SAP Automation GUI Interfaces (BC-FES-AIT)

HELP.BCFESDE9

Release 4.6C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Contents

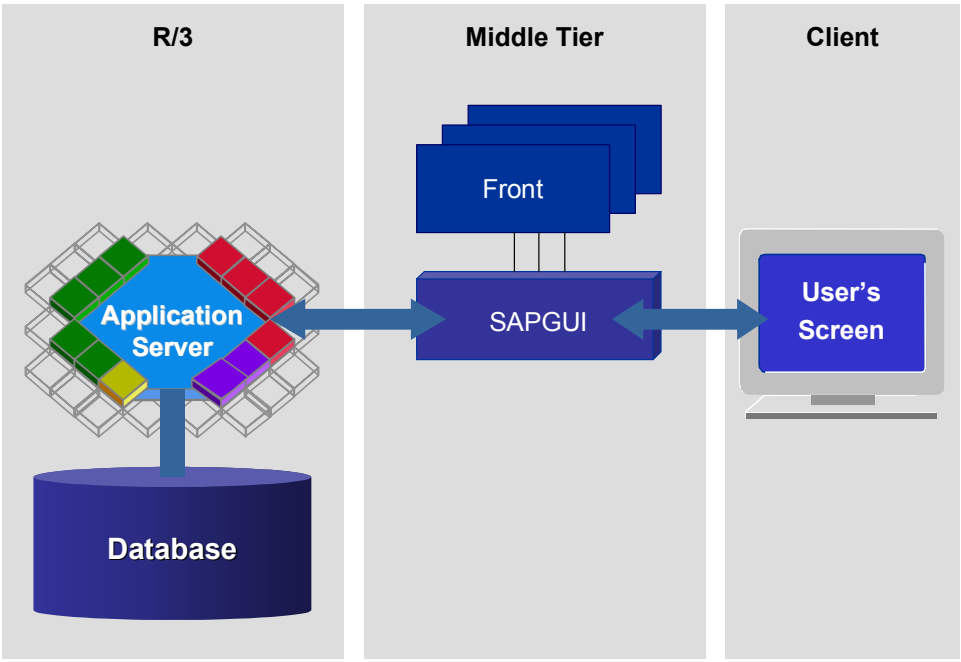
SAP Automation GUI Interfaces (BC-FES-AIT)

SAP Automation GUI Interfaces (BC-FES-AIT)

SAP GUI Interfaces

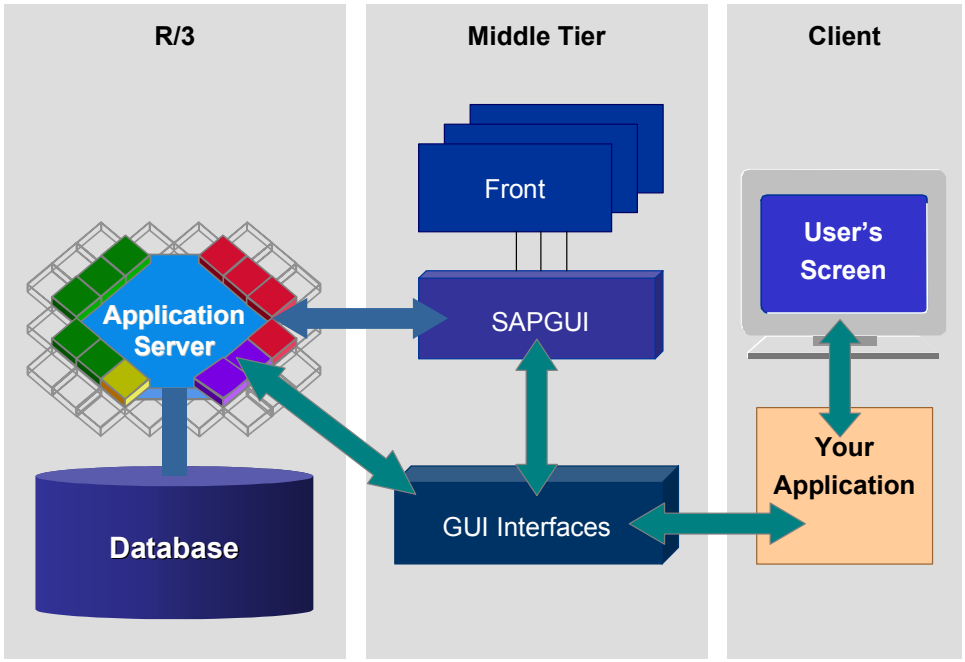
The R/3 application server sends and receives data to and from its SAPGUI when displaying R/3 application screens to an end user. The SAPGUI process interacts with one or more Front processes, one for each user session.

The following diagram shows the standard SAPGUI and its Front processes.



Using the GUI interfaces to R/3, an external program can access the data that is communicated between the R/3 application server and its SAPGUI as a method of getting data from or into R/3.

As the following diagram illustrates, the GUI interfaces allow an external program to access R/3 screen data by communicating with the R/3 application server and the SAPGUI.



By using the GUI interfaces an external program can replace the standard SAPGUI with another user interface, which can be either graphical or non-graphical (for example, it can be voice driven).

Programming using the GUI interfaces also allows an external program to monitor or record an end user interaction with SAPGUI screens.

Using the GUI interfaces method requires no knowledge of ABAP programming. It also eliminates the need to learn the business and transaction logic behind an existing R/3 application for which the external program is providing an alternative user interface.

However, since R/3 transaction screens may change between R/3 versions, using this method of integration with R/3 is the most vulnerable to changes.

SAP Automation GUI Interfaces

SAP Automation [Ext.] is a suite of products that allow external programs to integrate with R/3.

SAP Automation GUI Interfaces is a set of products that uses the SAP GUI interface (also called GUI channel).

These products allow desktop and PC programmers to:

- **Develop another GUI interface for existing R/3 applications**

  The alternative GUI you develop may better fit your users or it may better satisfy the application requirements in your company.

- **Integrate alternative interfaces with R/3**

You can integrate non-GUI interfaces, such as interactive voice response (IVR) telephone systems, multimedia kiosks, or World Wide Web pages, into external programs that access R/3.

For example, the Human resources (HR) component of R/3 is geared towards the HR department personnel entering data through the standard SAPGUI set of transactions and screens. You may use the SAP Automation GUI interfaces to create an application that allows any employee to enter their personal data over the Web or even through the regular phone system, instead.

The advantage of using the SAP Automation GUI Interfaces as a way to provide alternative interface to R/3 applications, over programming using RFC or business object BAPIs, for example, is that you preserve the business rules and application logic built into the various R/3 screens and transactions. With SAP Automation GUI Interfaces you are replacing only the interface to the R/3 application; you are not building the R/3 application from start to finish. You are therefore leveraging on the research and development effort put forth by the SAP development team into designing business process, and into integrating the application with the rest of the enterprise software. For example, when programming an alternative personal data entry as in the above example, you provide a different way for an employee to change his or her address. You do not have to then program the functions that propagate the change of address into the payroll system. You leave this functionality to the R/3 HR component.

Use the SAP Automation GUI interfaces if you need a quick way to provide new interfaces to integrate with an existing R/3 application. However, if you need to develop a new application and integrate it to an R/3 system, the better alternative is a combination of programming in ABAP and using SAP Automation RFC interfaces.

- **Monitor or record an interaction of an end user with R/3 screens**

  Client applications can use the SAP Automation GUI interfaces alongside the standard SAPGUI, recording, for example the user's data entry at the SAPGUI screens.

  Using the SAP Automation GUI Interfaces a client application can then play back the user actions on R/3 screens.

  Recording an end-user interaction can be used for testing or for user interface design, for example.

SAP Automation GUI can therefore be viewed as a first enabling step to moving R/3 interface development to the presentation server, leaving the business logic at the ABAP application level.

Examples of Application

Partner companies have developed several new application interfaces with SAP Automation GUI. Here are some examples:

- A kiosk application for self-service entry, update, and review of personnel qualifications
- Telephone interfaces to financial and logistics applications
- An interface for service monitoring of the R/3 plant management system, which can call or fax the responsible party when it sees that equipment is failing frequently

Within SAP, we have used SAP Automation GUI to develop self-service Human Resources applications.

Availability

Though SAP Automation GUI software began shipping with R/3 Desktop Integration in Release 3.0C, the software works with Release 2.1 and later of the R/3 application servers. Starting with 3.1H, the SAP Automation GUI software also works with R/2 systems.

Components of the SAP Automation GUI Interfaces

The following table describes the components of the SAP Automation GUI Interfaces product.

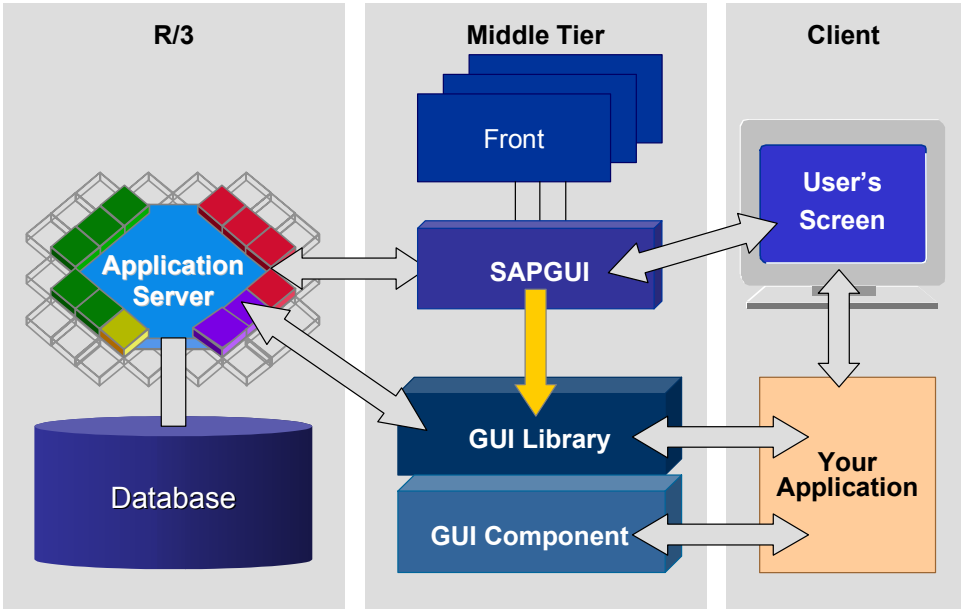
Component	Platform and Files	Description
GUI Library [Page 30]	C-language API on Windows. The library is available in the 32-bit DLL form. (GULlib.dll GUILib.lib)	A set of APIs that, like "screen-scraping" programs, take the contents of R/3 screens and make them accessible to your C program through standard data structures and function calls. Using these APIs you can write programs that provide an alternative interface (GUI or non-GUI) to the standard R/3 application screens.
GUI Component [Page 179]	Available both as an out-of-process server and an OLE control: (ITOLE.exe ITOLE.ocx)	The GUI component exposes most of the GUI Library functionality to Microsoft COM-compliant programs and applications, such as programs written in Visual Basic. It also opens access to R/3 to Windows applications that can serve as OLE Automation controllers, including Microsoft Excel, Lotus Notes, Borland's Delphi, and HAHTSITE. The GUI Component is easier to use than the underlying GUI Library, but it does not offer the full range of functionality that the GUI Library offers.
Code Generator [Page 278]	Autosap.exe Autosap.ocx	As the highest-level SAP Automation GUI Interfaces, the GUI Code Generator is a macro recorder for the GUI Component. The GUI Code Generator lets you generate Visual Basic code as you work with the SAP system, specifying input and output fields as desired, and also lets you capture SAP screens as Visual Basic forms. You can also generate the code in HAHTtalk Basic or Delphi's Object Pascal language. You can customize the GUI Code Generator for your specific needs.

The SAP Automation GUI kit includes the complete source code for the GUI Code Generator. The sources provide a detailed example of how to use the SAP Automation GUI components.

New Features in Release 4.5

Release 4.5A

- The SAP Automation GUI now uses GUILib to communicate with R3 instead of the Merlin interface.
- The GUI Component is available as an OLE control as well as an out-of-process executable server.
- The GUI Component now supports early IDispatch binding, thus allowing type-safe binding from Visual Basic.
- Both the GUI Library and the GUI Component now support the ability to get an event from Front. This means that you can capture end-user actions on the standard SAPGUI Front directly, as shown in the following illustration:



To take advantage of this feature, use the GV_GETFRONTEVENT flag [Page 117] of the It_GetEventEx function [Page 115] when using the GUI Library, or use the GetEventFront method [Page 216] of the SapEvent object when using the GUI Component.

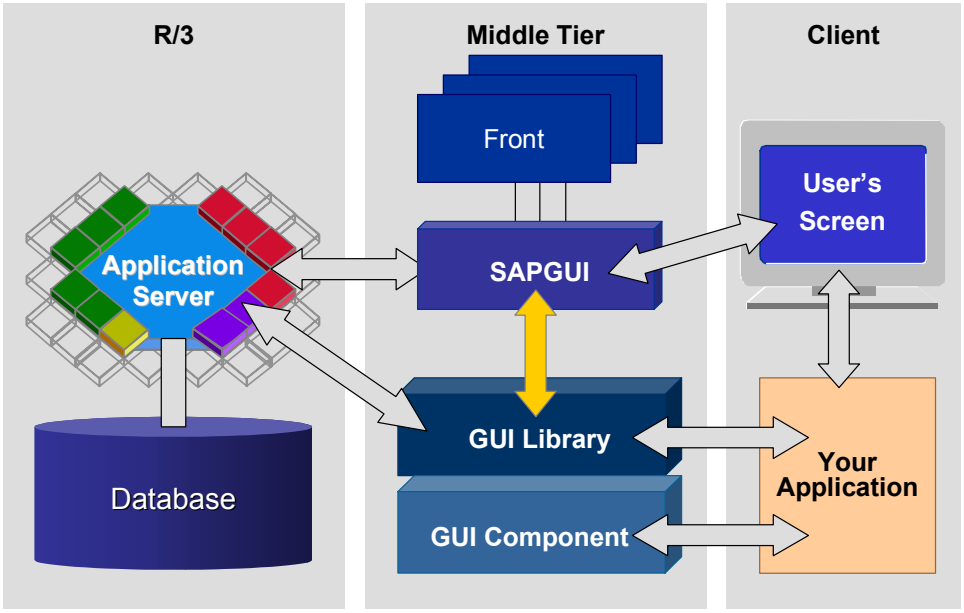
- Multiple sessions are supported by the GUI Library and by the OCX version of the GUI Component. See the topic Using Multiple Sessions for the GUI Library [Page 40] and for the GUI Component [Page 183].
- SAP's /H/S/ style host name specifications are now supported.
- Message and OK Code help are now supported.
- Scrolling without scrolling keys is now supported.
- Hotspot and UppercaseInput styles are now supported.

- New SapEvent properties [Page 198]: Client, DataColumns, DataColumnScreenSize, DataColumnStart, DataRows, DataRowScreenSize, DataRowStart, GetAllMenus, Menus, MenuToSend, MessageHelpFlag, ModalLeft, ModalHeight, ModalTop, ModalWidth, OKHelpFlag, R2, ScrollColumnToSend, ScrollRowToSend, SendScrollColumnsFlag, SendScrollRowsFlags, SetSizeFlag, Username.
- New SapEvent methods: Connect [Page 203], TableEntry [Page 245].
- New SapControl properties [Page 266]: Area, Block, Container, Group, Has3D, HotSpot, TableControlInfo, UppercaseInput.
- New SapKey properties [Page 261]: Info, Order, Toolbar, ToolbarHasIcon, ToolbarIconCode, ToolbarIconName, ToolbarIconText.
- New SapMenu properties [Page 256]: Active, Child, Expanded, Flags, Name, Next, Parent, Popup, PopupRequested.
- New SapTableControlInfo properties [Page 269]: Columns, ColumnSelectionType, ControlOKCode, DataRows, FixedColumns, Flags, Rows, RowSelectionType, ScrollOKCode, StartColumn, StartRow.
- New SapTabStripControlInfo properties [Page 269]: GetNumButtons, GetNumLocalButtons, GetLeftButton, GetActiveButton, SetActiveButton, GetNumButtonRows, GetButtonHeight, GetTabOrientation, GetScrollArrowPos, GetTextOritentation, GetTabStyle.

Release 4.5B

GUI Library

- New It_SendEventEx function [Page 137] expands the functionality of the It_SendEvent function, by allowing you to specify a flag, SV_SENDTOFRONT, with which you can send an event to the SAPGUI Front, instead of to the R/3 application server. See the topic Using the Event Structure [Page 45].



New Features in Release 4.5

- New It_AbortGetEvent function [Page 109] allows you to abort the operation of getting an event.
- New It_SetDelSessionHook function [Page 143] allows you to register a callback watching for a deletion of a session.
- New ItEv_GetSessionCount function [Page 163] gives you the number of active sessions.
- In previous versions if you ran SAPGUI Front alongside an application using the GUI Library you had to use the same release of SAPGUI Front. Now you can use GUI Library applications alongside SAPGUI Front of any R/3 releases starting with release 3.1H or higher.

GUI Component

- The EXE version of the GUI Component now supports multiple sessions
- New SendEventToFront method [Page 227] of SapEvent allows you to send an event to the SAPGUI Front, instead of to the application server.
- New NewServerConnectionEx method [Page 222] of SapEvent allows you to connect to a specific aspplication server.

GUI Code Generator

The GUI Code Generator is now able to render Tab strip controls.

New Features in Release 4.6

Release 4.6A

- New flags for the connection functions (It_NewConnection [Page 127], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131] in the GUI Library and in SapGuiFlags [Page 275] of the GUI Component) allow you to specify how the coordinates of controls on the screen are expressed: whether coordinates of a control are relative to its parent control or to the screen. The flags affect the treatment of coordinates for all events associated with the connection created with the connection function. See the description of the Connection Function Flags [Page 128].
- New GUI Library functions:
  - It_IsGuiRunning [Page 123]
  - ItEv_GetAccelerator [Page 158]
  - ItEv_GetControlTooltip [Page 162]
  - ItEv_SupportFlags [Page 178]
- New GUI Component methods:
  - SendEventToFront
  - SupportFlags

Release 4.6B

No new features were introduced in release 4.6CB.

Release 4.6C

- To accommodate SAPGUI screens with ActiveX controls on them, the GUI Library and the GUI Component provide a new connection flag, (SAPGUI_ACTIVEX in the GUI Library and SapGuiActiveX in the GUI Component).

  Using this flag results in the correct display of ActiveX controls on the SAPGUI Front screen, and it allows the user's actions on ActiveX controls to be communicated back to the R/3 application server.

  However, the support for ActiveX control in the two products is limited: the ActiveX control information bypasses the GUI Library and the GUI Component. As a result, your application may get information describing only part of the controls on the screen.

  When using this flag your application can only get events. It cannot send events.

  For a more detailed description of this feature, see Using Screens with ActiveX Controls [Page 69].
- The various connection functions/methods of the GUI Library and of the GUI Component now support multiple connections.

  This means that you can issue multiple connection function calls thereby creating additional connections.

  GUI Library connection functions (which now allow for multiple connections) are:

It_NewConnection [Page 127], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131]

GUI Component SapEvent connection methods (which now allow for multiple connections) are:

Connect [Page 203], NewGroupConnection [Page 220], NewServerConnection [Page 221], NewServerConnectionEx [Page 222]

You can call different types of connection functions for the different connection, for example, you can call It_NewGroupConnection for one connection, and then use It_NewServerConnection to start a new connection.

Restrictions

The SAP Automation GUI software provided with R/3 4.6C is for Win32 platforms only (32-bit Windows 95, Windows 98, and Windows NT 4.0 and later). On Windows NT 4.0 systems, Service Pack 3 or 4 are strongly recommended.

Documentation and user interfaces in this release are in English only.

This version of the software has several restrictions. Changes may be made to the programming interfaces in future versions of R/3, both to accommodate new uses of the SAP Automation GUI as well as to relax these restrictions.

The following topics discuss the restrictions for the separate products:

GUI Library Restrictions [Page 20]

GUI Component Restrictions [Page 21]

GUI Library Restrictions

- RFC and OLE calls from ABAP to the presentation server are not supported.
- The front-end multiplexer is not supported.
- Status bar events are not supported.
- Compiler support is limited to Microsoft Visual C++ version 5.0.

GUI Component Restrictions

- Early Vtable binding is not supported. However, late binding as well as early DispID binding are supported.
- If you are using Microsoft Visual Basic or Visual C++, support is limited to version 5.0 of these compilers

Kit Contents

If you choose to install the SAP Automation GUI during SAP Automation [Ext.] installation, all the files related to the three SAP Automation GUI products are installed.

This topic lists the files that are installed during a SAP Automation installation. See the separate topics for the files you need to use when working with each of the products:

- GUI Library [Page 24]
- GUI Component [Page 25]
- GUI Code Generator [Page 27]

Program Files

The SAP Automation GUI kit includes the following library and executable files, located in the \SAPGUI\RFCSDK\IT\ directory:

guilib.dll	GUI Library
itole.exe	GUI Component OLE server
itole.ocx	GUI Component OLE control
autosap.exe	GUI Code Generator
itoleapp.tlb	GUI Component OLE server type library
itole.ini	GUI Component OLE server initialization file
picbutton.ocx	Required for working with the Code Generator

Header and Sample Files

In addition, the kit provides the following header and sample files. The following files are initially installed in the RFCSDK\IT subdirectory:

guilib.lib	Library file for linking guilib.dll into C/C++ programs
guilib.h	Header file for GUI Library
sample.c	Sample C program for using the GUI Library
itsheet.xls	Excel sample program for GUI Component
itvfp.prg	Visual FoxPro sample program for GUI Component
autosap.vbp	GUI Code Generator project file

The complete Visual Basic source to the GUI Code Generator is also included. See the Files You Need When Using the Code Generator [Page 27] for detail.

