

SAP Automation GUI Code Generator (BC-FES-AIT)



HELP.BCFESDEB

Release 4.6C



Copyright

© Copyright 2001 SAP AG. Alle Rechte vorbehalten.

Weitergabe und Vervielfältigung dieser Publikation oder von Teilen daraus sind, zu welchem Zweck und in welcher Form auch immer, ohne die ausdrückliche schriftliche Genehmigung durch SAP AG nicht gestattet. In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden.

Die von SAP AG oder deren Vertriebsfirmen angebotenen Software-Produkte können Software-Komponenten auch anderer Software-Hersteller enthalten.

Microsoft[®], WINDOWS[®], NT[®], EXCEL[®], Word[®], PowerPoint[®] und SQL Server[®] sind eingetragene Marken der Microsoft Corporation.

IBM[®], DB2[®], OS/2[®], DB2/6000[®], Parallel Sysplex[®], MVS/ESA[®], RS/6000[®], AIX[®], S/390[®], AS/400[®], OS/390[®] und OS/400[®] sind eingetragene Marken der IBM Corporation.

ORACLE[®] ist eine eingetragene Marke der ORACLE Corporation.

INFORMIX[®]-OnLine for SAP und Informix[®] Dynamic Server[™] sind eingetragene Marken der Informix Software Incorporated.

UNIX[®], X/Open[®], OSF/1[®] und Motif[®] sind eingetragene Marken der Open Group.

HTML, DHTML, XML, XHTML sind Marken oder eingetragene Marken des W3C[®], World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA[®] ist eine eingetragene Marke der Sun Microsystems, Inc.

JAVASCRIPT[®] ist eine eingetragene Marke der Sun Microsystems, Inc., verwendet unter der Lizenz der von Netscape entwickelten und implementierten Technologie.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo und mySAP.com sind Marken oder eingetragene Marken der SAP AG in Deutschland und vielen anderen Ländern weltweit. Alle anderen Produkte sind Marken oder eingetragene Marken der jeweiligen Firmen.

Symbole

| Symbol | Bedeutung |
|-----------------------------------------------------------------------------------|------------|
|  | Achtung |
|  | Beispiel |
|  | Empfehlung |
|  | Hinweis |
|  | Syntax |
|  | Tip |

Inhalt

| | |
|-------------------------------------------------------------|-----------|
| SAP Automation GUI Code Generator (BC-FES-AIT) | 5 |
| GUI Code Generator and Related Products | 6 |
| Recording the SAP Session | 7 |
| Starting Session Recording | 8 |
| Opening a Connection | 9 |
| Logging On | 10 |
| Using the Generator GUI | 11 |
| SAP GUI and Code Generator Display Differences | 12 |
| Adding a Program Variable | 13 |
| Creating a New Subroutine | 14 |
| Inserting a Message Check..... | 15 |
| Controls window | 16 |
| Logging Off | 17 |
| Adapting the Generated Program | 18 |
| Understanding Generated Code | 19 |
| Avoiding Menus | 25 |
| Finding Controls | 26 |

SAP Automation GUI Code Generator (BC-FES-AIT)

The SAP Automation Code Generator lets you program SAP alternate user interfaces by example. You can create Visual Basic programs simply by interacting with the SAP system through the code generator. The code generator:

