen type *Subscreen*.
3. Arrange the fields within the subscreen so that they appear in the main screen exactly where you want them to be.

If the subscreen is defined to be larger than the available area in the main screen, then only that part of the subscreen will be visible that fits in the area available (measured from the upper left corner).

To use a subscreen, you must call it in the flow logic (both PBO and PAI) of the main screen. The CALL SUBSCREEN statement tells the system to execute the PBO and PAI events for the subscreen as part of the PBO or PAI events of the main screen. You program the ABAP/4 modules for a subscreen exactly as you would program modules for a main screen.

The flow logic of your main program should look as follows:

```
PROCESS BEFORE OUTPUT.  
  ...  
  CALL SUBSCREEN <area> INCLUDING '<program>' '<screen>'.  
  ...  
PROCESS AFTER INPUT.  
  ...  
  CALL SUBSCREEN <area>.  
  ...
```

Area is the name of the subscreen area you defined in your main screen. This name can have up to ten characters. *Program* is the name of the program to which the subscreen belongs and *screen* is the subscreen’s number.

If the subscreen and the main program lie in different module pools, then neither can know about the global data in the other. Data in the main program must be explicitly passed to (and from) the subscreen. (This is usually done using a function module that exports or imports the data, with the corresponding MOVE statement in the code for the subscreen.)

Multiple subscreens are possible in a single screen. You can also specify the subscreens dynamically at runtime. Subscreens have several restrictions. They can not:

- set their own GUI status
- have a named *OK code*
- call another screen
- contain an AT EXIT-COMMAND module
- support positioning of the cursor.

Manipulating the Cursor

In the PBO event, you can tell the system to place the cursor in a certain screen field. Cursor positioning is one way to make your transaction more user-friendly.

Positioning the Cursor in the Screen

When displaying a screen, the system automatically places the cursor in the first field ready for input.

However, you can specify the field where the cursor is to appear yourself. You have two options.

1. Enter the appropriate field name as the *Cursor position* when you define the screen in the Screen Painter. The *Cursor position* field appears in the *Screen Attributes*.
2. You can also set the cursor using the ABAP/4 command SET CURSOR FIELD. Use the SET CURSOR statement as follows:

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

The <field name> can be an explicit field name in quotes, or a variable containing the field name. To place the cursor in a particular position in a field, use the OFFSET parameter, entering the character position for the cursor in <position>:

```
SET CURSOR FIELD <field name> OFFSET <position>.
```

The system offsets the cursor by the given position from the start of the line.

If you have a step-loop in your screen, you can place the cursor on a particular element in the step-loop block. Use the LINE parameter, entering the line of the loop block where you want to place the cursor:

```
SET CURSOR FIELD <fieldname> LINE <line>.
```

If you want, you can use the OFFSET and LINE parameters together.