Required Third-party Files

Several Microsoft and third party libraries and controls are also required. They are installed as part of the installation of the SAP Automation suite, if needed. See the discussions on the files you need when using the GUI Component [Page 25] and the GUI Code Generator [Page 27].

Files You Need for Working with the GUI Library

When programming using the GUI Library you need the following files:

File	Comments
guilib.h	Compile with this file
guilib.lib	Library file for linking guilib.dll into C/C++ programs (static link)
guilib.dll	Use at run time. Supply this file to your end-user.

The sample program for using the GUI Library is sample.c [Page 71].

Required DLL Files

Several other DLL files are also required when using the GUI Library. They are installed as part of the installation of the SAP Automation suite [Ext.], if needed.

For example, the *librfc32.dll* file is an SAP DLL used by the GUI Library. It is installed as part of the SAP Automation installation.

Files You Need when Using the GUI Component

You can use the GUI Component in the following ways:

- As a runtime stand-alone executable program (using the EXE version of the GUI Component)
- As an out-of-process ActiveX server (using the EXE version of the GUI Component)
- As an in-process ActiveX server (using the OCX version of the GUI Component)

Files for the Stand-alone Executable Program

File	Comments
guilib.dll	Use at run time. Supply this file to your end-user.
itole.exe	GUI Component OLE server
itole.ini	GUI Component OLE server initialization file
You also need additional DLL files as listed below.	

Files for the Out-of-process ActiveX Server

File	Comments
guilib.dll	Use at run time. Supply this file to your end-user.
itole.exe	GUI Component OLE server
itoleapp.tlb (optional)	GUI Component OLE server type library
You also need additional DLL files as listed below.	

Files for the In-process ActiveX Server

File	Comments
guilib.dll	Use at run time. Supply this file to your end-user.
itole.ocx	GUI Component OLE control
You also need additional DLL files as listed below.	

Required DLL Files

Several other DLL files are also required when using the GUI Component. They are installed as part of the installation of the SAP Automation suite [Ext.], if needed.

For example, the *librfc32.dll* file is an SAP DLL used by the GUI Library. It is installed as part of the SAP Automation installation.

Third-party DLLs

Several Microsoft and other third-party DLLs are also required when using the GUI Component. They, too, are installed as part of the installation of the SAP Automation, if needed:

- mfc42.dll
- msvcrt.dll
- oleaut32.dll
- olepro32.dll

Sample Files

The following are sample files for using the GUI Component:

File	Type
itsheet.xls	Excel sample program
itvfp.prg	Visual FoxPro sample program

In addition, the source files of the Code Generator are a sample of using the GUI Component [Page 27].

Files You Need for Using the Code Generator

Using the Code Generator as a Sample GUI Component Application

The SAP Automation GUI Code Generator is an implementation of an application using the GUI Component. Viewing its source files can help you program similar applications.

GUI Code Generator Source Files

The following are GUI Code Generator source files. :

itAbout.frm	itDynamic.frm	itMainForm.frm
itAbout.frx	itDynamic.frx	itMainForm.frx
itAddComment.frm	itFormBuild.bas	itRecordFormat.frm
itConnect.frm	itHelpInfo.bas	itRunProgram.frm
itControls.frm	itLogon.bas	itSapRunner.cls
itControls.frx	itLogon.frm	itTransaction.frm

In addition, you need to use the GUI Code Generator project file: *autosap.vbp*

Using the Code Generator Application Program

We have also compiled the above source files into the GUI Code Generator program file, which is *autosap.exe*.

The GUI Code Generator program generates Visual Basic modules (.BAS) as a result of user interaction with SAPGUI. You can use this code as a starting point for creating applications using the Basic programming language.

File You Need When Using the Code Generator Application Program

When using the GUI Code Generator as a program to generate Basic code you need the following files:

File	Comments
guilib.dll	GUI Library
itole.ocx	GUI Component OLE control
autosap.exe	GUI Code Generator

Required DLL Files

Several other DLL files are also required when using the GUI Code Generator. They are installed as part of the installation of the SAP Automation suite [Ext.], if needed.

For example, the *librfc32.dll* file is an SAP DLL used by the GUI Library. It is installed as part of the SAP Automation installation.

Third-party DLLs

Several Microsoft and other third-party DLLs are also required when using the GUI Code Generator. They, too, are installed as part of the installation of the SAP Automation, if needed:

- msvcirt.dll
- oc30.dll
- msvbvm50.dll
- comctl32.ocx
- comdlg32.ocx
- vsflex32.ocx

Building and Running the Programs

Building the GUI Library

Use Visual C++ 5.0 to build programs using the SAP Automation GUILib Library interface.

Include the *guilib.h* header file in your source file and link with *guilib.lib*. These files may be moved or copied to your project directories.

Preparing for Using the GUI Component and the Code Generator

Programs using the GUI Component or the Code Generator do not need to link to the GUI Library.

The SAP installation procedure for SAP Automation performs the necessary task of registering the GUI Component control and server with Windows.

However, if you are getting the SAP Automation GUI products outside of the normal releases of the product, you may have to register the GUI Component control and server yourself.

To register the GUI Component server issue the following command:

```
itole -RegServer
```

Similarly, the GUI Component control can be registered by running:

```
regsvr32 itole.ocx
```

To unregister the GUI Component control run the same command with the */u* switch:

```
regsvr32 /u itole.ocx
```

Running SAP Automation GUI Programs Alongside SAPGUI

You can use SAP Automation GUI programs along with the standard SAPGUI on Windows platforms. You can use SAPGUI of R/3 releases 3.1H or higher.

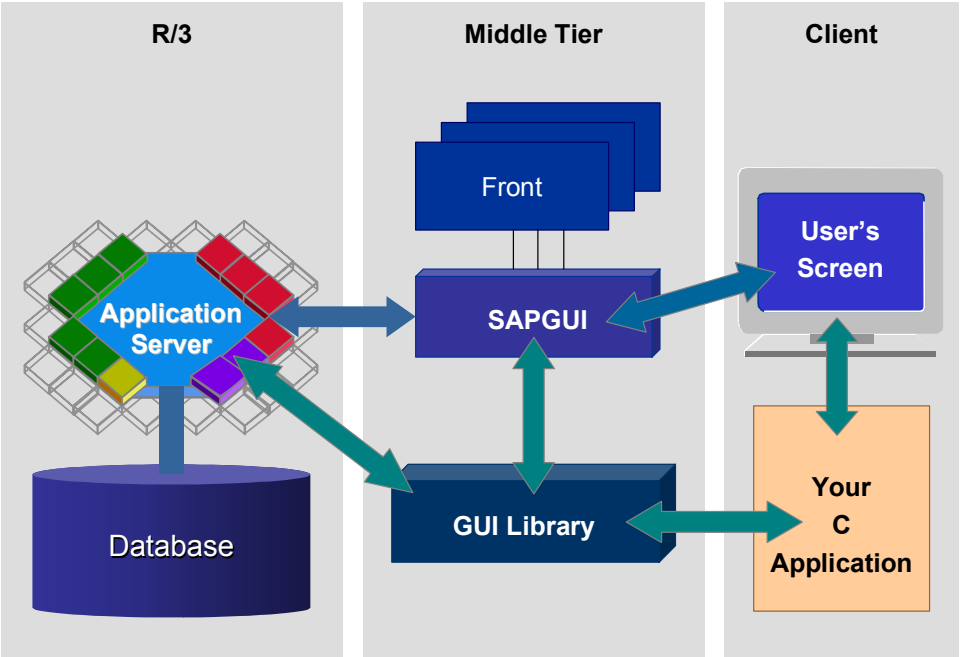
A version of SAPGUI is automatically registered after you run SAPGUI (*sapgui.exe*) of that version once on your machine.

An alternative to having the SAPGUI registered is to have the SAPGUI executable in the same directory.

The GUI Library

GUI Library

The following diagram shows the interaction of the SAP Automation GUI Library with R/3 and with your application, for applications written in C.



When an R/3 end-user access R/3 screens, SAPGUI interacts between the user and the R/3 application server.

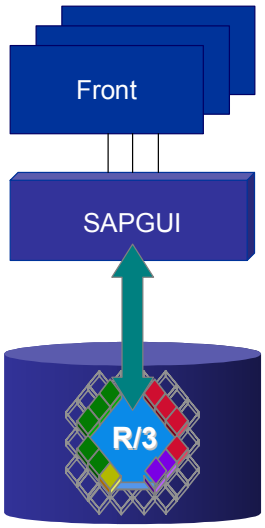
The GUI Library allows your C program to access the data stream that is sent between the R/3 application server and its SAPGUI process and use this data in your program. The GUI Library component interacts with the R/3 SAPGUI and the R/3 application server, placing the screen data that is communicated between them in a special data structure. Your C program can then read or write data into or out of that data structure.

A C program using the GUI Library can use the interaction between an end user and SAPGUI screens, or it can replace the GUI screens with its own interface, while blocking the SAPGUI screens from being displayed.

Architecture

The Standard SAPGUI

On Windows systems, the R/3 SAPGUI interface consists primarily of two executable files: FRONT.EXE and SAPGUI.EXE. There is one SAPGUI process for each R/3 system connection. As the following diagram illustrates, SAPGUI.EXE starts one FRONT.EXE for each R/3 session.



The R/3 SAPGUI communicates with the R/3 application server through a terminal-like protocol.

The GUILib.DLL

The GUILibrary DLL interface provides a rich set of API calls that can be used to manipulate GUI screens in transactions and drive them programmatically. This API is rich enough that SAP has used it to develop GUI component and the macro recorder on top of it.

The GUILib.dll is a shared or dynamic link library that runs in the same memory space as the client process. Since it runs in the same memory space, the GUILib can pass information conveniently through the memory buffer.

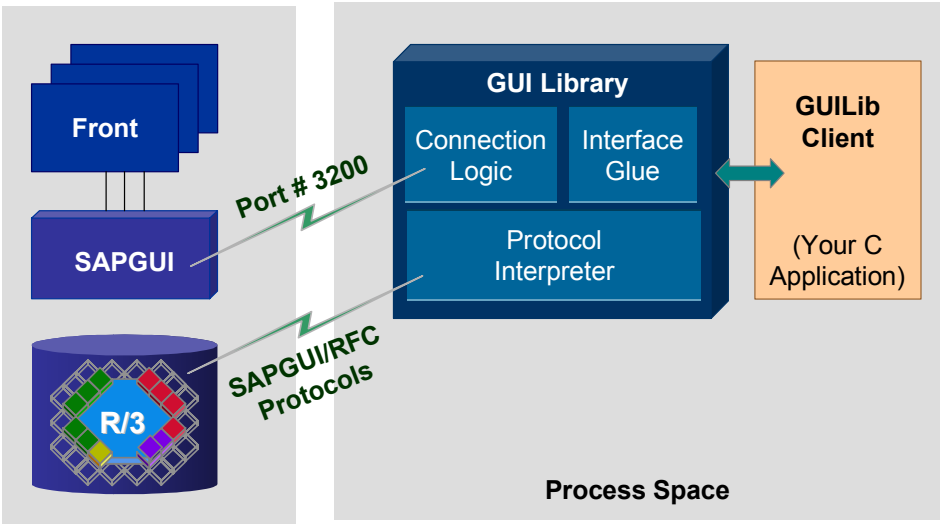
Internal Architecture of the GUI Library

SAP Automation GUI application programs can communicate with the R/3 application server by making GUI Library API calls. Internally, the GUI Library uses either the SAPGUI protocol or the RFC protocol as needed to communicate with the R/3 application server.

With the IT_StartSAPGui [Page 146] or the IT_NewConnection [Page 127] methods, the SAP Automation GUI application program can connect to SAPGUI. Internally, the GUI Library connects to the SAPGUI using Port number 3200.

The following diagram shows the internal parts of the GUI Library along with their role in the connection to R/3.

The GUILib.DLL



A Bit of History

The GUI Library product evolved from a product called Merlin. As a result, the word *merlin* still exists in various places in the GUI Library code.

The GUI Library product used to be called Intelligent Terminal. This is why many of the function and structure names in the GUI Library have the prefix *IT*.

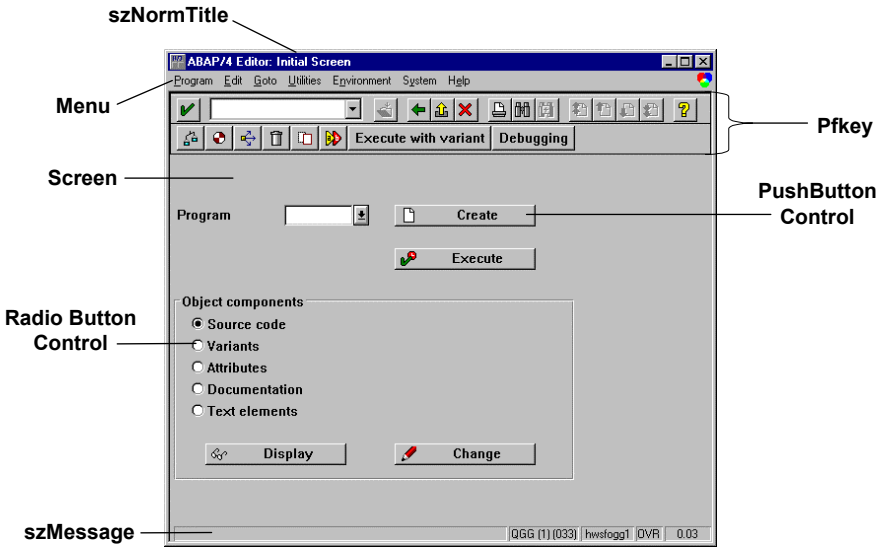
Using the GUI Library

The SAPGUI Screen and the GUI Library

The GUI Library divides the R/3 SAPGUI screen into the following major parts:

Part	Represents
Menus	The menu bar
PFKeys	The toolbar and its buttons
Screen	The area inside the screen with the various controls and fields

The following diagram shows the parts of the SAPGUI screen:



The Event Structure and the SAPGUI Screen

The GUI Library uses a structure, called the event structure [Page 36] (IT_EVENT), and several related structures to store the state of the SAPGUI screen and its data.

Your application communicates with

The *szNormTitle* in the diagram above is a member of the IT_EVENT structure containing the string of the title of the screen.

The *szMessage* member of the IT_EVENT structure contains the string of the message at the bottom of the R/3 SAPGUI screen.

The GUI Library Event Structure (IT_EVENT)

The most important structure in the GUI Library is the event structure (It_Event). It represents a single R/3 screen.

The GUI Library stores in it a DYNP (screen) event, which is a packet of information going to or from the R/3 application server. This could be a description of a dialog, a screen, messages, or menu information. The event structure encapsulates the union of all this information.

The client application can retrieve information from the event structure, or it can place information in that structure.

The event structure points to other GUI Library structures [Page 37] that contain more detailed information on the contents of the R/3 screen.

For detailed information on the event structure see the reference section for It_Event [Page 76].

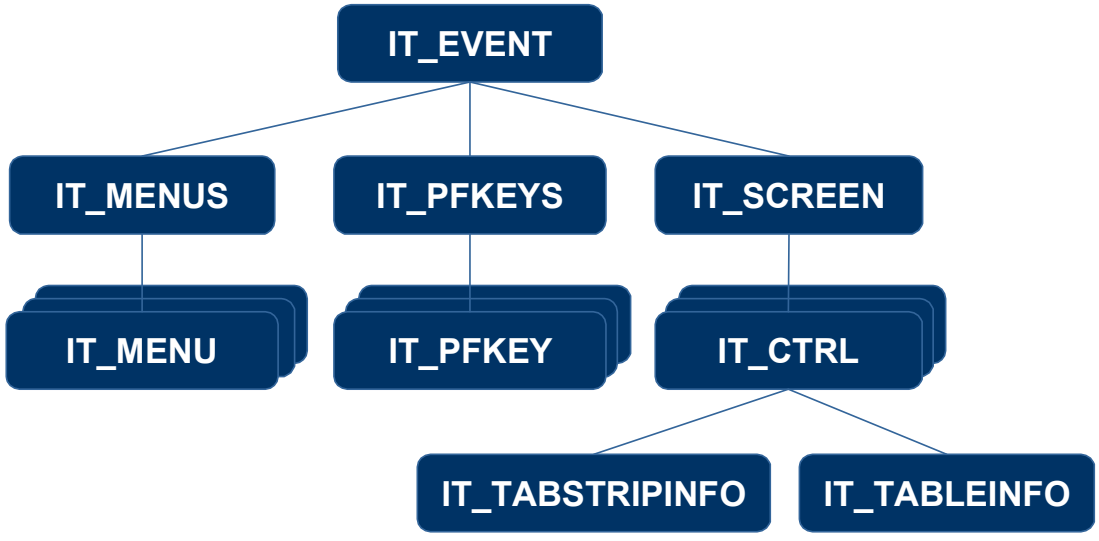
Other GUI Library Structures

The details of the different components of the screen associated with the reside in various structures related to the IT_EVENT structure.

The following table describes the information each of these additional structures contain:

Structure	Contains
IT_MENUS [Page 97]	A count of the menu items on the screen, and a pointer to the array of menu item structures
IT_MENU [Page 98]	Information on a single menu item
IT_PFKEYS [Page 94]	A count of the buttons on the toolbar, and a pointer to the array of button structures
IT_PFKEY [Page 95]	Information on a single button on the toolbar
IT_SCREEN [Page 84]	Information on the screen area, including cursor position in it, a count of the controls on the screen, and a pointer to an array of these controls
IT_CTRL [Page 87]	Information on a single control on the screen, such as the control's position and contents or state
IT_TABSTRIPINFO [Page 103]	Information on a tab strip, if exists on the screen
IT_TABLEINFO [Page 100]	Information on a table control, if exists on the screen

The following diagram shows the relationship between the GUI Library structures.



Working R/3 Transactions and Screens

Procedure

The following procedure summarizes the typical sequence of tasks you would perform when using the GUI Library to work with a certain application or transaction in R/3:

1. Connect and Log onto R/3 [Page 39].
2. Go to the desired R/3 transaction, by using one of the following methods:
   - Use the ItEv_SetOKCode [Page 171] to set the transaction code to the desired transaction, and then send the event to the application server with It_SendEvent [Page 135].
   - Use the It_GetTransaction [Page 120] function to go to the desired transaction (you do not need to use It_SendEvent when using the It_GetTransaction).
3. Get the data for the transaction screen from the R/3 application server, by using It_GetEvent [Page 114].
4. Let the user enter data into the transaction screen, or change the contents of the event structure programmatically, with the ItEv_* functions.
5. Send the new contents of the event structure to the R/3 application server with the It_SendEvent [Page 135] function.

You repeat the operations of sending the screen data to the R/3 application server with It_SendEvent and then capturing its response with the It_GetEvent as necessary.

See the sample program SAMPLE.C [Page 71] for an example of using these functions.

When working with R/3 transaction screens you can programmatically perform any task an end user can perform with the R/3 transaction screen. You can:

- Select a menu option [Page 49]
- Enter data into fields on the screen
- Select from a list or matchcodes [Page 53]
- Select an option from a radio button set [Page 54]
- Choose a button [Page 50]

Connecting to R/3 Application Server

Use

To use any functionality of the R/3 application server, you must first acquire a connection handle.

Procedure

1. Ask for a new connection to R/3, by using the It_NewConnection function [Page 127].

   This establishes a connection to the R/3 application server, returns a connection handle, and starts single SAP session.

   You can optionally specify that the standard SAPGUI is to be displayed. In this case, a SAPGUI and one Front are invoked.

2. Log onto the SAP system, by using the It_Login function [Page 125].

See the sample program SAMPLE.C [Page 71] for an example of using these functions.

You can handle up to six sessions in every connection. See the details in the topic Using Multiple Sessions [Page 40].

You can handle multiple connections by calling the connection functions multiple times. You may use up to 99 connections.

Result

The It_NewConnection [Page 127] returns this connection handle. The It_NewConnection opens a network connection to the application server and based on the flags passed to it, it starts SapGui-FRONT.

GUILib and SapGui-FRONT are connected via standard port number 3200, through which all information is passed back and forth.

A separate thread is created for each connection, in order to read data from the R/3 application server. When SapGui-FRONT is used, a separate thread is used to read SapGui-FRONT.

Handling Multiple Sessions

Use

You can use up to six sessions for every R/3 connection you have. The first session is always created when you establish a connection to the R/3 application server.

If SAPGUI Front is running you can only have one connection.

The following procedure shows how to handle multiple R/3 sessions.

Procedure

1. Ask for a new connection to R/3, by using the It_NewConnection function [Page 127].

   This establishes a connection to the R/3 application server, returns a connection handle, and starts the first SAP session.

   You can optionally specify that the standard SAPGUI is to be displayed. In this case, a SAPGUI and one Front are invoked.

2. Use It_Login [Page 125] to log the user or the program onto the R/3 system.

3. Register a callback function to watch for a new session being created, by using the It_SetNewSessionHook function [Page 145].

   The new session can be created as a result of an end user asking for it, by using the menu option *System→Create session*, for example.