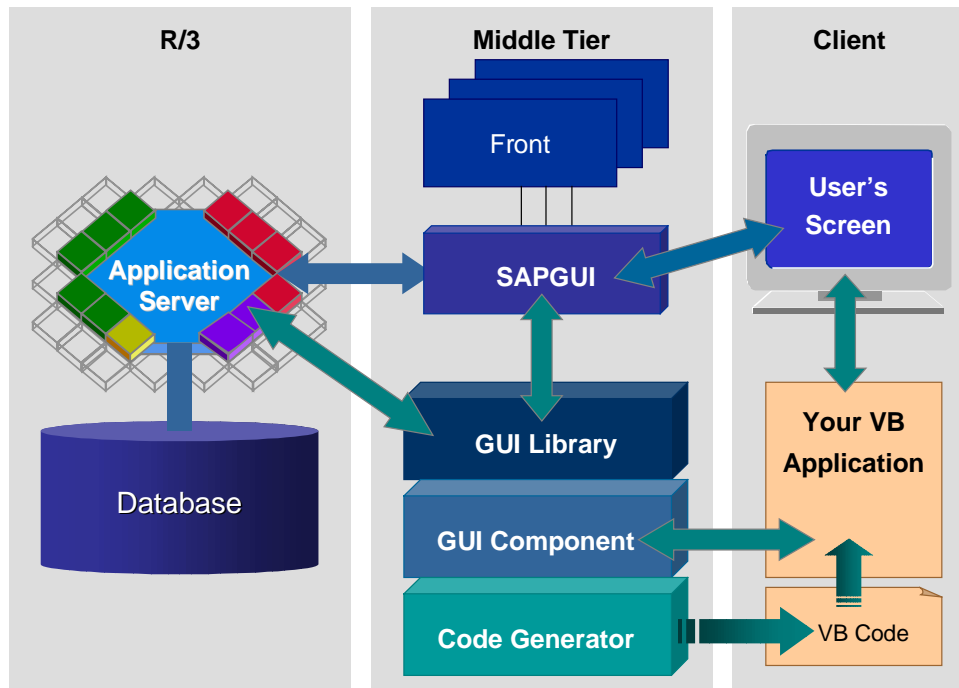
- presents you with a running SAP interface
- records all your actions in the SAP interface
- generates an executable program that mimics your actions

These programs can then be used in tools such as Visual Basic and HAHTsite to create alternate user interfaces to SAP R/3 and R/2 Systems. The code generator simplifies the construction of interactive voice response, World Wide Web, multimedia, or alternative GUI interfaces to R/3.

GUI Code Generator and Related Products

GUI Code Generator and Related Products

The following diagram shows how the GUI Code Generator, which is built upon the GUI Component, can help you write VB applications that record an end-user interaction with a SAPGUI-like screen.



The Code Generator does not replace the SAPGUI screens. It merely records the user interaction with the SAPGUI screens and creates the appropriate VB code.

You can use the code produced by the Code Generator in a VB program to either replay the user interaction or for other purposes. Your VB program that performs the replay needs to use the GUI Component for its interaction with R/3.

Recording the SAP Session

The SAP Automation Code Generator lets you program SAP alternate user interfaces by example. You can create Visual Basic programs simply by interacting with your SAP system through the code generator:

1. [Start session recording \[Seite 8\]](#).
2. [Open a connection \[Seite 9\]](#).
3. [Log on to the R/3 System \[Seite 10\]](#).
4. [Use the Generator GUI \[Seite 11\]](#) to perform all your R/3 activities.
5. [Log off again \[Seite 17\]](#).

Starting Session Recording

Starting Session Recording

When you call up the SAP Automation Code Generator, you get a window with several drop down menu items that displays fields for the database, the CPU, and a selection box that indicates whether or not you are running version 3.0 or later.

Before recording an SAP session, you need to specify some information:

1. Choose the recording format.

By default, the code generator produces programs in Visual Basic. But you can request programs in HAHTtalk Basic or Object Pascal instead. To do this, choose *Generate* → *Recording format*.

In the resulting dialog window, select the kind of program you want and choose *OK*.

2. Start recording.

Choose *Generate* → *Record* to start recording your session. This function calls up a standard "Save as" window to let you specify a file name for the program. The file extensions suggested depend on the recording format you requested.

Specify a file name and choose *Save* to confirm.

3. Open a *Controls* dialog box.

You need to define variables (and other structures) for your program. The *Controls* box lets you specify these by listing all fields available in the current SAP screen. Choose *Generate* → *Controls* to get this box.

Opening a Connection

To open a connection to the SAP System, choose *File* → *Open*. A dialog box appears listing system names from your **saplogon.ini** file. Select a system (or enter system information directly) and press *OK*.

The Code Generator establishes the connection and presents you with an SAP interface. To start the SAP session, you still need to log on. See [Logging On \[Seite 10\]](#).

Logging On

Logging On

To log on, you can either:

- enter logon data in the SAP screen (required for R/2 Systems)
- use the Generator's logon box

Choose *File* → *Logon* (in the SAP Automation GUI Code Generator box) to get a Logon dialog box. Enter your logon data.

When you use this dialog box, the generator can skip the copyright and system messages boxes when generating your program code.

Using the Generator GUI

The Generator GUI and SAP GUI

The Code Generator uses everything you do in the GUI to generate your program. This includes normal user actions on the screen, such as entering data; choosing pushbuttons and menu functions; or receiving system messages.