   You can also create a new session programmatically in one of the following ways:

   - Using the ItEv_SetOKCode function [Page 171] with "/o" as a parameter
   - Using the ItEv_SetMenu function [Page 168] with "Create session" as a parameter

4. Once the R/3 application server creates a new session, regardless of the reason, you need to capture the handle to the new session into a variable.

   You do so within the callback handler function for the It_SetNewSessionHook function.

5. Register a callback function to watch for the deletion of any session, by using the It_SetDelSessionHook function [Page 143].

You can use the ItEv_GetSessionCount function [Page 163] at any time to check the number of sessions that are open.

Example

The following example handles two sessions.

The main program handles the connection to R/3 and handles the first session. In the first session, the program invokes a specific screen in a specific transaction (screen *410* in transaction *bibs*). The session handle is kept in the *hMr* variable.

The *newses* function gets called when R/3 gets a request for a new session. In the second session, the program invokes another R/3 transaction, namely transaction *se38*.

The handle of the second session is kept in the *hMr2* variable. If you wish to handle more than two sessions, use an array of variables, instead of the single variable *hMr2*.

```
#include <stdio.h>
#include "guilib.h"
#include <winbase.h>

static int newses(HANDLE ptr);

HANDLE  hMr2;
IT_EVENT *pEvt2;
int main()
{

      char *server = "myhost";
      char *system = "MYSYS";
      char *client = "005";

      char *user = "***";
      char *passwd = "***";
      char *lang = "e";

      HANDLE hMr;
      IT_EVENT *pEvt;

      hMr = It_NewConnection(server, system, SAPGUI_FRONT);

      if(hMr)
      {
            It_SetNewSessionHook((HANDLE)hMr, newses);
      }

      if (hMr == NULL) {
            exit(1);
      }

      It_GetEvent(hMr, &pEvt);

      It_Login(hMr, client, user, passwd, lang);
      It_GetEvent(hMr, &pEvt);

      ItEv_SetOKCode(pEvt, "bibs");
      It_SendEvent(hMr, &pEvt);
      It_GetEvent(hMr, &pEvt);

      ItEv_SetOKCode(pEvt, "410");
      It_SendEvent(hMr, &pEvt);

      while (1) {

            It_GetEventEx(hMr, &pEvt, GV_GETFRONTEVENT);

            if (pEvt->eventtype & EVT_SEND_FRONT_EVT_REQ) {
                  It_SendEvent(hMr, &pEvt);
```

```
            }

            int rc = It_GetEvent(hMr, &pEvt);
            if (!rc)
                  break;

            if (pEvt->eventtype & EVT_END_OF_SESSION) {
                  break;
            }

           Sleep (1000);
            if (hMr2)
            {
                  It_GetEvent(hMr2, &pEvt2);
                  ItEv_SetOKCode(pEvt2, "se38");
                  It_SendEvent(hMr2, &pEvt2);
                  It_GetEvent(hMr2, &pEvt2);
            }
      }

      return 0;
}

static int newses(HANDLE ptr)
{
      printf("new session <0x%lx>\n", ptr);
      hMr2 = (HANDLE) ptr;
      return TRUE;
}
```

Disconnecting from R/3

Procedure

If you are connected to R/3, follow this procedure to disconnect:

1. Check if SAPGUI Front is running, by checking the value of the EVT_FRONT_RUNNING bit in the eventtype member of IT_EVENT [Page 80].

   Note that your program may have started SAPGUI in one of two ways:

   - Using the SAPGUI_FRONT flag with any of the connection functions
   - Calling the It_StartSAPGUI function [Page 146] after logon

2. If you SAPGUI Front is running, then stop it by using the It_StopSapGui function [Page 147].
3. Use the It_Logoff function [Page 126] to disconnect from the R/3 server.
4. Free the connection handle, by using the It_FreeConnection function [Page 112].

Example

The following example shows a cleanup function that closes the SAPGUI Front it it is open, logs the user off and closes the connection.

```
//Parameters: hConn - A handle to the server connection
loginflag - A flag which tells whether login succeeded or not
void Cleanup(HANDLE hConn, int loginflag)
{
PIT_EVENT pEvt = 0;
if(hConn)
{
     if(It_GetEventEx((HANDLE)hConn, &pEvt, GV_ISSAPGUIRUNNING))
     {
          if (It_StopSapGui(hConn) == 0)
          {
               printf("It_StopSapGui failed!");
          }
     }
     if(loginflag)
          It_Logoff(hConn);
     It_FreeConnection(hConn);
}
}
```

Terminating the Connection to R/3

The client application initiates termination by calling It_Logoff [Page 126]. A subsequent call to It_GetEvent [Page 114] will return an event of type EVT_END_OF_SESSION. On receipt of this message, the client should not make any further SAP Automation GUI calls, as the subsystem has terminated.

Using the Event Structure (It_Event)

Use

The program using the GUI Library can:

- Get screen information as it is sent from the R/3 application server
- Send screen information back to the R/3 application server

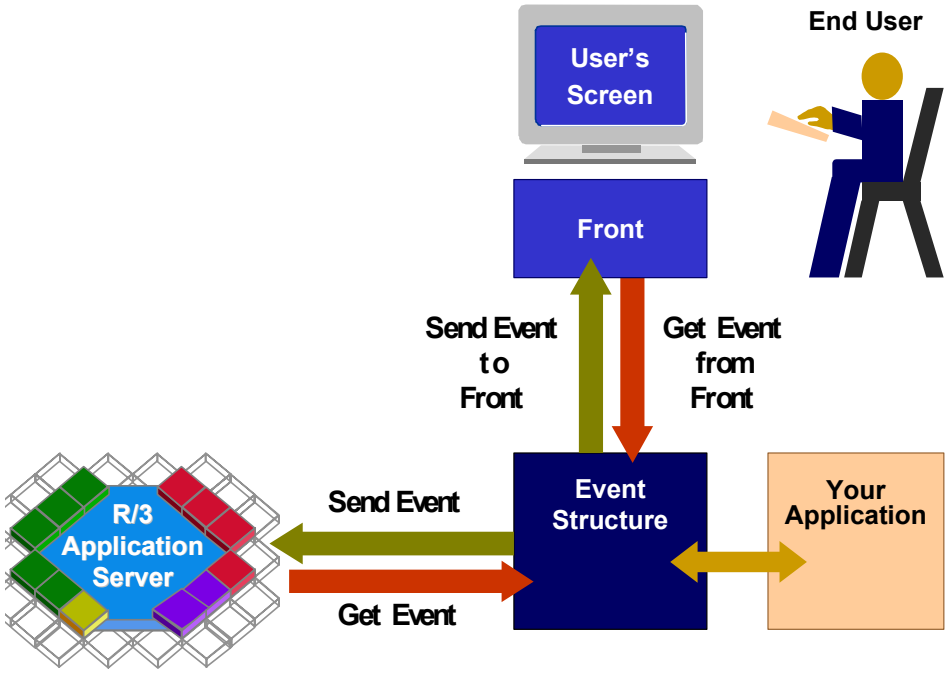
Your program does so by getting an event using the It_GetEvent (or It_GetEventEx) and sending an event using It_SendEvent (or It_SendEventEx) functions respectively, as illustrated in the following diagram.

If your application is controlling or recording the interaction of an end user with the standard SAPGUI screens, your program can also:

- Send screen data from the event structure to Front, which results in the display of this screen data to the end user
- Get the screen data after the user has changed it from Front into the event structure.

Your program uses the It_SendEvent and It_GetEventEx with special flags for sending or getting the data from Front (as opposed to from the R/3 application server)

The following illustration summarizes the flow of screen data between the R/3 application server, the event structure, and Front.



Prerequisite

You must establish a connection to R/3 [Page 39] before getting or sending data to R/3.

Procedure

1. The client application first reads an event by calling It_GetEvent [Page 114] or It_GetEventEx [Page 115]. These are blocking calls, meaning that they return only when the server has sent a response.

   The program calling It_GetEvent and It_GetEvent Ex waits until a DYNP event arrives. It_GetEvent and It_GetEvent Ex then return the result in a pointer to the It_Event structure.

   If any error occurs, then these functions return FALSE; otherwise they return TRUE.

2. Before sending an event back to the application server, the client application can alter the event structure either by direct manipulation of the event structure or by using the various ItEv_* functions.

3. The client would then call It_SendEvent [Page 135], to pass the It_Event structure.

   If no error occurs, It_SendEvent processes the It_Event structure and prepares a packet to be sent to the application server.

   It_SendEvent then frees up the It_Event and zeros out the It_Event pointer.

   After this point, the client application should not access the event structure, as this event is already sent and is no longer valid.

   To send a modified event to the server, you can also use:

   - It_SendReturn [Page 140]

- It_SendPFKeyID [Page 139]

4. After It_SendEvent is called, the client can call It_GetEvent to get the application server's response.

Making a Copy of the Event Structure

You can keep a copy of the event structure for later use or for off-line processing, by calling It_Dup [Page 110].

This function returns a copy of the event structure, which should not be used for communicating data to the application server, meaning that you should not use it with It_SendEvent.

When you are done with the copied structure, you should call It_FreeEvent [Page 113] to free the structure.

Peeking at the Event Structure

It_PeekEvent [Page 132] performs similar function to the It_GetEvent, only it allows you to see the contents of the same event several times before you actually get the event with an It_GetEvent* call.

This is especially useful if two procedures need to access the same event information.

In the example below, the program uses the It_PeekEvent as a way to handle possible errors. It uses the It_PeekEvent to check that the event returned no error. Only if no error occurred does it use It_GetEvent.

Since the It_PeekEvent fills the event structure, you need to use It_FreeEvent [Page 113] to free the event structure after using the It_PeekEvent.

Example

The flexibility of this architecture can be demonstrated by looking at the code of It_Login [Page 125], which is implemented using It_GetEvent and It_SendReturn [Page 140], (a variant of It_SendEvent).

```
DWORD DLEX It_Login (HANDLE hMerlin, char* client, char* name,
                     char* passwd, char* lang)
{
PIT_EVENT pEvt = 0;
if (It_GetEvent(hMerlin, &pEvt))
{
    ItEv_SetValue(pEvt, 1, client);
    ItEv_SetValue(pEvt, 3, name);
    ItEv_SetValue(pEvt, 5, passwd);
    ItEv_SetValue(pEvt, 7, lang);

    It_SendReturn(hMerlin, &pEvt);

    It_PeekEvent(hMerlin, &pEvt);
    if(pEvt->eventtype & EVT_MESSAGE)
{
It_FreeEvent(&pEvt);
return FALSE;
}
    // Bypass copyright screen
```

Using the Event Structure (It_Event)

```
    It_GetEvent(hMerlin, &pEvt);
    It_SendReturn(hMerlin, &pEvt);
}
return TRUE;
}
```

Using Menus

Menus can be directly accessed in the 3.1G version of the SAP Automation GUI software.

However, directly accessing a menu is usually not the most efficient method for accessing menu functionality, as menu traversal involves several round trips to the application server.

Several alternatives are available. These techniques are also required to access menu entries in SAP Automation GUI software prior to 3.1G.

Every menu choice is assigned to an OK code within the SAP Menu Painter software. Sending the OK code associated with the menu choice has the same affect as selecting the menu entry.

To determine the OK code for a menu entry within the current R/3 screen:

1. Get the program name and GUI status for the current screen from the System Status dialog box.
2. Go to the Menu Painter (transaction SE41).
3. Enter the program name and GUI status in the text entry boxes and choose the Display button.
4. The menu bar is displayed at the top of the screen. Double click on the menu bar entries and submenu entries until you see the menu entry you are looking for. The OK code is the 4-letter code in the “Func” column to the left of the menu entry.

The effect of many menu entries can be simulated without going to the Menu Painter. Menu entries that go to other transactions can be simulated using It_GetTransaction [Page 120] in the C API, or the Transaction [Page 246] method in the OLE Automation Server. These methods add the “/n” prefix before the transaction code argument and set the OK code to the resulting value.

Note that when an OK Code for another transaction is set in an event using the “/n” prefix, other event settings are generally ignored. If the current transaction needs to be finished before moving to the next transaction, first send one event to finish the current transaction. Then get the returned event and send a second event to set the OK Code for the next transaction.

Some menu paths duplicate function key entries. In this case, the appropriate function key can be sent in order to move to the appropriate screen. Function keys also have OK codes associated with them, so sending the function key has the same effect as manually sending the OK code.

Choosing Buttons on Toolbars

Toolbar buttons in R/3 transactions are associated with keys, which are included in the SAP Automation GUI keys collection.

With the R/3 3.0 GUI, these keys are usually listed in the quick info (that is, the ToolTip) for each toolbar button. Comparing the ToolTip or the toolbar button name with the list of keys available for an event will usually lead to the key to use to access a certain button.

See the topic Windows Virtual Key Values [Page 51] for listing of the key values.

Windows Virtual Key Values

The key names displayed in ToolTips are interpreted relative to F1 key. Windows represents the F1 key as 112 decimal (70 hex) and uses the constant VK_F1.

The following table lists the various types of keys, their equivalent constant and numeric values.

Key(s)	Constant	Numeric Representation
F1	VK_F1	112 decimal (70 hex)
F1 to F12	VK_F1 through VK_F12	112 through 123 decimal
Shift-F1 through Shift-F12	VK_F13 through VK_F24	124 through 135 decimal (The “Shift-” corresponds to adding 12 to the non-shifted function key value.)
Ctrl-25 and up (SAP’s “F0” key, also VK_DIVIDE)	Most of these keys do not have VK constants assigned	Decimal values 136 and up (Add the number of the “Ctrl-” key to 111 decimal to get the appropriate value. For example, Ctrl-30 is 141 decimal (111 + 30).)

Using Controls

Using Matchcodes

When using matchcodes and other lists, the number of entries sent by the application server to the presentation system is usually more than the number of entries shown within the window.

To extract all the entries at once from a matchcode list send the Next Page or Page Down keys (virtual keys VK_F23 or VK_NEXT) until the end of the list is reached.

When the first data element in the new event is the same as the first data element in the old event, this generally indicates that the old event contained the last page in the list.

The standard matchcode dialog box display the data to be entered in the field using color number 4.

When there are multiple types of matchcodes and you know the type you want, you can move directly to that matchcode selection without the intermediate dialog steps. To do this, enter an equal sign, the single-character matchcode type, and a period into the input field, then send the default return key. For instance, to get to matchcode type A, enter the following in the input field.

```
=a.
```

Selecting a Radio Button

When a radio button is selected programmatically, you generally also need to deselect the previously selected button in the same group. Only one radio button per group should remain selected.

Radio buttons are in the same group if they have the same area ID, block ID, and group ID.

Handling a Tab Strip Control

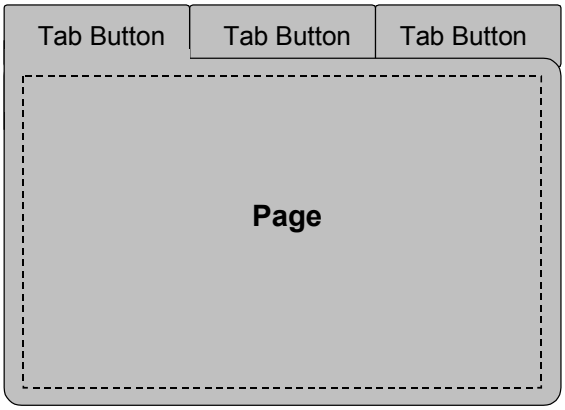
A tab strip control is a control that allows you to display different controls and fields on different pages. The tab strip contains buttons which determine which page is active.

The tab strip control is described in the the control structure (IT_CTRL) [Page 87]. More specific information on the tab strip control is in IT_TABSTRIPINFO [Page 103].

Tab Strip Control Parts

A tab strip control in SAPGUI has the following parts:

- The tab button, which is identified by its name
- The page, which is the area inside the tab strip, that is, the area containing the controls that are displayed when the user chooses the tab



Page Types

There are two types of pages in SAPGUI:

Page Type	Description
Server page	When the user chooses the tab for a server page, the application gets the data from the R/3 application server.
Local page	When the user chooses the tab for a local page, the application gets the data from the local cache (because it already had gotten this data from the server.)

The R/3 application defines which pages are local and which pages are server pages.

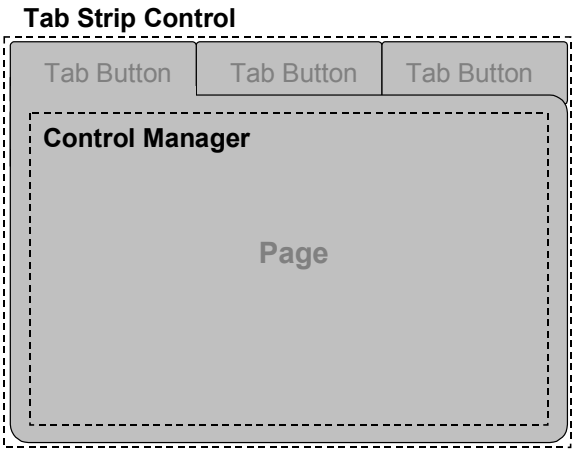
Tab Strip Control Hierarchy

The tab strip control is the parent control of the tabs in the tab strip. It is also the parent of a Control Manager (control of dlgtype [Page 90] CTRL_MANAGER).

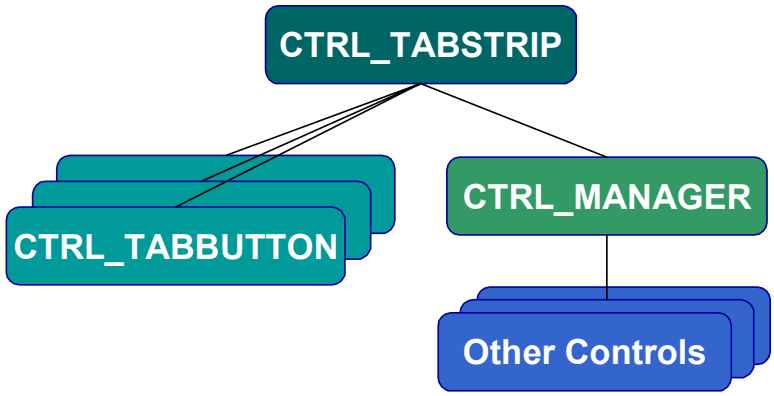
The Control Manager in turn is the parent of all the controls on the tab strip page. For example, if the tab strip page contains two frames, these two frames are the children of the Control Manager. The Control Manager is not visible on the screen. It merely acts as a container for the other controls on the tab strip control page.

Handling a Tab Strip Control

The following diagram shows the controls on a tab strip control.



The following diagram shows the hierachy of the controls in a tab strip, that is, it shows the parent-child relationship of these controls.



The information on the various controls which are a part of the tab strip is in the control structure (IT_CTRL) [Page 87].

Using Tab Strip Controls

general information on a tab strip control (as on any other control) is in the control structure. Additional information that is more specific to the tab strip, such as how many pages are in the tab strip, is in the It_TabStripInfo structure [Page 103]. To obtain this information you can use the ItEv_GetControlInfo function [Page 161].

Switching to a Particular Tab

1. Use ItEv_SetTabButton [Page 177] to specify which tab to switch to, that is, to specify which tab should become active. You can provide either the name, the value, or the index of the tab control.
2. Use It_SendEvent [Page 135] to send the tab-switching event to the application server.

   This sends the contents of the event structure to the server.

Example

The following code shows an example of switching to a particular tab, using its index. The program obtains the tab index with the ***ITCTRL_IDX*** macro.

```
ItEv_SetTabButton(pEvt, ITCTRL_IDX(8);
It_SendEvent(hMr, &pEvt);
```

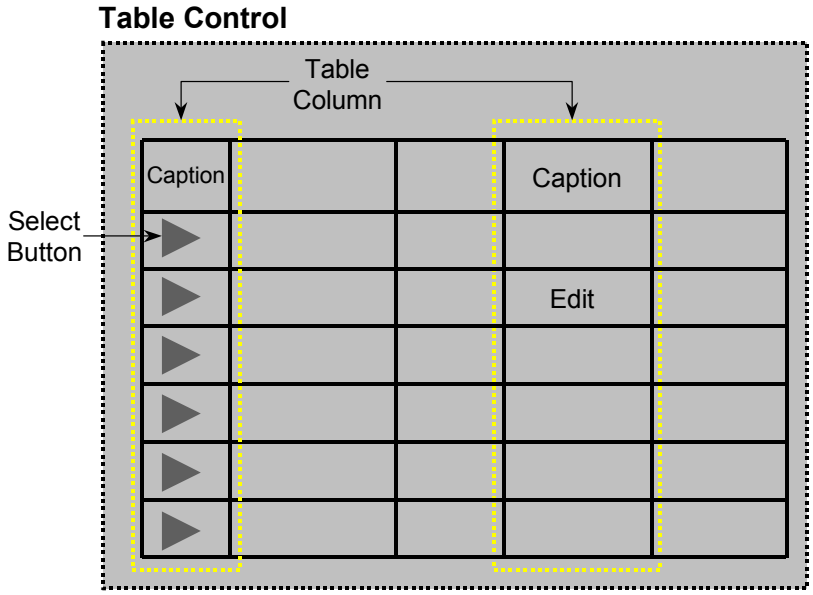
Handling a Table Control

A table control is a control that contains an R/3 table data. It has columns and rows of data.

The table control is described in the the control structure (IT_CTRL) [Page 87]. More specific information on the table control is in the IT_TABLEINFO structure [Page 100].

The Controls in a Table Control

The following diagram shows the parts of a table control.



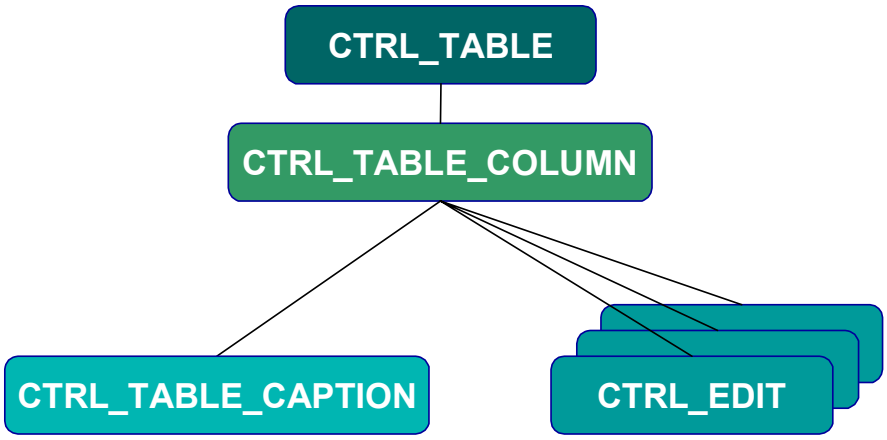
Controls Hierarchy of a Table Control

The Table control (control type [Page 90] CTRL_TABLE) is the parent of all the controls in the table.

The Table control is the parent control for all the Table Column controls (CTRL_TABLE_COLUMN).

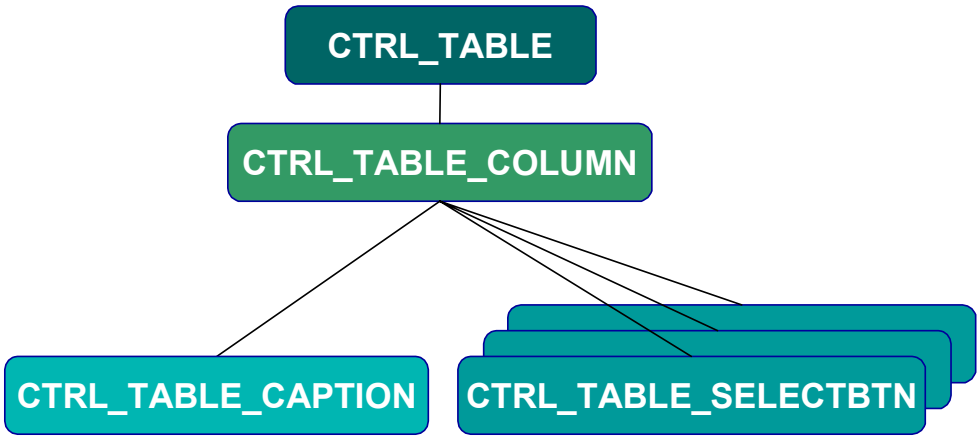
Each Table Column control is the parent of the Column Caption control (CTRL_TABLE_CAPTION), which is the title of the column. In addition, the Table Column control is the parent of all the cells of data in the column, which are of CTRL_EDIT type.

The following diagram summarizes this relationship.



When the table contains selection buttons (which allow you to select a specific row in the table), the column of the selection buttons also constitutes a Table Column control. In this case the Table Column control is the parent of a caption (CTRL_TABLE_CAPTION), which has no text, and it is also the parent of all the row selection buttons (CTRL_TABLE_SELECTBTN).

The following diagram summarizes this relationship.



The information on the various controls which are a part of the table is in the control structure (IT_CTRL) [Page 87].

Using Table Controls in SAPGUI

The following operations are allowed on a table control in SAPGUI:

- Scrolling up and down to view different rows in the table
- Scrolling left and right to view more columns in the table
- A column may be defined as fixed, and then the end user cannot scroll over this column.
- Switching the location of two columns in the table, using drag and drop

- Changing the width of a column

Using Table Controls in the GUI Library

The GUI Library provides all the above operations on a table control.

Displaying a Table Control

You display a table control as you would any other control on the screen.

Working with a Table Control

The following steps describe a typical procedure for working with the table control:

1. Use ItEv_GetControlInfo [Page 161], with the table control as a parameter, to get the extended information on the table control, such as the number of rows and column in it, or the number of fixed columns it contains. This fills the It_TableInfo structure [Page 100] with the extended table control information.
2. Change the information in the It_TableInfo structure, if needed. For example, the TabVerScrollbarStartRow determines the first row that is displayed on the screen. To scroll to a particular row, for example, change the value of this parameter to be the number of the row to be displayed as the first row after scrolling.
3. Use ItEv_SetControlInfo [Page 165] to apply the changes to the It_TableInfo structure.
4. Send the event to the server with It_SendEvent [Page 135].

   This will send the changes in the It_TableInfo structure to the application server.
5. Use It_GetEvent [Page 114] if you wish to refresh the screen with new table data.

Example: Scrolling

The following example scrolls in a table control to display row number 5 as the first row. It assumes that control number 16 on the screen is a table.

```
pEvt->eventtype |= MES_VSCROLL ;

IT_TABLEINFO tbinfo;
int   len = ItEv_GetControlInfo(pEvt, ITCTRL_IDX(16),&tbinfo,
sizeof(tbinfo));
tbinfo.TabVerScrollbarStartRow = 5;
len = ItEv_SetControlInfo(pEvt, ITCTRL_IDX(16),&tbinfo,
sizeof(tbinfo));
if (It_SendEvent(hMr, &pEvt) == FALSE) {
      printf("error in sendevent\n");
}

if (It_GetEvent(hMr, &pEvt) == FALSE) {

      printf("error in getevent\n");

}
```

Switching Between Columns

Use the ItEv_SetTableColumnPermutation function [Page 174] to switch between two columns in the table control.

The ItEv_SetTableColumnPermutation function uses an array that stores the order of the columns in the table control. Changing the values in the array changes the order of the columns in the table control.

The following example prompts the user to enter the columns to switch, and then switches between them. It assumes that control number 16 on the screen is a table.

```
printf("Swap: ");
int a, b, pCols[10];
for (i = 0; i < 10; i++)
      pCols[i] = i;
scanf("%d %d", &a, &b);
pCols[a] = b;
pCols[b] = a;
ItEv_SetTableColumnPermutation(pEvt, ITCTRL_IDX(16), pCols, 10);
It_SendEvent(hMr, &pEvt);
It_GetEvent(hMr, &pEvt);
```

Changing the Width of a Column

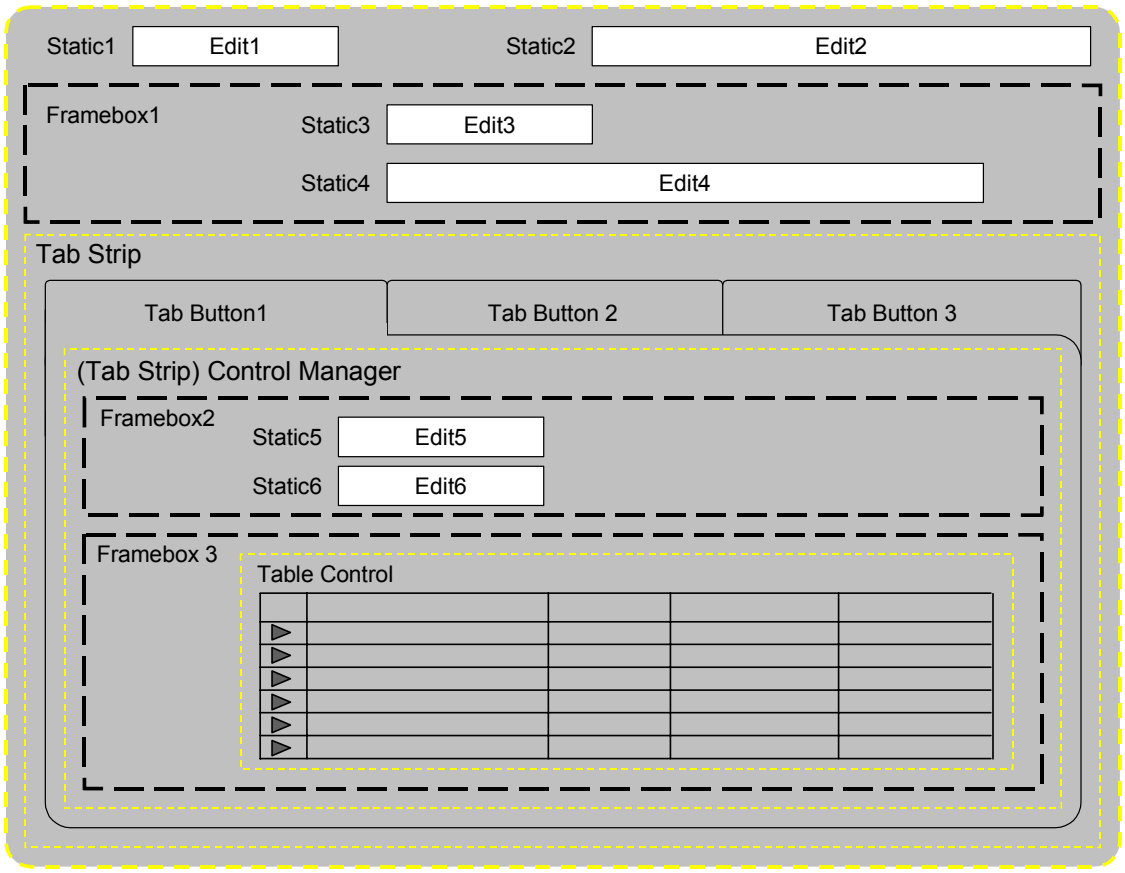
Use the ItEv_SetWidth function [Page 176] to change the width of a table control column.

Control Hierarchy

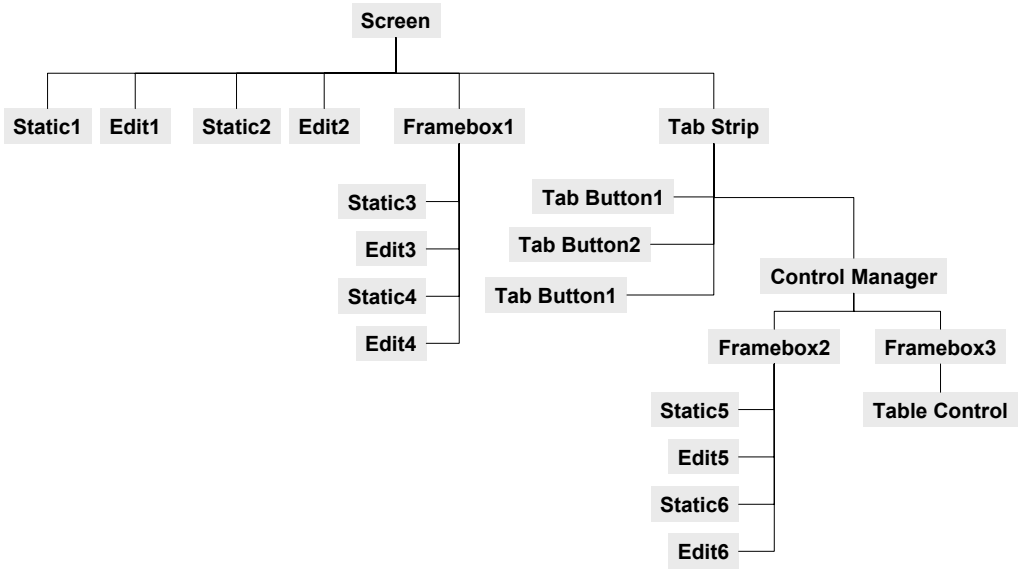
Some controls on the screen may contain other controls. For example, a framebox control (control type [Page 90] CTRL_FRAMEBOX) is a container for other controls. A tab strip control page contains other controls, and therefore the tab strip control manager, which represents the tab strip page, is a container for other controls.

Since you specify the coordinates of a control [Page 64] relative to its parent control, it is important to know which control is the parent of the control you are working with.

The following illustration shows a possible arrangement of controls on a screen. In yellow are controls that are not visible on the screen.



The following diagram shows the parent-child hierarchy of the controls in the above screen example.



For more details on the controls within a table control [Page 58] and within a tab strip control [Page 55], see the relevant topic.

Specifying Coordinates of Controls

Default Coordinates are Relative to the Parent Control

The coordinates of controls on the screen are specified as their zero-based location in terms of row and column numbers. For example, the coordinates of a control at the leftmost column and first row of the screen is 0,0.

By default, the coordinates of a control on the screen are relative to the coordinates of the parent of the control, if the control has a parent.

For example, if the control is a label (static) inside a frame, the frame control is the parent of the static control. Its coordinates are expressed relative to the frame, regardless of where the frame is on the screen. The top-left corner of the frame is considered 0,0. If the static control is the first top-most and left-most control within the frame its coordinates are probably not 0,0, but rather, 1,1, because there is probably some space between the frame and the label.

If a control does not have parents, its coordinates are relative to the screen.

Note, however that some controls are invisible on the screen, yet they can act as parents to other controls. For example, a tab strip control is an invisible control that acts as the parent of the tabs and the page area. Since it is invisible, the tabs can start at the 0,0 coordinates relative to the tab strip control.

See the topic Control Hierarchy [Page 62] for an example arrangement of controls on the screen and their parent-child relationship. Also see the discussion of the control hierarchies within a tab strip control [Page 55] and within a table control [Page 58].

Changing to Absolute Coordinates

You can use the SAPGUI_ABSOLUTE_COORD flag of the connection functions [Page 128] to specify that the coordinates should be expressed in absolute terms during the connection. This means that coordinates are always specified relative to the screen.

You can also use the SAPGUI_45A_COORD flag of the connection functions [Page 128] to specify that the coordinates should be expressed as a mixture of absolute and relative location, keeping the behavior of 4.5A GUI Library in this respect intact.

If you use these flags, they apply to all coordinate specifications during the connection.

Finding Controls

The GUI Library provides several functions to help you find controls on the screen. The following table describes these functions.

Function	Finds a Control Based on…
ItEv_FindControl [Page 154]	Its name, index, or value
ItEv_FindControlEx [Page 155]	Its name, index, or value, as in ItEv_FindControl. In addition, you can specify a starting point and a skip value, that is, the number of controls to skip.
ItEv_FindControlByPos [Page 156]	Location on the screen in character coordinates
ItEv_FindPFKeyID [Page 157]	Name of the key

Callback Functions and Macro Recording

The SAP Automation GUI Library provides support for a hook function to be called whenever a function from a selected list of GUILib.DLL functions is called.

A client using this functionality can then monitor the interactions of the GUI Library client, either an interactive user or a driver program driving through GUI Library.

As operations are being applied to the It_Event structure [Page 36], these callbacks enable you to generate code or macro-record the interactions.

Only changes in It_Event can be detected.

Example

The following code is the definition of the hook function in the header file:

```
typedef DWORD (CALLBACK *LPFNLOGCALLBACK) (HANDLE hMr,
        PPIT_EVENT ppEvt, long fnc_called, long changed,
        LPARAM idx, LPARAM word02);
```

You need to register the callback in your program once:

```
DWORD DLEX It_RegisterCallback(HANDLE hMr,
                               LPFNLOGCALLBACK lpfncallback);
```

Your *lpfncallback* hook function is called for every ItEv_* call.

When called, the parameters to the callback function will provide all the details of what was changed in the It_Event structure.

fnc_called	GUILib function that was called
changed	A bit field of components in the It_Event structure that were changed
idx	Index of control or PFKey that is affected (depends on fnc_called)
word02	Additional information

Listing Screen and Control Information

Use

There are two functions for creating a report that lists information about the screen that is associated with the current event:

Function	Description
ItEv_DumpEvent [Page 149]	Lists information on all the items in the event. This includes information on the screen, its menu, its toolbar buttons, and all the controls on the screen
It_ListControls [Page 124]	Lists information on all the controls that are on the screen, which is associated with the event.

Note that both functions list the same information on the controls that are on the screen. However, the It_ListControl function lists a subset of the information that the ItEv_DumpEvent lists.

For the controls on the screen the two functions list five attributes of each of the controls:

- The id of the screen (the *iModal* member of the IT_SCREEN structure [Page 84])
- The *name*, *value*, *left*, and *top* members of the IT_CTRL structure [Page 87]

Both of these functions allow you to either print the information or to display the information on the standard output device (most likely the screen).

Procedure

1. Write a C function to specify the format of the report for listing the screen or the control information.
2. Call It_SetDumpHook [Page 144] to point to this C function (the report format specification function).
3. Call the It_ListControls [Page 124] or the ItEv_DumpEvent [Page 149] function to create the report.

Example

The following is a sample print function to be used later by the It_ListControls function:

```
int putt(char *ptr)
{
return printf("%s", ptr);
}
```

The following example prints the information on the controls:

```
int putt(char*);

/* provide the print function to be used when It_ListControls is called */
if (It_SetDumpHook(hndl, putt) == FALSE)
      {
      printf(" SetDumpHook Error while SetDumpHook!-- It_SetDumpHook\n");
      // Handle error case;
      }
```

Listing Screen and Control Information

```
/* print out the list of controls. This calls putt to print the controls */
if (It_ListControls(hndl) == FALSE)
      {
      printf(" List control error. Program aborted!\n");
      // Handle error case;
      }
```

Using Screens with ActiveX Controls

Use

The GUI Library offers a limited support for ActiveX controls on the SAP GUI screen.

If any of the controls on an R/3 application screen with which are working is an ActiveX control, you should use the SAPGUI_ACTIVEX flag of the connection functions [Page 128]. Using this flag allows screens with ActiveX controls to be displayed correctly to the end user. It also allows the R/3 application server to receive the correct information when the end user uses any of the ActiveX controls on such screens.

Note that you must use the SAPGUI_ACTIVEX flag consistently in one application: if you use it in one call to a connection function, you must use it in all connection function calls in your application.

Once you use this flag in your application, however, any ActiveX control data is communicated between the R/3 application server and the SAP GUI Front directly, bypassing the GUI Library (and as a result, bypassing the GUI Component). The data of other SAP GUI controls is still communicated to the GUI Library.

Notice that this may result in incomplete data in the event structure (and its related structures): the data for regular SAP GUI controls is included in the event structure, while ActiveX control data is not included in the event structure.

Because ActiveX control information is missing from the event structure, the data in the event structure may be sometimes inconsistent with the data that the R/3 application server and the GUI Front have.

As a result, an application using this flag can only get events from R/3 or from Front, but it cannot send events to R/3 or to Front.

Therefore, you should use this flag with caution, and only when necessary, that is only for R/3 applications that contain any ActiveX controls.

Requirements and Restrictions

- The SAP GUI Front must be launched when using this flag. The SAPGUI_ACTIVEX connection flag is ignored if SAP GUI Front is not running.
- This feature only works with SAP GUI release 4.6A and higher. The SAPGUI_ACTIVEX connection flag is ignored if you are using an earlier release of the SAP GUI.
- Logon and logoff cannot be performed through the application: you or an end user must log on interactively, using the SAP GUI Front dialog.

Procedure

To work with screens that contain any ActiveX controls on them:

1. Call the desired connection function using both the SAPGUI_FRONT and the SAPGUI_ACTIVEX connection flags [Page 128].

   The SAPGUI_FRONT flag starts up the GUI Front when the connection establishes.

   The SAPGUI_ACTIVEX flag allows for screens with ActiveX controls information to pass between the R/3 application server and the SAP GUI Front (screens).

2. You or the end user of your application must log onto the R/3 system manually, that is, using the SAP Logon dialog interactively.
3. You can capture user actions at the SAP GUI screens, by getting the various events. Note, that data on any ActiveX controls on the screen is not captured. Only data of other controls is available to you through the event and related structures.

   You cannot send any of the events to R/3 or to Front.
4. You or the end user must manually log off the session.

Sample Program: SAMPLE.C

This section describes the sample code provided with the GUILib.

Compile this program with the GUILib header file, link it with guilib.lib. Also make sure that the GUILib DLL is in the same directory as sample.exe or in the system directory.

The sample demonstrates how to write a program using the SAP GUILib library and how to implement functions that communicate with the R/3 System.

The program results in listing of the current users in a given R/3 system.

The program prompts the user to enter R/3 application server information and user information.

Analysis of the Sample Program

- Connect with R/3 application server:

  ```
  hHandle = It_NewConnection(Hostname, SystemID, SAPGUI_FRONT);
  ```

  This function establishes a connection between the R/3 application server and GUILib. Flag is set as SAPGUI_FRONT to start SAPGUI with a connection to the R3 system at startup. FRONT will display screen information on SAPGUI while your application moves along. This function returns a non-zero handle to a SAP Automation GUILib session if successful. If the returned handle is zero, the connection to R/3 failed.

- Log onto R/3 system:

  ```
  if ((It_Login (hHandle, Client, User, Passwd, Language))==0)
  {
              printf ( "\n Failed to log on\n");
              It_StopSapGui(hHandle);
              It_FreeConnection(hHandle);
              return ;
  } else
              printf (" Successful Login as %s on %s\n",User,
                                                  Hostname);
  ```

  This function logs in to an SAP system with a standard login screen using the specified client, user name, password, and language. If the login fails, the program should call It_StopSapGui to stop the SAP FRONT, then call It_FreeConnection to close the connection with R/3 system.

- Go to transaction se38:

  ```
  It_GetTransaction(hHandle, "se38")
  ```

  It_GetTransaction invokes the specified transaction. The second parameter in this function is a pointer to a transaction code, such as transaction “se38”.