There are a few restrictions. You cannot use applications that display business graphics or make RFC OLE calls. (If you call them up, nothing happens.) You also cannot use multiple sessions. If an application tries to create a second session, the program enters an endless loop.

In essence though, you conduct an SAP session in the Generator GUI just as you would in the standard SAP interface. The differences are described in:

[SAP GUI and Code Generator Display Differences \[Seite 12\]](#)

[Avoiding Menus \[Seite 25\]](#)

Requesting Generator Code

The Generator also lets you specify additional program code. You can request:

- [variable declarations \[Seite 13\]](#)
- [message checks \[Seite 15\]](#)
- [subroutines \[Seite 14\]](#)

For example, whenever your program inputs data or receives output (to or from a screen field), you must define a program variable for the field. For more information, see:

[Controls window \[Seite 16\]](#)

SAP GUI and Code Generator Display Differences

SAP GUI and Code Generator Display Differences

You conduct an SAP session in the Generator GUI just as you would in the standard SAP interface. There are, however, some differences:

- **Menus appear as tree structures**
SAP menus appear in the left column of the Generator display. To choose menu items, single-click on them. The system displays sub-menus, or jumps to the screen for the menu function. The ampersand symbol (denoting the SAP underscore) precedes the character defined as shortcut key for the menu function.
- **Toolbars contain only icons, no text buttons**
SAP toolbar pushbuttons appear as help buttons in the Generator display. Drag the mouse cursor over the button for a display of the button text.
- **Modal dialog boxes appear as whole screens**
A modal dialog box replaces the current screen, with the modal toolbox displayed at the top (rather than the bottom) of the screen.
- **Standard Windows 95 font is used**
The change in font can cause some misalignment in fields.

Unsupported SAP Features

Some SAP features are unsupported in SAP Automation. You cannot use:

- **business graphics or RFC/OLE.**
If you call applications that use these, nothing happens.
- **multiple SAP sessions**
You cannot use multiple sessions. The System>Create session menu function is disabled. If you call applications that create a second session, the application enters an endless loop.

Adding a Program Variable

To add a program variable:

1. Select the field in the field list.
The Generator suggests a variable name for the field in the *Name* field.
2. In the *Name* field, modify the variable name, if desired.
3. Ensure that the *Name Type* field is set correctly to *Input* or *Output*
4. Choose *Add Name*.

The Generator inserts a variable declaration and related code. This code varies depending on whether you requested an input or output variable. For an input variable, the Generator assumes you want to transfer a value to an input field in an SAP screen. As a result, you get code like:

```
Dim DevClassVal As String
bOK = Sap.SetControlValue(iCtrl, DevClassVal)
```

where the requested variable is used as the source field for an assignment to the currently selected control. When adapting the program, of course, you must ensure that the input variable has the correct value (probably from an external interface).

For an output value, the Generator assumes you want a variable to be used as a target field. The corresponding control is the source field in the assignment:

```
Dim PostalCode As String
PostalCode = Sap.Controls(iCtrl).Value
```

Creating a New Subroutine

Creating a New Subroutine

The Generator places all generated statements in a subroutine. You can specify a new subroutine whenever desired. When you do, the Generator closes the current subroutine and starts a new one. Variables needed in every subroutine are also added:

```
...  
End Sub  
  
Sub NewRoutine()  
Dim iCtrl As Integer          ' Control index variable  
Dim bOK As Boolean           ' Return code variable  
...
```

All session actions after this point are placed in the new subroutine. All actions performed before you have defined a subroutine go in a default subroutine started at the beginning of your session.

Use the *Controls* window to specify the new subroutine:

1. Select *Subroutine* in the *Name Type* field.
2. In the *Name* field, modify the suggested subroutine name, if desired.
3. Choose *Add Name*.

Inserting a Message Check

You can insert code that checks whether the system has sent a message. If the program finds a message, it displays:

```
Dim msg1 As String
  If Sap.MessageFlag Then
    ' MsgBox Sap.Message
    msg1 = Sap.Message
  End If
```

This code is useful for trapping success or error messages. For example, the generated program can identify that an operation has succeeded and display it to the user.

Controls window

Controls window

Use the *Controls* window to specify input and output variables, messages and subroutines. For convenience, this window lists all the fields in the current screen by name and description.