- Make pEvt point to a properly structured IT_EVENT structure:

  ```
  It_GetEvent(hHandle, &pEvt)
  ```

  pEvt must be initialized to zero before calling this function.

- Print the list of controls on the screen:

  ```
  It_SetDumpHook(hHandle, putt)
  ```

This function provides the printing function to be used when It_ListControls is called.

The second parameter in this function is a pointer to the printing function ***putt.*** Putt is defined in the sample program that provides the standard printing function.

- Print a list of control information:

  It_ListControls(hHandle)

  This function prints a list of control information using the printing function specified by It_SetDumpHook.

- Determine if the control exists

```
const char* PROGRAM_FIELD = "RS38M-PROGRAMM" ;
      CtrlIndex = ItEv_FindControl(pEvt, PROGRAM_FIELD,
                        FC_FIND_TYPE(CTRL_MATCH)||FC_FIND_FIELD);
      if (CtrlIndex < 0)
      {
            printf(" Failed to find a CTRL_MATCH control named \
                  RS38M-PROGRAMM!\n");
            return ;
      }
```

  This function looks up the first control with RS38M-PROGRAMM as the field name and with CTRL_MATCH as the type of control. The second parameter is a pointer to the control. By default, it accepts the field name of the control. The third parameter is a flag to control the search. In this example, we search the first control by field name and type of control, with the flag set as FC_FIND_TYPE(CTRL_MATCH) || FC_FIND_FIELD). If the control is found, a 0-based index of control will be returned. Otherwise, return –1.

- Set the program's name (RSM04000):

  ItEv_SetValue (pEvt, ITCTRL_IDX(CtrlIndex), PROGRAM)

  This function fills the program name into the matchcode field, just as in R/3 system, you would enter the program name in se38.

- Simulate the 'Execute' key to execute the above program:

  ItEv_SetCurPosByCtrl(pEvt,ITCTRL_IDX(3))

  It_SendReturn(hHandle, &pEvt)

  Those two functions simulate pressing the "Execute" button on SAPGUI. ItEv_SetCurPosByCtrl is used to position the cursor on a pushbutton. It_SendReturn has the same effect as pushing the toolbar button "Execute" when using the SAPGUI.

- Print out the controls information on the screen, which displays the current users in specified R/3 system:

```
It_GetEvent(hHandle, &pEvt)
      /* print out the title menu on the screen */
      for (k = 2; k < 17 ; k++)
            printf("%s ", pEvt->screen.pCtrl[k].value);
      /* print out the content of login informaton */
      for (i = 1; i < (int) pEvt->screen.iCtrlCnt/15; i++)
      {
            for (j = i * 15+4; j < (i+1)*15 + 4; j += 2)
                  printf("%s | ", pEvt->screen.pCtrl[j].value);
```

```
        printf("\n");
    }
    printf("\n  %s\n",
       pEvt->screen.pCtrl[pEvt->screen.iCtrlCnt-3].value);
```

After executing the program, a user screen will be displayed. Each displayed field is a control. This part of the code formats the values of those controls and prints them out at the console.

- Logoff from R/3 and disconnect from the R/3 application server:

```
It_StopSapGui(hHandle)
It_Logoff(hHandle)
It_FreeConnection(hHandle)
```

The normal logoff consists of three steps: stop SAPGUI FRONT, logoff from R/3 as a user and disconnect from the R/3 system. It_FreeConnection includes freeing the allocated resources.

GUI Library Reference

This section describes each of the data structures and function calls in the SAP Automation GUILib, a C-language application-programming interface.

Data Structures

The data structures used by the GUI Library collectively hold the information on a single SAPGUI screen.

The following data structures are available:

IT_CTRL [Page 87]	IT_EVENT [Page 76]	IT_MENU [Page 98]
IT_MENUS [Page 97]	IT_PFKEY [Page 95]	IT_PFKEYS [Page 94]
IT_SCREEN [Page 84]	IT_TABLEINFO [Page 100]	IT_TABSTRIPINFO [Page 103]

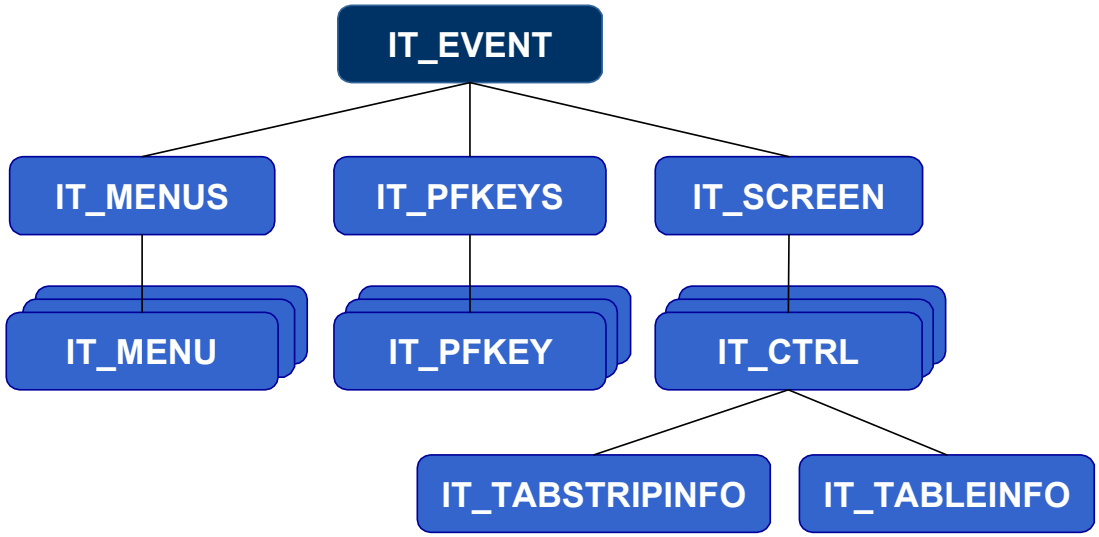
The IT_EVENT structure is the main structure that describes an event.

IT_EVENT

Use

IT_EVENT is the event structure, which represents a single R/3 screen.

The event structure contains some of the overall information of the screen, while the more detailed information resides in the related structures. The IT_EVENT structure contains information about the screen as a whole. For example it contains the title of the screen, the text of the message at the bottom of the screen as well as details about the scroll bar of the screen. It then points to the IT_MENUS [Page 97], IT_PFKEYS [Page 94], and IT_SCREEN [Page 84] structures, which contain more information on the event: the IT_MENUS holds information on the menu bar, the IT_PFKEYS details the buttons on the toolbar, and the IT_SCREEN contains more information about the user area of the screen, on which the various controls reside.



You can read or change the contents of the event structure and its related structure in one of the following methods:

- Directly through your C code
- By calling one of the ItEv* functions

Syntax

```
typedef struct IT_EVENT_t {
 long cbSize;
 long Version;
 long eventtype;
 HANDLE hMerlin;       // Handle to IT session
 IT_SCREEN screen;
 IT_PFKEYS pfkeys;
 IT_MENUS menus;
```

```
  union {
    int key;            // send key or
    HANDLE hMenu;       // send menu or
    long lPos;          // send scroll message
    };

  /* EVT_OKCODE */
  char okcode[MAX_OK];

  /* EVT_MESSAGE */
  char szMessage[MAX_MESSAGE];

  /* Static information */
  unsigned short  nDiagVersion;
  char szDB[128];
  char szCPU[128];
  unsigned short  nModeNumber;
  char szTCode[64];
  char szUsername[64];
  char szClient[64];

  /* EVT_TITLE */
  char szNormTitle[MAX_TITLE];

  /* EVT_SESSION_ADDED | EVT_SESSION_REMOVED */
  int iSessionId;

  } IT_EVENT;
typedef IT_EVENT *PIT_EVENT;
typedef PIT_EVENT *PPIT_EVENT;
#define IT_EVENT_SIZE        (sizeof(IT_EVENT))
```

Members

cbSize	Specifies the size of the whole event structure in bytes.
Version	Specifies the R/3 release (version) number of the system from which the event structure is taken
eventtype [Page 80]	A bit mask that indicates which type of information exists in the event structure and all of its related structures. After getting an event from R/3 you can use this field for determining which fields are valid in the event structure. Before sending an event to the R/3 application server your application must set the relevant bits in this mask. See the detailed description of eventtype and its bits by following the link.
hMerlin	Handle to the SAP Automation GUI session.
screen	The structure containing the detailed information of the screen. See IT_SCREEN [Page 84] structure.
pfkeys	The structure containing the detailed information of the toolbar and its buttons. See IT_PFKEYS [Page 94] structure.

IT_EVENT

menus	The structure containing the detailed information of the menu bar. See the IT_MENUS [Page 97] structure.
key	The key to send (See the topic Windows Virtual Key Values [Page 51] for listing of the key values). Filled in by client. Your program must set the MES_KEY bit of eventtype [Page 80] to on to indicate that you are sending a key.
hMenu	Which menu option was requested. Filled in by the client. Note that to send the menu option request, you must turn the MES_MENU bit of eventtype [Page 80] on.
lPos	Position to scroll to on the screen. This is used for either horizontal or vertical scrolling. It specifies the row to appear at the top of the screen after a vertical scroll, or the first column on the screen when scrolling horizontally. To show the first row at the top of the screen or to show the first column as the beginning of the screen use 1 for lPos. Works in conjunction with the MES_HSCROLL or MES_VSCROLL of eventtype [Page 80].
okcode	OK code is what you enter in the command field in the SAPGUI screen. Most of the time it is a transaction code. Filled by the client.
szMessage	Error, warning, or informational message. Valid (that is, you can only check its value) if the EVT_MESSAGE bit in eventtype is on.
nDiagVersion	DIAG is an internal protocol used for the communication between the application server and Front. It has different versions, each of which may include different features. nDiagVersion indicates the version of DIAG. This is static information, which is only available if the EVT_STATIC_INFO bit of eventtype [Page 80] is on.
szDB	Name of database. (the Name field in the Database data group on the System: Status screen in SAPGUI. This is static information, which is only available if the EVT_STATIC_INFO bit of eventtype [Page 80] is on.
szCPU	Name of CPU. This is equivalent to the Server name field in the Host data group on the System: Status screen in SAPGUI. This is static information, which is only available if the EVT_STATIC_INFO bit of eventtype [Page 80] is on.
nModeNumber	Mode number, which is the session number. This is static information, which is only available if the EVT_STATIC_INFO bit of eventtype [Page 80] is on.
szTCode	Reserved
szUsername	Username for this session. This is static information, which is only available if the EVT_STATIC_INFO bit of eventtype [Page 80] is on.

szClient	Client for this session. This is static information, which is only available if the EVT_STATIC_INFO bit of eventtype [Page 80] is on.
szNormTitle	Title of this screen.
iSessionId	Reserved

See also

IT_SCREEN [Page 84], IT_PFKEYS [Page 94], IT_MENUS [Page 97].

Eventtype Member of IT_EVENT

Eventtype is a set of bits creating a mask that indicates which type of information exists in the event structure (IT_EVENT [Page 76]) and all of its related structures.

Using the Eventtype Bits

The GUI Library uses the eventtype data to communicate the type of information that exists in the event structure between your program and the R/3 application server.

Some of the bits in eventtype are set by your application, while others are set by the GUI Library as a result of the information coming in from R/3.

The values of the bits in eventtype play an important role when you either get or send the contents of the event structure to or from the R/3 application server.

Bits Set up by Your Application

Your application sets some of the bits in eventtype before sending an event to the application server with either the It_SendEvent or the It_SendEventEx functions.

These bits indicate to the application server which type of information exists in the event structure you are sending to it. When you send certain types of information from the event structure to R/3, you must set the relevant eventtype bits.

Eventtype MES_* Values

The following table lists the eventtype bits that are set by your application before sending an event. The table describes when you should set each of the bits:

Bit	Turn on to Indicate that the Event You Are Sending Contains…
MES_KEY	A PFKey
MES_OKCODE	An OKCODE. This is the contents of the SAGGUI command filed, where you usually enter a transaction number.
MES_MENU	A request for a specific menu option
MES_SET_CUR_POS	A set of one or more controls on the screen, and one of them has focus
MES_SET_SIZE	A change in the size of user area of the screen
MES_HSCROLL	A request for a horizontal scroll. Specify the row or column position to scroll to with the lPos member of IT_EVENT.
MES_VSCROLL	A request for a vertical scroll. Specify the row or column position to scroll to with the lPos member of IT_EVENT.
MES_HELP_MES	A request for the help of the current message
MES_HELP_MENU	A request for the help of the current menu
MES_HELP_OK	A request for the help on the OK code field (the Command field in the SAPGUI). Note that this type of help information is not particularly interesting, because it always returns the same help information: how to use the Command field.

MES_HELP_FKEY	A request for the help of the push-button key. This is the help you get when pressing F1 while a toolbar button has focus. Note that this feature is not useful in recent R/3 releases.

Procedure

For example, if you wish to send an event that contains choosing of a toolbar key (sending a PFKey to R/3), you must perform the following steps:

1. Set the MES_KEY bit in eventtype on. This bit indicates that the event structure you are sending contains a PFKey.
2. Specify the key to send to R/3, by entering this value in IT_PFKEY.
3. Send the event, by using It_SendEvent or It_SendEventEx.

Example

The following example changes the size of the screen. It sets the MES_SET_SIZE to indicate that there is a change in the size of the screen, and then sends the event to the application server:

```
pEvt->eventtype |= MES_SET_SIZE ;
    printf("Row: %d\tCol:%d \n", pEvt->screen.dimrow, pEvt->screen.dimcol);
    unsigned short temp = pEvt->screen.dimrow * 1.4;
    pEvt->screen.dimrow = temp;
    temp = pEvt->screen.dimcol * 1.2;
    pEvt->screen.dimcol = temp;
    if (It_SendEvent(hndl, &pEvt) == FALSE) {
        printf("error in sendevent\n");
        return;
    }
```

The GUI Library offers several helper functions for changing some of the controls on the screen (some of the functions that start with ItEv_Set*). These functions internally set the appropriate MES_* bit when necessary, so that you do not have to do so when using these functions

For example, the function ItEv_SetOkCode assisgns the OKCode you specify to the okcode member of the IT_EVENT structure, and it sets the MES_OKCODE bit on:

```
lstrcpy(pEvt->okcode, okcode);
pEvt->eventtype |= MES_OKCODE;
```

Bits Set up by GUI Library

The GUI library sets the other bits in eventtype as a result of the information coming from the R/3 application server regarding the data that exists in the event structure (and all of its related structures).

You use the values of the bits in eventtype after getting an event from with either the It_GetEvent or the It_GetEventEx functions, to see which type of information exists in the event. For example, if EVT_MESSAGE is on, you know that the szMessage member of the IT_EVENT structure contains a message, and you can obtain the contents of this message. If the EVT_MESSAGE is off, you do not need to check for such a message.

Eventtype Member of IT_EVENT

Eventtype EVT_* Values

The following table lists the eventtype bits that you can query after sending an event. The table describes what each of the bits indicates:

Bit	If On, It Indicates that…
EVT_SCREEN	The event structure contains the screen and controls definition. Note that if this bit is off it means that the event structure had been initialized, but it does not contain an event.
EVT_MESSAGE	The event structure contains message.
EVT_STATIC_INFO	The static information portion of the IT_EVENT structure is available, this means that the information of the szDB and nDiagVersion members, for example, is available.
EVT_TITLE	SzNormTitle contains the title of the screen.
EVT_PFKEY	The event structure contains key definitions.
EVT_DIALOG_DISMISSED	(This bit is not used in current release).
EVT_OKCODE	The event structure contains an okcode value.
EVT_MENU	The menu structure is active on the screen.
EVT_TERM_CLR	Terminal clear. This is not used much in recent releases of R/3. It was used mostly by older, screen-based R/2 systems, for example, to clear the screen before displaying another screen.
EVT_DYNPRO_INFO	szProgramName and szScreenName fields in the Screen structure contain the name of the program and screen.
EVT_SEND_FRONT_EVT_REQ	The event was sent from the SAPGUI Front.
EVT_END_OF_SESSION	The session associates with this event had been closed. This could occur for example, as a result of a user logging off, or as a result of a user closing the window of the current session.
EVT_FRONT_RUNNING	SAPGUI Front is running. (Check this flag before sending an event to Front).
EVT_END_OF_TRANSACTION	End of the transaction was received for current transaction in process.
EVT_SESSION_ADDED	A new session associated with this event had started.

Example

The following example checks if there is a title to the window, and then prints it:

```
if(pEvt->eventtype & EVT_TITLE) {
      printf("Title: %s\n", pEvt->szNormTitle);
      }
```

The following example checks if there is a message on the screen, and then prints it:

```
if(pEvt->eventtype & EVT_MESSAGE) {
```

```
printf("Message !!: %s\n", pEvt->szMessage);
}
```

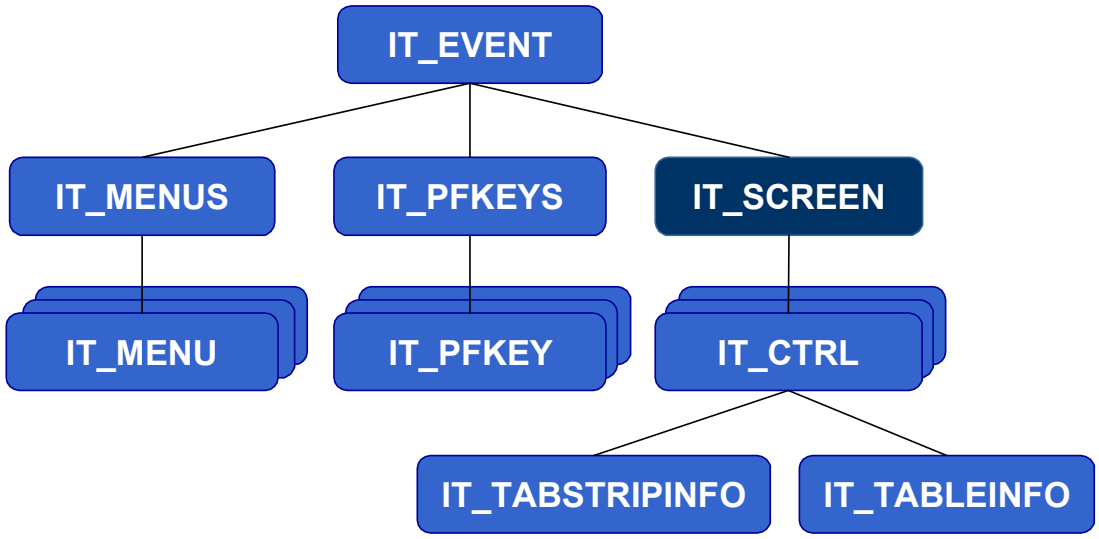
IT_SCREEN

Use

The IT_SCREEN structure provides information on the screen that is part of the event. It describes general attributes of the screen, such as its type and size.

The IT_SCREEN structure also points to an array of all the controls on the screen: one IT_CTRL structures for each control.

The parent structure, IT_EVENT, contains only one IT_SCREEN.



Syntax

```
typedef struct IT_SCREEN_t {
  long iCtrlCnt;
  PIT_CTRL pCtrl;
  long iModal;
  long flags;
  unsigned short scr_row;
  unsigned short scr_col;
  unsigned short scr_XWid;
  unsigned short scr_YWid;
  unsigned char cur_row;
  unsigned char cur_col;
  unsigned char dimrow;
  unsigned char dimcol;
  unsigned char dimlistrow;
  unsigned char dimlistcol;
  unsigned char VSliderSize;
  unsigned char HSliderSize;
  unsigned short lmrows;
```

```
unsigned char lmcols;
unsigned short lsrow;
unsigned char lscol;
char szProgramName[MAX_PROGRAM_NAME]; // if EVT_DYNPRO_INFO
char szScreenName[MAX_SCREEN_NAME];   // if EVT_DYNPRO_INFO
} IT_SCREEN;
```

Members

iCtrlCnt	Number of controls in this screen.
pCtrl	Array of IT_CTRL structures.
iModal	Screen modal number. Identifies the screen.
flags	Screen attributes
scr_row	Starting row for modal dialog box; 0 if not modal.
scr_col	Starting column for modal dialog box; 0 if not modal.
scr_XWid	Number of columns in modal dialog box; 0 if not modal.
scr_YWid	Number of rows in modal dialog box; 0 if not modal.
cur_row	Row of cursor position. If changed, MES_SET_CUR_POS is to be turned on
cur_col	Column of cursor position. If changed, MES_SET_CUR_POS is to be turned on
dimrow	Dimension of dynpro rows. If changed, MES_SET_SIZE is to be turned on.
dimcol	Dimension of dynpro columns. If changed, MES_SET_SIZE is to be turned on.
dimlistrow	Dimension of list rows (currently write-only). If changed, MES_SET_SIZE is to be turned on.
dimlistcol	Dimension of list columns (currently write-only). If changed, MES_SET_SIZE is to be turned on.
VSliderSize	Number of step loop or list rows being shown on screen.
HSliderSize	Number of step loop or list columns being shown on screen.
lmrows	Total number of rows in step loop or list.
lmcols	Total number of columns in step loop or list.
lsrow	Starting row for step loop or list in current screen.
lscol	Starting column for step loop or list in current screen.
szProgramName	Name of program, available only when EVT_DYNPRO_INFO is on in the parent event structure.
szScreenName	Name of screen, available only when EVT_DYNPRO_INFO is on in the parent event structure.

IT_SCREEN

Comments

This structure is valid only when EVT_SCREEN is turned on.

See also

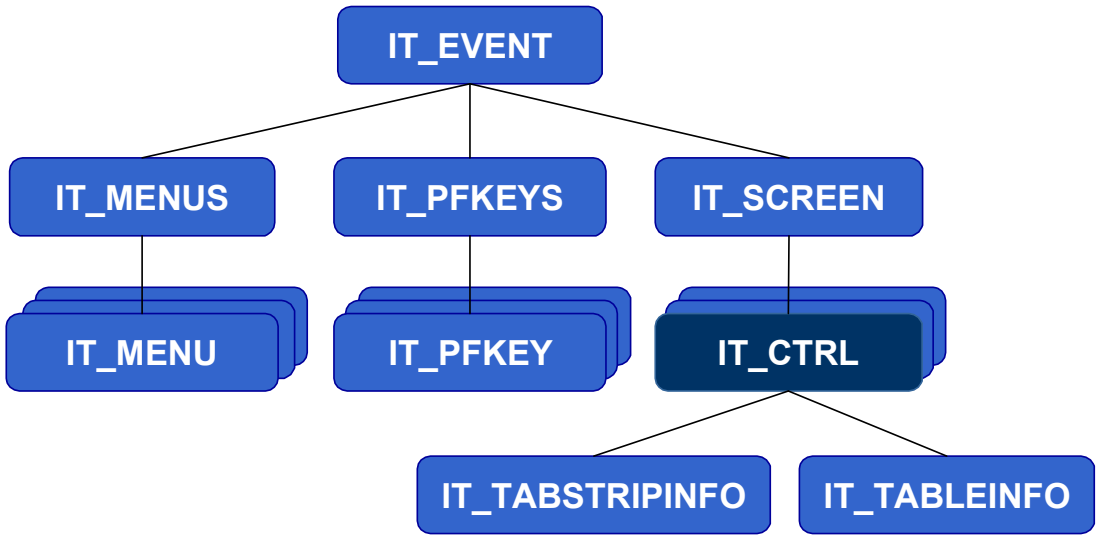
IT_CTRL [Page 87].

IT_CTRL

Use

The IT_CTRL structure provides information on a single control on the screen.

The parent structure, IT_SCREEN points to as many IT_CTRL structures as there are controls on the screen.



The IT_CTRL structure describes the type of control, its location on the screen, its relationship to other controls on the same screen, and it also indicates its state, for example, whether the control was changed or selected in the current event.

If the control is a table control or if it is a tab strip, then the IT_CTRL also points to the IT_TABLEINFO [Page 100] or IT_TABSTRIPINFO [Page 103] structures, holding information specific to the table or the tab strip respectively.

Syntax

```
typedef struct IT_CTRL_t {
  short dlgtype;
  short flags;
  char name[MAX_NAME];
  char value[MAX_VALUE];
  long top;
  long left;
  long bottom;
  long right;
  long dwStyle;
  long dwOffset;
  unsigned short nAreaID;
  unsigned short nBlockID;
```

IT_CTRL

```
  unsigned short nGroupID;
  unsigned short nContainerID;
  HANDLE hCtl;
  short cParent;
  short cChild;
  short cNext;
  short nColor;
  char szTableName[MAX_TABLE_NAME];/* if CTRL_FLAGS_FIELDNAMES */
  char szFieldName[MAX_FIELD_NAME];/* if CTRL_FLAGS_FIELDNAMES */
  char szDataElement[MAX_DATA_ELEM];     /* only GV_EXTENDED_SCREEN */
  char szDESupplement[MAX_DE_SUPP];/* only GV_EXTENDED_SCREEN */
  } IT_CTRL;
typedef IT_CTRL *PIT_CTRL;
#define IT_CTRL_SIZE        (sizeof(IT_CTRL))
```

Members

dlgtype [Page 90]	Values describing the control type
flags [Page 92]	Indicates additional information about the control if the control had changed during this event.
name	String name of control. If CTRL_FLAGS_FIELDNAMES is on, the batch input name of the control; otherwise a string indicating the type of control.
value	Contents or value of control.
top	Coordinates of top of the control on screen. See the discussion of Specifying Control Coordinates [Page 64].
left	Coordinates of the left of the control on screen. See the discussion of Specifying Control Coordinates [Page 64].
bottom	Coordinates of bottom of the control on screen. See the discussion of Specifying Control Coordinates [Page 64].
right	Coordinates of right of the control on screen. See the discussion of Specifying Control Coordinates [Page 64].
dwStyle [Page 93]	Flags indicating visual style of the control:
dwOffset	If CTRL_FLAGS_CONTROLINFO is set (which indicates that the control described by IT_CTRL is either a table or a tabstrip control), then dwOffset is the offset to add to the IT_EVENT pointer to get a to the table control or the tab strip control information. If the control is not a table control or a tabstrip control, then dwOffset is zero.
nAreaID	Screen area ID.
nBlockID	Block ID within a step loop.
nGroupID	Group ID for radio buttons. Radio buttons with the same area, block, and group Ids are in the same group and are mutually exclusive.

nContainerID	Container ID for a manager. For non-manager controls, the container ID for the control's parent manager.
hCtl	Handle to the control.
cParent	Index of parent of this control. -1 if no parents.
cChild	Index of the first child of this control.
cNext	Index to the next peer of this control.
nColor	Color number of specified control.
szTableName	Name of table, available when CTRL_FLAGS_FIELDNAMES is set.
szFieldName	Name of field, available when CTRL_FLAGS_FIELDNAMES is set.
szDataElement	Data element, available only when GV_EXTENDED_SCREEN is used.
szDESupplement	Data element supplement, available only when GV_EXTENDED_SCREEN is used.

Comments

All fields in IT_CTRL are read-only except value and bModified. To send information back to the server, change "value" and set bModified to TRUE.

See Also

IT_EVENT [Page 76], IT_SCREEN [Page 84], IT_TABLEINFO [Page 100], Control Hierarchy [Page 62], Specifying Control Coordinates [Page 64], Finding Controls [Page 65].

Control Type (Dlgtype) Values

Dlgtype is a member of the IT_CTRL structure [Page 87]. Its values describe the control type.

The following table lists the various control types and it describes the meaning of the control type when their name is not intuitive:

Dlgtype	Description
CTRL_STATIC	A label
CTRL_EDIT	An edit box. Also a table cell (the intersection of a row and a column)
CTRL_PASSWORD	Edit box for entering a password, which uses asterisks to hides the data entered
CTRL_PUSHBUTTON	
CTRL_RADIOBUTTON	
CTRL_CHECKBOX	
CTRL_FRAMEBOX	
CTRL_LINE	
CTRL_MATCH	Matchcode down arrow
CTRL_LISTSTATIC	Group of checkboxes
CTRL_GRAPHSTATIC	
CTRL_MATCHFIX	Matchcode that is not changeable
CTRL_ICON	
CTRL_LISTCHECKBOX	
CTRL_TABLE	A table control
CTRL_TABLE_COLUMN	A control that includes a whole column in a table control. It contains the column caption (CTRL_TABLE_CAPTION) and all the cells in the column (each of which is a CTRL_EDIT). In the column of the row selection buttons (CTRL_TABLE_SELECTBTN) it also contains the row selection buttons.
CTRL_TABLE_CAPTION	The title of a table column in a table control
CTRL_TABLE_SELECTBTN	Row selection button control
CTRL_TABSTRIP	Tabstrip control containing tabs and a tabstrip page area
CTRL_TABBUTTON	The tab button in a tab strip control
CTRL_MANAGER	Control manager for the page area of a tab strip

See the topics discussing the tab strip control [Page 55] and the table control [Page 58] for a description of the relationship between the various elements within these controls.

IT_CTRL Flags Values

A member of ths IT_CTRL structure [Page 87], flags is a bit mask that provides information on controls that had changed during the current event.

Check these values after getting an event to see what changed on the screen.

The following table lists the various flags and their meaning:

Flag Bit	If set, Indicates that…
CTRL_FLAGS_SELECTED	The control had been selected (this is relevant for checkbox and radio button controls, for example)
CTRL_FLAGS_MODIFIED	The control had been modified (this is relevant for an edit box control for example)
CTRL_FLAGS_FIELDNAMES	szTableName or szFieldName members of the IT_CTRL structure are available.
CTRL_FLAGS_SELECTABLE	Currently not in use.
CTRL_FLAGS_CONTROLINFO	The control is either a table control or a tabstrip control. The IT_CTRL therefore contains the extended information that is available only for these two types of controls.
CTRL_FLAGS_USER_UPDATE	The value field of the control had changed.
CTRL_FLAGS_SELECT_POS	The cursor position had changed.
CTRL_FLAGS_LOCAL_TAB	The tab page is a local tab page. (As opposed to being a server tab page). See the topic Handling a Tab Strip Control [Page 55].

DwStyle Values

A member of ths IT_CTRL structure [Page 87], dwStyle is a bit mask describing the drawing style of the control on the screen. The following is the list of values it can take:

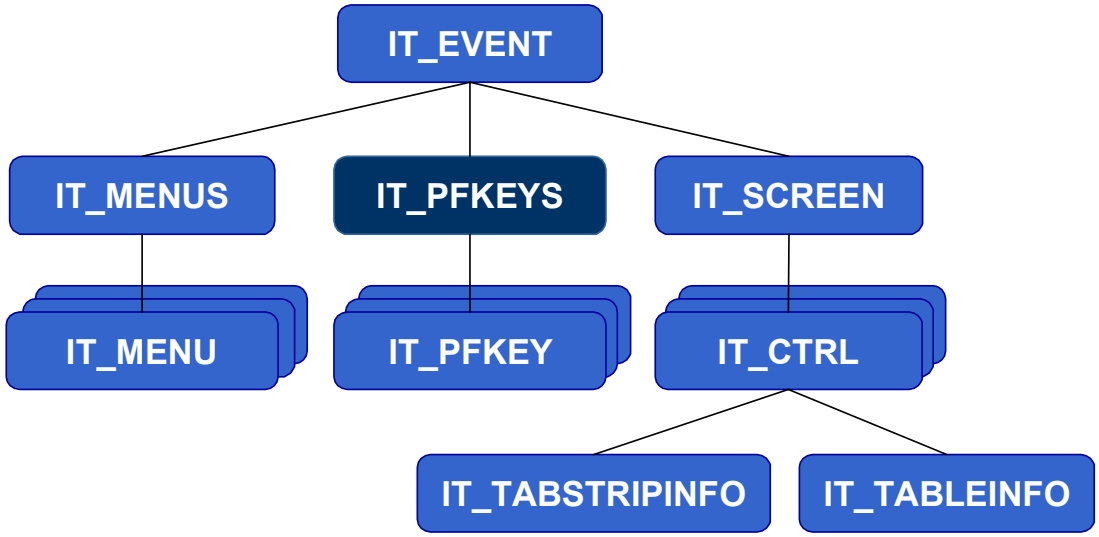
CTRL_CS_LEFT	CTRL_CS_FIXCOMBOBOX
CTRL_CS_CENTER	CTRL_CS_BORDER
CTRL_CS_RIGHT	CTRL_CS_CAPTION
CTRL_CS_SHADOW	CTRL_CS_TITLEBAR
CTRL_CS_GRAB	CTRL_CS_VSCROLL
CTRL_CS_FOCUS	CTRL_CS_HSCROLL
CTRL_CS_RDONLY	CTRL_CS_BUTTONBAR
CTRL_CS_VISIBLE	CTRL_CS_DRAGABLE
CTRL_CS_SELECT	CTRL_CS_DROPABLE
CTRL_CS_DISABLE	CTRL_CS_SYMBOLFONT
CTRL_CS_PROPFONT	CTRL_CS_UPPERCASE
CTRL_CS_INVERSE	CTRL_CS_INPUT_UPPER
CTRL_CS_SHORT_FCODE	CTRL_CS_HOTSPOT
CTRL_CS_3D	CTRL_CS_PASSWORD
CTRL_CS_COMBOBOX	CTRL_CS_SYMBOLRIGHT
CTRL_CS_INTENSIVE	

IT_PFKEYS

Use

The IT_PFKEYS structure indicates how many function keys exist in the window associated with the current event. It also points to an array of IT_PFKEY structures [Page 95], each of which contains information about a single function key.

The list of function keys associated with a screen can be viewed by clicking the right mouse button in a SAPGUI screen.



Syntax

```
typedef struct IT_PFKEYS_t {
  int iPfkeyCnt;
  PIT_PFKEY pPfKey;
  } IT_PFKEYS;
typedef IT_PFKEYS *PIT_PFKEYS;
```

Members

iPfkeyCnt	Number of PFKEYs
pPfKey	Array of IT_PFKEY structures

See Also

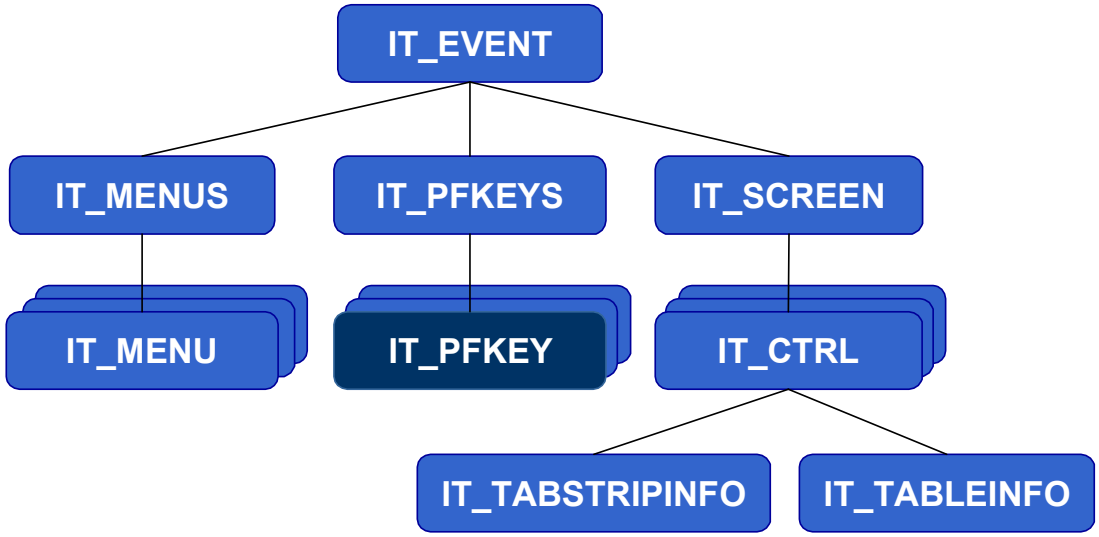
IT_PFKEY [Page 95].

IT_PFKEY

Use

The IT_PFKEY structure describes a single function key (used to be called a PFKey).

The list of function keys associated with a screen can be viewed by clicking the right mouse button in a SAPGUI screen. The IT_PFKEY structure describes one of the keys in that list.



Syntax

```
typedef struct IT_PFKEY_t {
  char name[80];
  char toollabel[80];
  char info[80];
  char accel[8];
  short flags;
  short order;
  int iVKValue;
  int iValue;
} IT_PFKEY;
typedef IT_PFKEY *PIT_PFKEY;
#define IT_PFKEY_SIZE           (sizeof(IT_PFKEY))
```

Members

name	Name of function key. This is equivalent to the function text in SAPGUI.
toollabel	The label of the button on the toolbar (if the key has a toolbar button).

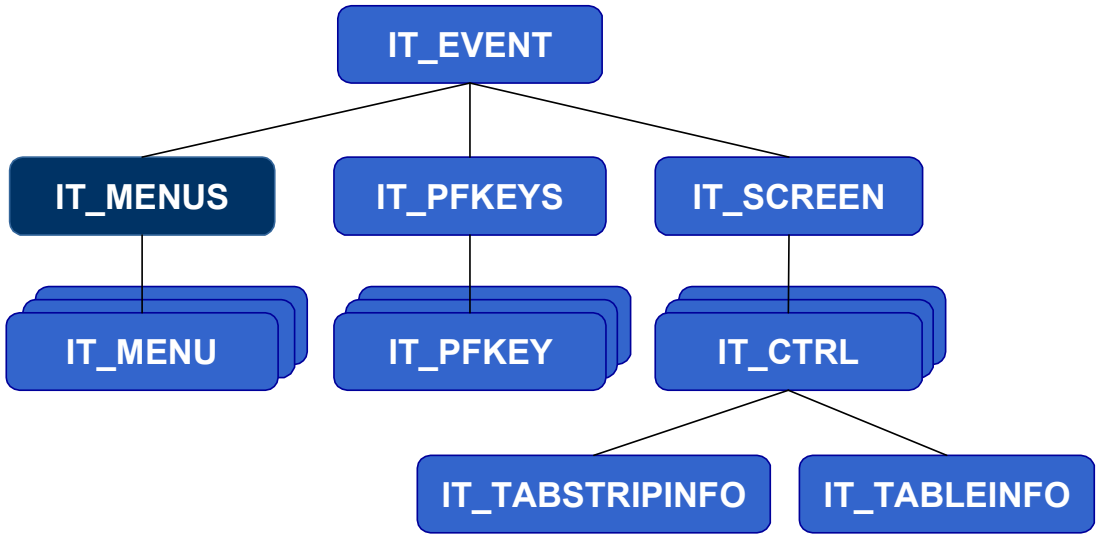
IT_PFKEY

info	Tooltip text, which is the text that shows when a user moves the mouse over the toolbar button (if the key has a toolbar button). This is the information in the Info text field in the ABAP Menu painter.	
accel	(not used for function keys)	
flags	Indicates whether the key appears on the toolbar. To see if the key is a toolbar key, mask the flas with PF_SHORT.	
order	Order of the key's button in the toolbar (if the key has a toolbar button).	
iVKValue	Value in Windows Virtual-Key codes [Page 51].	
iValue	SAP PFKEY value. This is the SAP code for the function keys. This code is for the F1 to F12 keys and key combination as follows:	
	F1 to F12	1 to 12
	Shift+F1 to Shift+F12	13 to 24
	Ctrl+F1 to Ctrl+F12	25 to 36
	Ctrl+Shift+F1 to Ctrl+Shift+F12	37 to 48

IT_MENUS

Use

IT_MENUS indicates how many menu items exist in the window associated with the current event. It also points to an array of IT_MENU structures, each of which describes a single menu items.



Syntax

```
typedef struct IT_MENUS_t {
  int iMenuCnt;
  PIT_MENU pMenu;
  } IT_MENUS;
typedef IT_MENUS *PIT_MENUS;
```

Members

iMenuCnt	Number of menus
pMenu	Array of IT_MENU structures

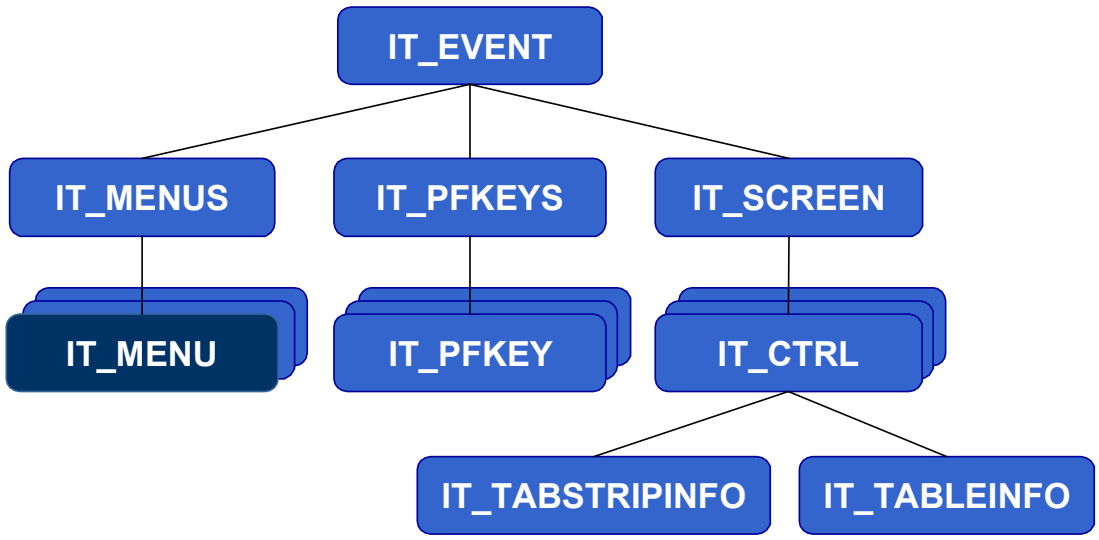
See Also

IT_MENU [Page 98].

IT_MENU

Use

The IT_MENU structure describes a single menu item, describing its status and type, as well as its location in the menu.