You must have turned on recording (choose *Generate* → *Record* from the code generator menu) to add elements with the *Controls* window. However, the field display is always available.

For information on using the *Controls* window, see:

[Adding a Program Variable \[Seite 13\]](#)

[Creating a New Subroutine \[Seite 14\]](#)

[Inserting a Message Check \[Seite 15\]](#)

Logging Off

You can log off by choosing:

- *File* → *Logoff* in the Code Generator menu (the most efficient method)
- *System* → *Logoff*, or any other logoff method in the SAP GUI

Both methods automatically close the connection with the SAP System. You don't need to do this explicitly.

Adapting the Generated Program

Adapting the Generated Program

After you generate a Visual Basic program, you can adapt it as desired. See the following tips on using generated code.

[Understanding Generated Code \[Seite 19\]](#)

[Avoiding Menus \[Seite 25\]](#)

[Finding Controls \[Seite 26\]](#)

Understanding Generated Code

This section presents a generated program. In this example, the recording user uses transaction PA20 (*Display Human Resources Master Data*) to perform two tasks:

- get an employee's *Addresses*
- get an employee's *Maternity Protection* data

In the course of the session, the recorder requests input and output variables, a second subroutine ("Maternity") for the Maternity Protection display, and message checking code. The message checking code is requested at the point where, should a user enter a male employee, the system responds with a message that maternity protection applies only to females.



You can always display and edit the generated program in Windows' Notepad, as well as in Visual Basic, HAHTtalk Basic or Object Pascal.

```

-----
' The ScreenCheck routine is always generated, and checks
' whether the next screen recorded in the program is in fact the
' next screen generated by the runtime system.
-----
Private Sub ScreenCheck(ByVal ProgName As String, _
                        ByVal ScrName As String)
    If Sap.ProgramName <> ProgName Or _
        (Len(ScrName) > 0 And Sap.ScreenName <> ScrName) Then
        MsgBox "Unexpected screen " & ProgName & " " & ScrName,
vbCritical, "Playback"
        Stop
    End If
End Sub

-----
' The OKCheck routine is always generated, and checks the
' value of a Boolean. This routine is most often used to check the
' value of the bOK variable (return code variable for calls to the
' SAP System).
-----
Private Sub OKCheck(ByVal bIsOK As Boolean, ByVal sMsg As String)
    If Not bIsOK Then
        MsgBox sMsg, vbCritical, "Playback"
        Stop
    End If
End Sub

-----
'
' Default subroutine begun, and variables/controls set up
-----

```

Understanding Generated Code

```

-
Sub RecordedMacro()
    Dim iCtrl As Integer          ' Control index variable
    Dim bOK As Boolean           ' Return code variable
    ' Uncomment following 2 lines if Sap is not declared globally
    ' Dim Sap As Object
    ' Set Sap = CreateObject("SapTerminal.Event")

-----
-
' The recorder logs on, and the resulting SAP interface is
' positioned on the screen.
-----
-
    bOK = Sap.Connect("orlando1", "61", SapGuiMerlin)
    OKCheck bOK, "Error in opening connection"

    Sap.RowDimension = 24
    Sap.RowListDimension = 24
    Sap.ColumnDimension = 80
    Sap.ColumnListDimension = 80
    Sap.SetSizeFlag = True
    OKCheck bOK, "Error in opening connection"

    ' txtPassword is placeholder for actual password value
    bOK = Sap.Logon("", "good", txtPassword, "")
    OKCheck bOK, "Error in logon"
    ScreenCheck "SAPMSYST", "0040"

-----
-
' The recorder calls transaction PA20.
-----
-
    Sap.OKCode = "/npa20"
    bOK = Sap.SendEvent
    OKCheck bOK, "Error in sending default key"
    ScreenCheck "SAPMP50A", "1000"

-----
-
' The recorder selects the Personnel Number field
-----
-
    iCtrl = Sap.FindByField("RP50G-PERNR", 0, SapMatch)
                                ' Personnel number (index 1)

-----
-
' The recorder requests an input variable definition for the
' Personnel Number field in the VB program.