Syntax

```
typedef struct IT_MENU_t {
  char szName[MAX_MENU_NAME];
  char info[MAX_MENU_NAME];
  short flags;
  int iVKValue;
  int iValue;
  HANDLE hMenu;                 // Internal handle
  short cParent;
  short cChild;
  short cNext;
  short cFlags;
  } IT_MENU;
typedef IT_MENU *PIT_MENU;
#define IT_MENU_SIZE            (sizeof(IT_MENU))
```

Members

szName	Name of menu item
info	info for menu item
flags	Flags for this key

iVKValue	Value in Windows Virtual-Key codes
iValue	SAP PFKEY value (0 to 99)
hMenu	Internal handle for menu
cParent	Index of parent menu
cChild	Index of child menu
cNext	Index of next peer menu
cFlags	Flags for this menu

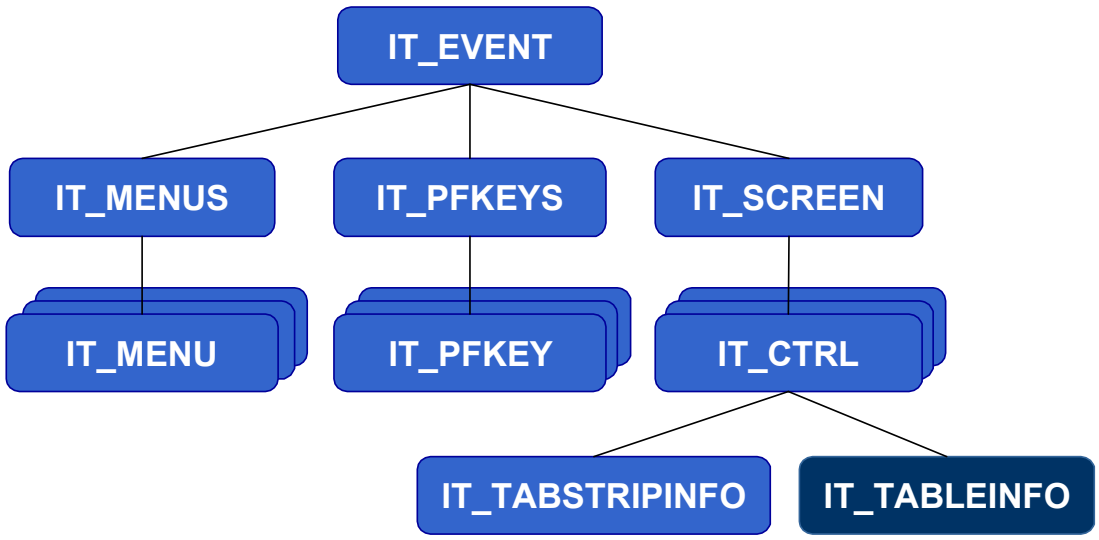
CFlags Values

MN_POPUP	This menu is a popup menu.
MN_EXPANDED	This popup menu has been expanded.
MN_ACTIVE	The menu is active.
MN_POPUP_REQUESTED	This popup menu is currently being expanded.

IT_TABLEINFO

Use

The IT_TABLEINFO structure describes a table control [Page 58] on the screen associated with the current event.



Syntax

```
typedef struct IT_TABSTRIPINFO_t {
short sType;
short sSize;
int flags;

int iTabs; /* number of pages */
int iLocalTabs; /* number of local pages      */
int iRows; /* number of tab rows     */
int iHeight; /* height of each row     */

short cLeft; /* Idx of left tab */
short cActive; /* Idx of active tab */
} IT_TABSTRIPINFO;

typedef IT_TABSTRIPINFO *PIT_TABSTRIPINFO;
typedef PIT_TABSTRIPINFO *PPIT_TABSTRIPINFO;
#define IT_TABSTRIPINFO_SIZE (sizeof(IT_TABSTRIPINFO))
```

Members

sType	Type of parent control

sSize	Size of control info
flags [Page 102]	Table control style flags
TabNumColumns	Number of columns in table
TabNumRows	Number of row in table shown on the screen
TabNumFixCols	Number of fixed columns
TabVerScrollbarLines	Total number of data rows in table
TabVerScrollbarStartRow	Starting row for current table display
TabHorScrollbarStartCol	Starting column for current table display
TabControlId	OK Code to get this table control's control icon

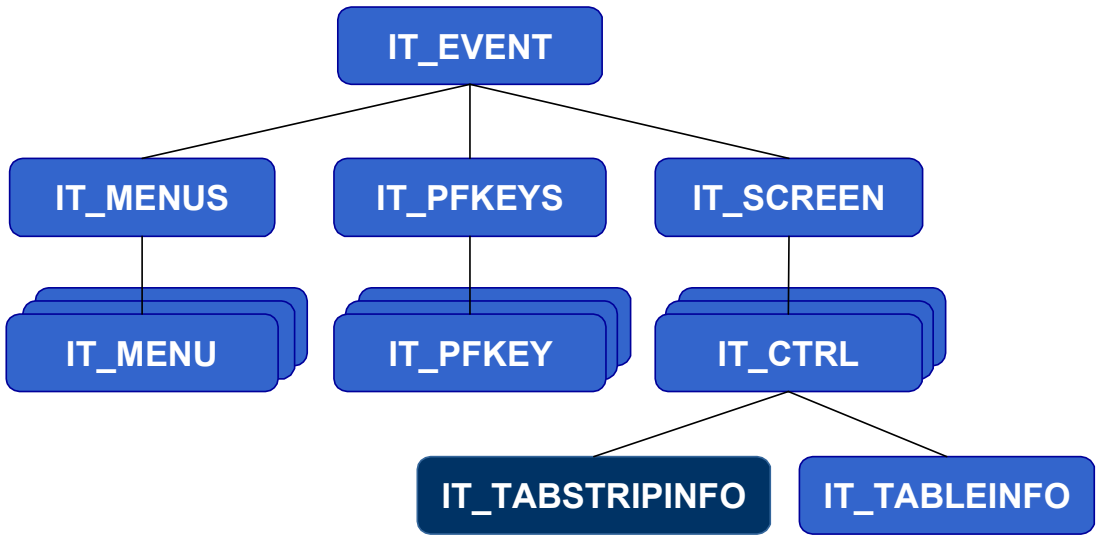
Table Control Style Flags

TABLE_CS_SELTYPE_SINGLE_ROW
TABLE_CS_SELTYPE_SINGLE_COL
TABLE_CS_SELTYPE_MULTIPLE_ROW
TABLE_CS_SELTYPE_MULTIPLE_COL
TABLE_CS_SELTYPE_NONE_ROW
TABLE_CS_SELTYPE_NONE_COL
TABLE_CS_SELECTOR_ROW
TABLE_CS_H_GRID
TABLE_CS_V_GRID
TABLE_CS_CUSTOMIZE

IT_TABSTRIPINFO

Use

IT_TABSTRIP describes a single tab strip control [Page 55] on the screen associated with the current event.



Syntax

```
typedef struct IT_TABSTRIPINFO_t {
      short sType;
      short sSize;
      int flags;
      int iTabs;
      int iLocalTabs;
      int iRows;
      int iHeight;
      short cLeft;
      short cActive;
} IT_TABSTRIPINFO;

typedef IT_TABSTRIPINFO *PIT_TABSTRIPINFO;
typedef PIT_TABSTRIPINFO *PPIT_TABSTRIPINFO;
#define IT_TABSTRIPINFO_SIZE (sizeof(IT_TABSTRIPINFO))
#define IT_GETCONTROLINFO(pEvt, idx) (((char *)pEvt)+pEvt-
>screen.pCtrl[idx].dwOffset)
```

Members

sType	Type of parent control

IT_TABSTRIPINFO

sSize	Size of control info
flags [Page 105]	Flags describing the drawing style for the tab strip control
iTabs	Number of pages in a tab strip
iLocalTabs	Number of local pages in tab strip shown on the screen
iRows	Number of tab rows
iHeight	Height of each row
cLeft	Index of the left most tab on the screen
cActive	Index of the active tab

Tab Strip Control Style Flags

A bit mask describing the drawing style of a tab strip control.

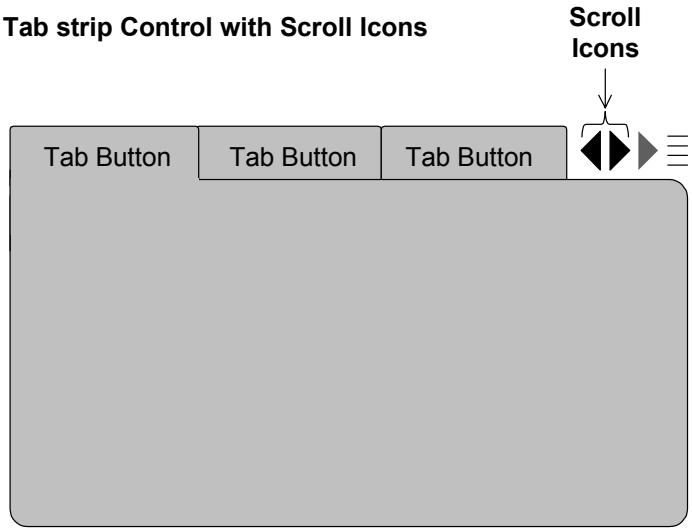
The following table lists the bits in the flag and their description:

Bit	Description
TABSTRIP_CS_TABS_TOP	Indicates that the tabs are at the top of the area of the tab strip control. Currently this is the only arrangement possible for tabs in a tab strip.
TABSTRIP_CS_TABS_BOTTOM	Not currently in use.
TABSTRIP_CS_TABS_LEFT	Not currently in use.
TABSTRIP_CS_TABS_RIGHT	Not currently in use.
TABSTRIP_CS_SCROLL_LL	If there are more tabs than can fit on top of the tab strip control area, the tab strip control includes two scroll icons: one for scrolling to the left and one for scrolling to the right. See the illustration below. If this bit is on, the two scroll icons appear to the left of the tabs.
TABSTRIP_CS_SCROLL_LR	If this bit is on, the left scroll icon is to the left of the tabs, and the right scroll icon is to the right of the tabs.
TABSTRIP_CS_SCROLL_RR	If this bit is on, the two scroll icons appear to the right of the tabs.
TABSTRIP_CS_TEXT_VERTICAL	If this bit is on, the text of the tab buttons is displayed vertically.
TABSTRIP_CS_TAB_AS_TAB	If this bit is on, the tab strip uses tabs. If it is off, the tab strip uses buttons to emulate the tabs at the top of the control. See the illustrations below.

The following illustration shows the scroll icons that appear if there are more tab buttons than can fit at the top of the tab strip control area. In this illustration, both scroll icons are to the right of the tabs. (The TABSTRIP_CS_SCROLL_RR bit is on in this case):

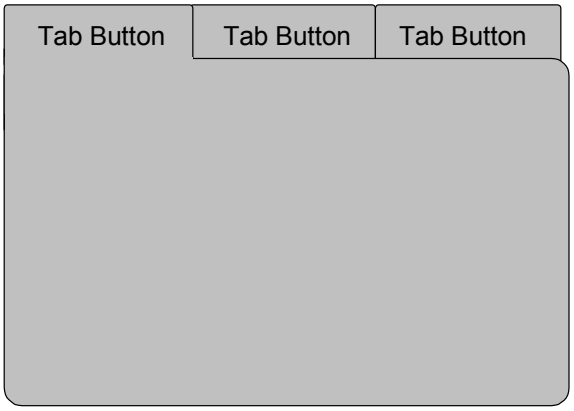
Tab Strip Control Style Flags

Tab strip Control with Scroll Icons



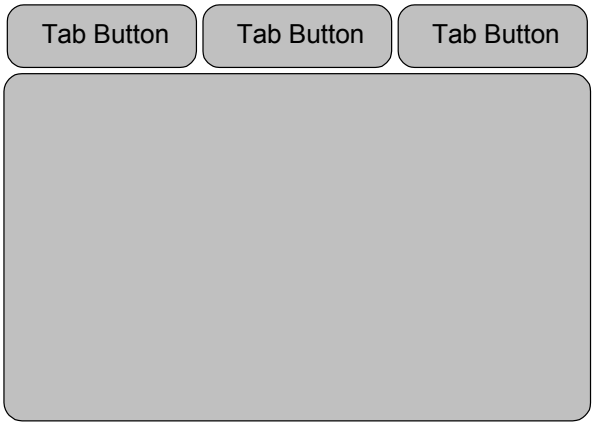
The following illustration shows a tab strip control with the tabs displayed as tabs (TABSTRIP_CS_TAB_AS_TAB bit is on):

Tab strip Control with Tabs as Tabs



The following illustration shows a tab strip control with the tabs displayed as buttons (TABSTRIP_CS_TAB_AS_TAB bit is off):

Tab strip Control with Tabs as Buttons



Note that when tab strip tabs are displayed as buttons, it is harder for the end user to tell which of the tabs is selected.

Functions

It_AbortGetEvent

Use

Aborts the operation of getting an event.

Syntax

```
void DLEX It_AbortGetEvent (HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE on success, FALSE on error.

It_Dup

Use

Duplicates an IT_EVENT structure.

Syntax

```
PIT_EVENT DLEX It_Dup (PIT_EVENT pEvt);
```

Parameters

pEvt	Pointer to the IT_EVENT structure

Return Value

Pointer to an IT_EVENT structure that duplicates the input parameter.

See Also

It_DupTo [Page 111].

It_DupTo

Use

Duplicates a source IT_EVENT structure into a target IT_EVENT structure you specify.

Syntax

```
DWORD DLEX It_DupTo(PIT_EVENT pEvt, PPIT_EVENT ppEvt);
```

Parameters

pEvt	Pointer to the IT_EVENT input structure
ppEvt	Pointer to a pointer to the IT_EVENT output structure

Return Value

Returns TRUE on success.

Comments

The tagert ppEvt parameter points to the pointer to the duplicated structure.

See Also

It_Dup [Page 110].

It_FreeConnection

Use

Frees a connection object, which closes the connection to R/3.

Syntax

```
DWORD DLEX It_FreeConnection (HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE on success, FALSE on error.

Comments

If the server connection is still valid, then call It_Logoff to log off from the server, and then call It_FreeConnection to close the connection.

It_FreeEvent

Use

Frees the IT_EVENT structure pointed to by the ppEvt parameter.

Syntax

```
DWORD DLEX It_FreeEvent(PPIT_EVENT ppEvt);
```

Parameters

ppEvt	Pointer to a pointer to the IT_EVENT structure

Return Value

Returns TRUE on success.

It_GetEvent

Use

Gets an event, by filling out the information from the R/3 application server into the GUI Library's event structure and all of the related structures.

A call to It_GetEvent is a blocking call, meaning that it waits for a server response.

Syntax

```
DWORD DLEX It_GetEvent(HANDLE hMr, PPIT_EVENT ppEvt);
```

Parameters

hMr	Connection handle
ppEvt	Pointer to a pointer to the IT_EVENT structure. The client defines a PIT_EVENT and initializes it to zero. On return this value will be changed to point to a properly structured IT_EVENT structure.

Return Value

Returns TRUE on success, FALSE on error.

Comments

It_GetEvent calls It_GetEventEx with standard flags (*flgs* set to zero).

It_GetEvent can read each event once. After getting the event information with It_GetEvent, the same event is no longer available for you to read again, meaning that a subsequent call to It_GetEvent will wait for the next event. If you wish to read the information of an even more than once, use It_PeekEvent [Page 132], instead.

After getting an event, you can read the bits in the eventtype member of IT_EVENT [Page 80] to get a preview of the type of information that exists in the event structure.

See Also

It_GetEventEx [Page 115], It_PeekEvent [Page 132]

It_GetEventEx

Use

It_GetEventEx is an extended version of the It_GetEvent [Page 114] function.

Similar to the It_GetEvent function, It_GetEventEx gets an event and fills out the GUI Library's event structure and all of the related structures with information on the event. Like the It_GetEvent function, It_GetEventEx is a blocking call.

It_GetEventEx extends the functionality of It_GetEvent in that it allows you to use flags to control how the event information is obtained.

One of the important flags you can use with the It_GetEventEx is the GV_GETFRONTEVENT, with which you specify that the event is obtained from the SAPGUI Front, instead of from the R/3 application server.

Syntax

```
DWORD DLEX It_GetEventEx(HANDLE hMr, PPIT_EVENT ppEvt, DWORD flgs);
```

Parameters

hMr	Connection handle
ppEvt	Pointer to a pointer to the IT_EVENT structure. The client application defines a PIT_EVENT and initializes it to zero. On return this value will be changed to point to a properly structured IT_EVENT structure.
flgs [Page 117]	Flags to control the operation of It_GetEventEx. Using It_GetEventEx with *flgs* set to zero is equivalent to using It_GetEvent.

Return Value

Returns TRUE on success, FALSE on error.

Comments

It_GetEventEx is the base level API to all the other It_GetEvent APIs.

After getting an event, you can read the bits in the eventtype memebr of IT_EVENT [Page 80] to get a preview of the type of information that exists in the event structure.

Example

```
// Go into a debug loop to dump out messages from the
// application server.
while(It_GetEventEx(hMerlin, &pEvt,
  GV_PEEK_EVENT | GV_DEBUG_DUMP)) {
      if(pEvt->eventtype & EVT_END_OF_SESSION)
        break;
}
```

It_GetEventEx

See Also

It_GetEvent [Page 114], It_PeekEvent [Page 132], It_NewConnection [Page 127].

It_GetEventEx Flags

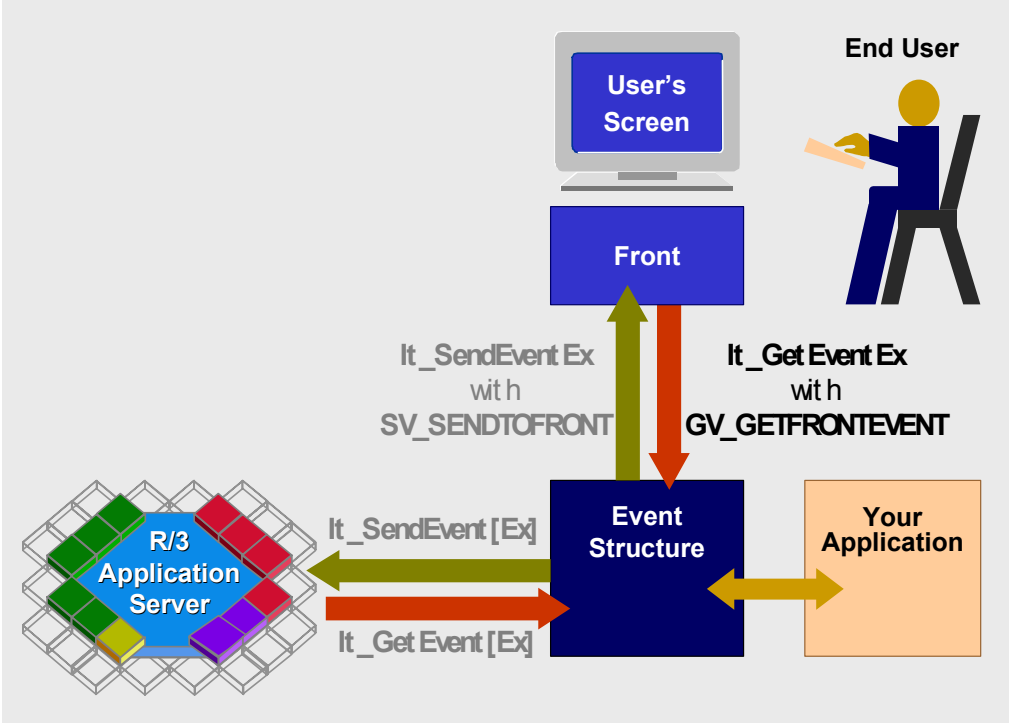
A set of bits creating a mask that controls how the It_GetEventEx is used.

The following table lists the bits in the It_GetEventEx flags and their role.

Bit	Role
GV_PEEK_EVENT	When you turn this bit on, It_GetEventEx behaves like an It_PeekEvent [Page 132]: it retrieves the event without clearing the internal flag indicating that the event has been read. This means that the same event can be retrieved again. Used mainly for error detection.
GV_DEBUG_DUMP	Turns on event structure information dumping. As a result, the information on all of the controls on the screen is displayed at the console. The information displayed at the console is the same as the information that is printed out when using It_ListControls [Page 124]. Useful for debugging.
GV_GETVERSION	Returns the R/3 version of the system from which the event is retrieved.
GV_QUERYEVENTSPENDING	Returns TRUE if any events are currently pending, meaning that an event had been sent to the R/3 application server, but R/3 had not responded yet.
GV_EXTENDED_SCREEN	Extracts the technical information on all the controls on the screen (the same information that you get when choosing F1 and then choosing *Technical Info* in a SAPGUI screen).
GV_ISCONNECTED	Checks if the connection to R/3 is valid. Returns TRUE if R/3 is connected.
GV_ISSAPGUIRUNNING	Returns TRUE if SAPGUI Front is running for this session.
GV_GETFRONTEVENT	Gets the event from the SAPGUI Front, instead of from the R/3 application server.

The following illustration shows the flow of the event information when using the GV_GETFRONTEVENT flag.

It_GetEventEx Flags



It_GetEventPtr

Use

Gets a pointer to the connection's current event.

Syntax

```
PPIT_EVENT It_GetEventPtr(HANDLE hMr);
```

Parameters

hMr	Handle to an opened connection

Return Value

Pointer to a pointer to the connection's current IT_EVENT structure.

It_GetTransaction

Use

Invokes the specified R/3 transaction.

Using the It_GetTransaction is a shortcut: it is equivalent to performing the following steps:

1. Setting the value of the *okcode* member of the IT_EVENT structure to the desired transaction code
2. Turning on the *MES_OKCODE* flag of in the *eventtype* member of the IT_EVENT structure
3. Sending a return or sending the event structure to the R/3 application server (with It_Send_Event, for example).

Syntax

```
DWORD DLEX It_GetTransaction(HANDLE hMr, const char *szTransaction);
```

Parameters

hMr	Connection handle
szTransaction	Pointer to a transaction code, for example *SO01*

Return Value

Returns TRUE on success, FALSE on error.