```

Understanding Generated Code

```
"-----  
-  
  Dim PersonnelNumber As String  
  bOK = Sap.SetControlValue(iCtrl, PersonnelNumber)  
  OKCheck bOK, "Error in setting input variable"  
  
"-----  
-  
" Input variable definition for a TxtPers field.  
"  
"  
  Dim txtPers As String  
  bOK = Sap.SetControlValue(iCtrl, txtPers)  
  OKCheck bOK, "Error in setting input variable"  
  
"-----  
-  
" Recorder enters "1502" in Personnel Number field  
"  
"  
  bOK = Sap.SetControlValue(iCtrl, "1502")  
  OKCheck bOK, "Error in setting text value"  
  
"-----  
-  
" Recorder selects "Addresses" data for the employee  
"  
"  
  iCtrl = Sap.FindByValue("Addresses", 0, SapAnyType, SapLeft) '  
index 18  
  bOK = Sap.SetControlSelected(iCtrl, True)  
  OKCheck bOK, "Error in setting selected value"  
  
"-----  
-  
" The Code Generator always repositions the cursor, but these  
statements  
" are not needed in this case. (You can delete them.)  
"  
"  
  bOK = Sap.SetCursorByControl(iCtrl)  
  OKCheck bOK, "Error in setting cursor position"  
  
"-----  
-  
" Recorder requests Display with F2 function key  
"  
"  
  bOK = Sap.SendKey(vbKeyF2)      ' Display  
  OKCheck bOK, "Error in sending key"  
  ScreenCheck "MP000600", "2001"
```

Understanding Generated Code

```

-----
-
" Recorder requests output variable definitions for address data
-----
-
    iCtrl = Sap.FindByField("P0006-NAME2", 0, SapEdit)
                        ' c/o (index 17)
    Dim co As String
    OKCheck iCtrl >= 0, "Error in setting output variable"
    co = Sap.Controls(iCtrl).Value

    iCtrl = Sap.FindByField("P0006-STRAS", 0, SapEdit)
                        ' Street and house no. (index 19)
    Dim StreetAndHouseNo As String
    OKCheck iCtrl >= 0, "Error in setting output variable"
    StreetAndHouseNo = Sap.Controls(iCtrl).Value

    iCtrl = Sap.FindByField("P0006-PSTLZ", 0, SapEdit)
                        ' Postal code/City (index 21)
    Dim PostalCode As String
    OKCheck iCtrl >= 0, "Error in setting output variable"
    PostalCode = Sap.Controls(iCtrl).Value

    iCtrl = Sap.FindByField("P0006-ORT01", 0, SapEdit)
                        ' Postal code/City (index 22)
    Dim city As String
    OKCheck iCtrl >= 0, "Error in setting output variable"
    city = Sap.Controls(iCtrl).Value

    iCtrl = Sap.FindByField("P0006-ORT02", 0, SapEdit)
                        ' District (index 24)
    Dim District As String
    OKCheck iCtrl >= 0, "Error in setting output variable"
    District = Sap.Controls(iCtrl).Value

    iCtrl = Sap.FindByField("P0006-LAND1", 0, SapMatch)
                        ' Country key (index 26)
    Dim CountryKey As String
    OKCheck iCtrl >= 0, "Error in setting output variable"
    CountryKey = Sap.Controls(iCtrl).Value

    iCtrl = Sap.FindByField("P0006-NAME2", 0, SapEdit)      ' c/o (index
17)
    bOK = Sap.SetCursorByControl(iCtrl)
    OKCheck bOK, "Error in setting cursor position"

-----
-
" Recorder returns with F3 function key
-----
-
    bOK = Sap.SendKey(vbKeyF3)      ' Back
    OKCheck bOK, "Error in sending key"

```

Understanding Generated Code

```

ScreenCheck "SAPMP50A", "1000"

-----
-
' Recorder terminates default subroutine, and defines a new
' "Maternity" subroutine. All subsequent actions are recorded
' as part of the new subroutine.
-----
-

End Sub

Sub maternity()
  Dim iCtrl As Integer
  Dim bOK As Boolean

  iCtrl = Sap.FindByField("RP50G-PERNR", 0, SapMatch)
                                ' Personnel number (index 1)
  Dim PersonnelNumber As String
  bOK = Sap.SetControlValue(iCtrl, PersonnelNumber)
  OKCheck bOK, "Error in setting input variable"

-----
-
' Recorder selects Maternity Protection data option
-----
-

  iCtrl = Sap.FindByValue("Maternity Protection", 0, _
                                SapAnyType, SapLeft) ' index 23
  bOK = Sap.SetControlSelected(iCtrl, True)
  OKCheck bOK, "Error in setting selected value"

  bOK = Sap.SetCursorByControl(iCtrl)
  OKCheck bOK, "Error in setting cursor position"

-----
-
' Recorder requests Display with F2
-----
-

  bOK = Sap.SendKey(vbKeyF2) ' Display
  OKCheck bOK, "Error in sending key"
  ScreenCheck "SAPMP50A", "1000"

-----
-
' Recorder requests message-checking code. If a system msg
' generated at runtime, its text will be assigned to the VB variable
' maternityMsg.
-----
-

  Dim maternityMsg As String

```

Understanding Generated Code

```
If Sap.MessageFlag Then
    ' MsgBox Sap.Message
    maternityMsg = Sap.Message
End If

iCtrl = Sap.FindByField("RP50G-PERNR", 0, SapMatch)
        ' Personnel number (index 1)
bOK = Sap.SetCursorByControl(iCtrl)
OKCheck bOK, "Error in setting cursor position"

'-----
-
' Recorder returns with F3
'-----
-
    bOK = Sap.SendKey(vbKeyF3)      ' Back
    OKCheck bOK, "Error in sending key"
    ScreenCheck "SAPMSYST", "0040"

'-----
-
' Recorder logs off using Code Generator option File->Logoff
'-----
-
    bOK = Sap.Logoff
    OKCheck bOK, "Error in logging off"

    ' Uncomment next line for standalone SAP program
    ' Sap.Quit
End Sub
```

Avoiding Menus

SAP menus are available for use in the Generator GUI, but selecting them does not lead to maintainable programs. SAP Automation accesses menus by their menu number (that is, their ordering in the menu bar):

```
Sap.MenuToSend = 6           ' &System  
bOK = Sap.SendEvent
```

However, menu numbers in a screen are subject to change. As a result, SAP makes these recommendations:

- When recording a session, use function keys or OK codes where possible (rather than menus). The ampersands in the menu texts tell you the shortcut keys defined for each menu function.
- When modifying a generated program, use the corresponding transaction codes or function key codes whenever possible. For example, rather than choosing the menu entry *Overview->Single line entry* for orders, use the function key:

```
bOK = Sap.SendKey(vbKeyF5)
```

To use a transaction code (instead of the menu path *Logistics → SID → Sales → Order → Display*), use the transaction code:

```
bOK = Sap.Transaction("va03")
```

Choose *System → Status* to find out the appropriate transaction code.

Finding Controls

Finding Controls

Whenever the user places the mouse cursor in an SAP field, the GUI Code Generator must identify the control corresponding to that field. While recording an SAP session, this information is automatically known. The controls originally created for the screen contain a control number, and the field's repository name and screen label, if available. The Generator uses this information to mark a control as selected, or perform other operations.

At runtime, however (when the generated program runs), the recorded information may no longer be valid. The collection of controls created for the runtime screen may differ from the collection used at program generation. This would be true if:

- a field's repository name has changed (rare, but possible)
- the system language was different (field labels are language-dependent)
- screen fields have changed (control numbers will change correspondingly)

For these reasons, the Generator must generate code that matches the assumed field with the controls that are actually generated at runtime. The code accesses these controls by repository name where possible, or by label name. Code for identifying the control (variable `iCtrl`) might be generated as follows:

```
iCtrl = Sap.FindByField("P0006-LAND1", 0, SapMatch)
                                ' Country key (index 26)
OKCheck iCtrl >= 0, "Error in setting output variable"
CountryKey = Sap.Controls(iCtrl).Value
```

The `OKCheck` routine checks that `iCtrl` does in fact contain a valid value, meaning that it found a control with a name matching the repository name `P0006-LAND1`.

If repository names and labels are not available, however, the Generator uses hard-coded control numbers to access controls:

```
CountryKey = Sap.Controls(26).Value
```

This kind of access is unreliable and should be avoided if possible. SAP recommends that you replace hard-coded control numbers with other code that navigates to the desired control.

For example, you might find a known control (identifiable by field name) and then navigates so many to the right, left, or down, from the landmark. For doing this, the following methods are useful:

- `FindByField`
- `FindByValue`
- `FindExtended`
- `FindFrom`
- `FindFromExtended`