Comments

It_GetTransaction is implemented using the Get Event/Send Event mechanism. The following code shows the implementation of the It_GetTransaction inside the GUI Library.

```
DWORD DLEX It_GetTransaction(HANDLE hMerlin, char *trans) {
  char tr[MAX_OK+4];
  PIT_EVENT pEvt = 0;
  sprintf(tr, "/n%s", trans);
  It_GetEvent(hMerlin, &pEvt);
  lstrcpy(pEvt->okcode, trans);
  pEvt->eventtype |= MES_OKCODE;
  return It_SendReturn(hMerlin, &pEvt);
  }
```

Example

```
// Start the outbox transaction
It_GetTransaction(hMerlin, "soo2");
```

See Also

ItEv_SetOKCode [Page 171], It_SendReturn [Page 140].

It_GroupLookup

Use

Gets a list of groups that are defined for a specific R/3 system.

This is the list you would get when using the SAP logon dialog, choosing the *Groups* button, selecting a system from the list and then choosing *List*.

Syntax

```
HANDLE DLEX It_GroupLookup (const char *id, const char *ms,
                        const char *router, Char **groups, int
*groupCnt);
```

Parameters

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
ms	Hostname of the message server. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one message server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the message server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
groups	List of groups available for given server ID
groupCnt	Number of groups found

Return Value

Returns TRUE if successful, otherwise returns FALSE.

See Also

It_NewGroupConnection [Page 130]

Example

The following code prints the list of groups:

```
// Where
//  id = SAP SystemID ,
//  ms = Hostname of the message server,
//  router = Destination router used to connect to the Message Server
//  groupCnt = Number of groups found.
//  groups   = List of groups available for given server ID

if (It_GroupLookup(id, ms, router, &groups, &groupCnt))
{

int i;
for(i = 0; i < groupCnt; i++)
```

It_GroupLookup

```
{
char *str = NEXT_STR(groups, i);
printf("\t%s\n", str);
}
}
else
{
printf("grouplookup: It_grouplookup failed. \n");
}
```

The following code returns the third group in the list of groups:

```
//    hndl =      Connection handle
hndl = It_NewGroupConnection(id, ms, router, NEXT_STR(groups, 2), SAPGUI_FRONT);
```

It_IsGuiRunning

Use

Indicates whether the SAP GUI Front is running or not.

Syntax

```
DWORD DLEX It_IsGuiRunning(HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE if SAPGUI is running, FALSE otherwise.

It_ListControls

Use

Reports information on all the controls on the screen. You can print this information, or you can display it on the screen.

To use this function you must first write a C function specifying the display or print format for the report. You then must point to your C function by using the It_SetDumpHook [Page 144] function. For more information and an example, see Listing Screen and Control Information [Page 67].

The information provided includes:

- The id of the screen (the *iModal* member of the IT_SCREEN structure [Page 84])
- The *name*, *value*, *left*, and *top* members of the IT_CTRL structure [Page 87]

Syntax

```
DWORD DLEX It_ListControls(HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE on success.

See Also

The ItEv_DumpEvent [Page 149] function displays a larger set of information: it lists all of the items on the screen associated with the current event, not only controls.

It_Login

Use

Logs into an SAP System with or without a standard login screen using the specified client, user name, password, and language.

Call It_Login after establishing a connection toR/3 (by using one of the connection functions: It_NewConnection [Page 127], It_NewGroupConnection [Page 130], or It_NewServerConnection [Page 131]).

Syntax

```
DWORD DLEX It_Login (HANDLE hMr, const char *client,
    const char *name, const char *passwd, const char *lang);
```

Parameters

hMr	Connection handle
client	Client string
name	User name string
password	Password string
lang	Language string

Return Value

Returns TRUE on success, FALSE on error.

See Also

It_Logoff [Page 126], It_NewConnection [Page 127], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131]

It_Logoff

Use

Logs off from an active connection.

Syntax

```
DWORD DLEX It_Logoff(HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE on success otherwise FALSE.

Comments

The connection handle is no longer valid after this call.

See Also

It_Login [Page 125], It_NewConnection [Page 127], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131]

It_NewConnection

Use

Connects to the SAP application server.

Returns a connection handle, which you then use in most subsequent It_* function calls.

Syntax

```
HANDLE DLEX It_NewConnection (const char *host,
    const char *systemno, long flgs);
```

Parameters

host	This is the host name of the application server. This can also be a net ID.
systemno	System number of the application server to connect to
flags [Page 128]	Flags to control the connection operation. Flags also allow you to specify how the coordinates of controls on the screen are expressed (affects all events associated with the connection). The same flags are used in all of the connection functions.

Return Value

Returns the handle to an SAP Automation GUI session if successful, otherwise returns FALSE.

Example

```
// Opens a connection to the D22 application server.
FLGS = 0; // connect to R3 without invoking SapGui.
hMerlin = It_NewConnection ("apd2105", "22", FLGS);
```

See Also

It_Login [Page 125], It_Logoff [Page 126], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131]

Connection Functions Flags

The connection functions flags control the connection operation.

The connection functions flags also allow you to specify how the controls on the screen are treated. More specifically, with these flags you can specify that the coordinates of the controls on the screen are expressed relative to the parent controls, as opposed to as absolute coordinates from the top of the screen.

Applies to

It_NewConnection [Page 127], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131]

Values

The following table lists the flags and their description.

SAPGUI_FRONT	Starts SAPGUI Front with connection startup. The default is to not invoke Front.
SAPGUI_R2	Indicates that the connection is to an R2 system. The default is to connect to an R3 system.
SAPGUI_FULLMENU	Gets the complete menu tree from the server. Note that SAPGUI in releases prior to 4.5A does not support the menu tree feature. Do not use this flag with SAPGUI Front of earlier releases.
SAPGUI_ABSOLUTE_COORD	Specifies that the coordinates of controls on the screen are expressed as their absolute location on the screen. (The default is to express the coordinates of controls on the screen as their location relative to their parent control, if they have a parent.)
SAPGUI_45A_COORD	Specifies that the coordinates of controls on the screen are expressed as a mixture of absolute and relative location, keeping the behavior of 4.5A GUI Library in this respect intact. This flag is provided only to preserve compatibility with applications using GUI Library of versions earlier and up to 4.5A. (The default is to express the coordinates of controls on the screen as their location relative to their parent control, if they have a parent.)

SAPGUI_ACTIVEX	Specifies that the data of ActiveX controls on an R/3 application screen is communicated directly between the R/3 application server and the SAP GUI Front, without being communicated to the GUI Library. This flag is provided as a limited support for ActiveX controls on a SAP GUI Screen. Use this flag with a caution: using this flag in your application prohibits you from sending any event to R/3 or to Front. The default is to *not* provide this ActiveX support. For more details, see the discussion in Using Screens with ActiveX Controls [Page 69].

The default value for the flags is zero (0). If you specify zero, the connection functions take the default behavior, which is to not invoke Front, to not provide ActiveX support, and to connect to an R/3 system. It also then uses relative coordinates for controls on the screen.

It_NewGroupConnection

Use

Connects to an SAP Application server R3 using the Group information.

Using Groups takes advantage of load balancing: when using It_newGroupConnection, the application server that is actually used is determined when you log on. Therefore, the It_NewGroupConnection function provides a more dynamic way of connecting to the SAP System than the It_NewServerConnection [Page 131], because it finds the most suitable application server for you to log onto.

This should be the preferred way to access an SAP System.

Syntax

```
HANDLE DLEX It_NewGroupConnection (const char *id, const char *ms,
                    const  char *router, Const char *group, long
flags);
```

Parameters

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
ms	Hostname of the message server. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one Message Server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the message server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
group	A group from a list of defined Groups in the system. Use the name of the group as it appears in the SAP Logon list. This group is used to logon to an arbitrary application server in this Group. You can obtain that list of groups by using the It_GroupLookup function [Page 121].
flags [Page 128]	Flags to control the connection operation. Flags also allow you to specify how the coordinates of controls on the screen are expressed (affects all events associated with the connection). The same flags are used in all of the connection functions.

Return Value

Returns the handle to an SAP Automation GUI session if successful, otherwise returns FALSE.

See Also

It_Login [Page 125], It_Logoff [Page 126], It_NewConnection [Page 127], It_NewServerConnection [Page 131]

It_NewServerConnection

Use

Connects to a specified SAP application server. Note that using It_NewServerConnection does not take advantage of load balancing as provided when you use It_NewGroupConnection [Page 130]. When using It_NewServerConnection you specify a particular server machine to connect to.

Syntax

```
HANDLE DLEX It_NewServerConnection (const char *id, const char *ms,
 Const char *router, const char *server, long flags);
```

Parameters

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
ms	Hostname of the message server. The message server routes the service to various application servers. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one message server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the message server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
server	The IP address of the application server to connect to. If you want to connect to an application server from a list of currently available servers in a system, you can obtain this list of servers by using the It_ServerLookup function [Page 141]. It_ServerLookup returns the IP addresses of the available servers.
flags [Page 128]	Flags to control the connection operation. Flags also allow you to specify how the coordinates of controls on the screen are expressed (affects all events associated with the connection). The same flags are used in all of the connection functions.

Return Value

Returns the handle to an SAP Automation GUI session if successful, otherwise returns FALSE.

See Also

It_Login [Page 125], It_Logoff [Page 126], It_NewConnection [Page 127], It_NewGroupConnection [Page 130], It_ServerLookup [Page 141]

It_PeekEvent

Use

Like the It_GetEvent [Page 114] and the It_GetEventEX [Page 115] functions, It_PeekEvent gets the information of an event into the event structure.

However, unlike It_GetEvent*, It_PeekEvent allows you to read the same event multiple times. (When using It_GetEvent, you can get a specific event only once. Any subsequent It_GetEvent call waits for the next event to occur. In contrast, using It_PeekEvent multiple times reads the same event).

You can use It_PeekEvent after sending an event (with It_SendEvent*) to look at the R/3 response. You can look at this response event multiple times before getting that event with an It_GetEvent* call.

Note, however, that if you first use It_GetEvent* to get the event, using It_PeekEvent cannot get that event anymore. It will be blocking for the next event to occur.

Syntax

```
DWORD DLEX It_PeekEvent(HANDLE hMr, PPIT_EVENT ppEvt);
```

Parameters

hMr	Connection handle.
ppEvt	Pointer to a pointer to the IT_EVENT structure. The client defines a PIT_EVENT and initializes it to zero. On return, this value will be changed to point to a properly structured IT_EVENT structure.

Return Value

Returns TRUE on success, FALSE on error.

See Also

It_GetEvent [Page 114], It_GetEventEx [Page 115], It_PeekTitle [Page 133].

It_PeekTitle

Use

It_PeekTitle combines an It_PeekEvent with a comparison of the title of the screen to a title you specify.

This can be useful for error-checking the results of an It_SendEvent call (to see that the resulting screen is correct).

Syntax

```
DWORD DLEX It_PeekTitle(HANDLE hMr, PPIT_EVENT ppEvt, char *ptitle);
```

Parameters

hMr	Connection handle
ppEvt	Pointer to a pointer to the IT_EVENT structure
ptitle	Event title to search for

Return Value

TRUE if title of event matches ptitle argument; FALSE otherwise. Match test is case-insensitive and searches for initial substring.

Comments

Note that screen titles are not always unique in R/3, and therefore checking the title of the screen does not guarantee that you have accessed the correct screen. When working with R/3 application servers from release 3.1H and later, checking the *szProgramName* and *szScreenName* fields of the IT_SCREEN structure can be a more accurate way of checking the identification of the screen.

See Also

It_PeekEvent [Page 132], IT_SCREEN [Page 84].

It_RegisterCallback

Use

Registers a callback function that monitors usage of GUI Library function calls. Once you register such a callback, then when your program calls various functions for example, when it calls ItEv_SetValue), your application's callback function is invoked.

You can use this for debugging an application, for example.

Syntax

```
typedef DWORD (CALLBACK *LPFNLOGCALLBACK) (HANDLE hMr,
    PPIT_EVENT ppEvt, long fnc_called, long changed,
    LPARAM idx, LPARAM word02);
DWORD DLEX It_RegisterCallback (HANDLE hMr,
    LPFNLOGCALLBACK lpfncallback);
```

Parameters

hMr	Connection handle
lpfncallback	Pointer to the callback function

Return Value

Returns a pointer to the previous callback function.

See Also

Callback Functions and Macro Recording [Page 66].

It_SendEvent

Use

Sends the contents of the event structure (and all the related structures) to the R/3 application server.

Use It_SendEvent after altering the event structure programmatically, or after the user had changed the screen, resulting

Before sending an event, make sure you set up the appropriate bits in the eventtype member of IT_EVENT [Page 80].

Syntax

```
DWORD DLEX It_SendEvent(HANDLE hMr, PPIT_EVENT ppEvt);
```

Parameters

hMr	Connection handle
ppEvt	Pointer to a pointer to the IT_EVENT structure. The value of PIT_EVENT should be one that was retrieved from It_GetEvent*. If successful, on return PIT_EVENT will be zeroed out and the buffer freed.

Return Value

Returns TRUE on success, FALSE on error.

Example

The following example simulates sending the Execute key (F8 on the keyboard, which has VK_F8 or 119 key value)

```
//  pEvt is a valid pointer to PIT_EVENT structure.
if (ItEv_SetPFKey(pEvt, VK_F8) == FALSE)
{
   printf("SetPFKey Error!--ItEv_SetPFKey\n");
   // Handle error case
}
if (It_SendEvent(hHandle, &pEvt) == FALSE)
{
   printf("Send Event Error while setpfkey!--It_SendEvent\n");
   // Handle error case
}
if (It_GetEvent(hHandle, &pEvt) == FALSE)
{
   printf("Get Event Error! Program aborted!\n");
   // Handle error case
}
```

See Also

It_SendEventEx [Page 137]

lt_SendEvent

It_SendEventEx

Use

It_SendEventEx is an extended version of the It_SendEvent [Page 135] function.

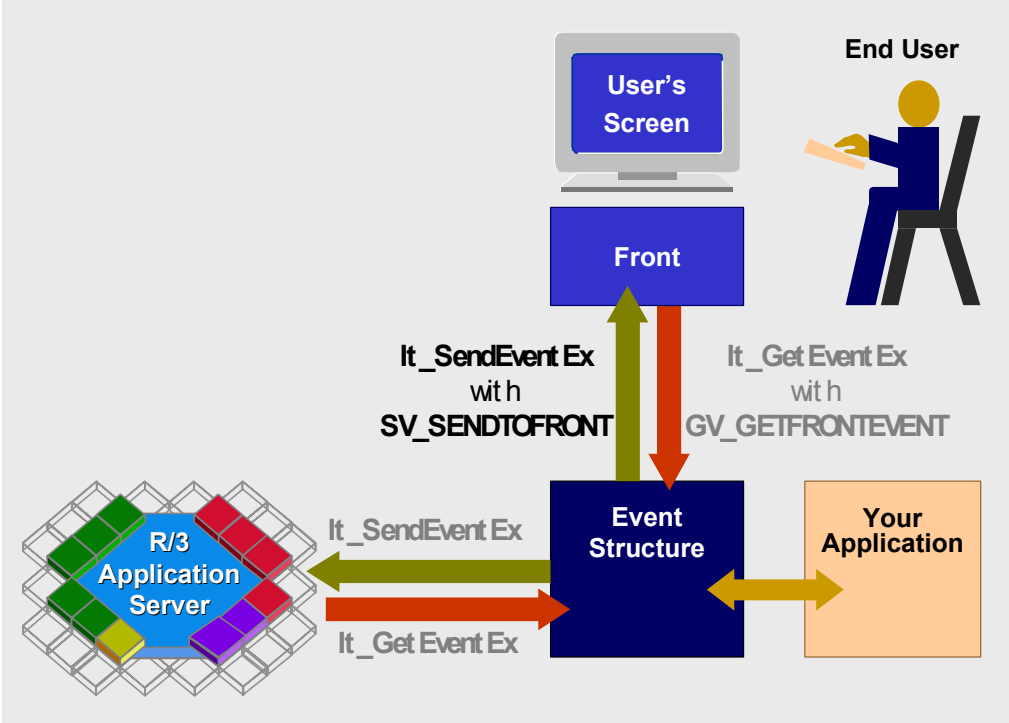
Similar to the It_SendEvent function, It_SendEventEx allows you to send out the GUI Library's event structure and all of its related structures to the R/3 application server.

In contrast to the It_SendEvent, however, It_SendEventEx allows you to send the contents of the event structure to the SAPGUI Front, instead of to the R/3 application server. You do so by using the *flgs* parameter of this function.

You can use It_SendEventEX after getting an event from either the R/3 application server or from Front, and after your program or the user has made changes to the data in the event structure or in any of the related structures.

You do not have to get an event before sending an event to Front.

The following illustration shows the flow of the event information when using the GV_GETFRONTEVENT flag of the It_SendEventEx function.



Syntax

```
DWORD DLEX It_SendEventEx(HANDLE hMr, PPIT_EVENT ppEvt, DWORD flgs);
```

Parameters

hMr	Connection handle

It_SendEventEx

ppEvt	Pointer to a pointer to the IT_EVENT structure. The value of PIT_EVENT should be one that was retrieved from It_GetEvent*. If successful, on return PIT_EVENT will be zeroed out and the buffer freed.
flgs	Flags to control the operation of It_SendEventEx. Currently the only *flgs* value is: **SV_SENDTOFRONT** Using this *flgs* value indicates that the event is should be sent to the SAPGUI Front, instead of to the R/3 application server. Using It_SendEventEx with *flgs* set to zero is equivalent to using It_SendEvent, which means that the event is sent to the R/3 application server. If you are using the It_SendEvent function to send the event to the R/3 application server (by setting *flgs* to zero), remember to set the relevant bits in eventtype member of IT_EVENT [Page 80] before you send the event to the R/3 application server. Note that if you are sending the event to Front you do ***not*** need to set the evettype bits.

Return Value

Returns TRUE on success, FALSE on error.

It_SendPFKeyID

```
DWORD DLEX It_SendPFKeyID(HANDLE hMr, PPIT_EVENT ppEvt,
    const char *pfkey);
```

Parameters

hMr	Connection handle
ppEvt	Pointer to a pointer to the IT_EVENT structure
pfkey	Name of the key to be sent (initial substring, case-insensitive)

Return Value

Returns TRUE on success, FALSE on error.

Comments

Finds the IT_PFKEY with a *name* field matching the pfkey argument, sets the IT_EVENT *key* field to the key's *iVKValue* field, and sends the event.

See also

It_SendEvent [Page 135], ItEv_SetPFKeyID [Page 173], ItEv_FindPFKeyID [Page 157].

It_SendReturn

```
DWORD DLEX It_SendReturn(HANDLE hMr, PPIT_EVENT ppEvt);
```

Parameters

hMr	Connection handle
ppEvt	Pointer to a pointer to the IT_EVENT structure

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sets the IT_EVENT *key* field to VK_RETURN and sends the event. This has the same effect as using the Enter key or toolbar button when using the SAPGUI.

See Also

It_SendEvent [Page 135], It_SendPFKeyID [Page 139].

It_ServerLookup

Use

Gets a list of IP Addresses of the servers that are defined for a specific R/3 system.

The list of IP addresses corresponds to the list you would get when using the SAP logon dialog, choosing the *Server* button, selecting a system from the list, and then choosing *List*.

Syntax

```
HANDLE DLEX It_ServerLookup (const char *id, const char *ms, const char *router,
Char **servers, int *serverCnt);
```

Parameters

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
ms	Hostname of the message server. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one message server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the Message Server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
servers	Returns list of the IP Adresses of the servers available for given server ID.
serverCnt	Number of servers found.

Return Value

Returns TRUE if successful, otherwise returns FALSE.

Comments

To obtain a single IP address from the list returned by It_ServerLookup, you can use the following macro:

NEXT_STR (list, index)

Parameters

list	List of IP addresses (as returned from the It_ServerLookup function)
index	Zero-based index to the list

You must free this list by using the free () call of the C runtime library after complete processing of this list.

Example

The following code prints the list of servers:

```
//  id = SAP SystemID ,
//  ms = Hostname of the message server,
//  router = Destination router used to connect to the Message Server
```

It_ServerLookup

```
//  servers = Returns list of the servers available for given server ID
//  Number of servers found

if (It_ServerLookup(id, ms, router, &servers, &servCnt))
{
int i;
for(i = 0; i < servCnt; i++)
{
char *str = NEXT_STR(servers, i);

printf("\t%s\n", str);
}
}
else
{
    printf("serverLoopup: It_serverlookup failed. \n");
}
```

The following code returns the third server in the list of servers:

```
// hndl = Connection handle
hndl = It_NewServerConnection(id, ms, router,  NEXT_STR(servers, 2), 0);
```

See Also

You can use the It_ServerLookup in conjunction with the It_NewServerConnection [Page 131]

It_SetDelSessionHook

Use

Registers a callback function that monitors a session being deleted.

Syntax

```
DWORD DLEX It_SetDelSessionHook(HANDLE hMr, void *ptr);
```

Parameters

hMr	Connection handle
ptr	Pointer to the callback function

Return Value

Returns TRUE on success, FALSE on error.

See Also

It_SetNewSessionHook [Page 145], Handling Multiple Sessions [Page 40]

It_SetDumpHook

Use

Points to your C function in which you specify the formatting to use when either It_ListControls [Page 124] or ItEv_DumpEvent [Page 149] is called.

You must call this function before calling wither It_ListControls or ItEv_DumpEvent. Otherwise the information is not displayed.

Compatible with the standard printf function.

For more information and an example, see Listing Screen and Control Information [Page 67].

Syntax

```
DWORD DLEX It_SetDumpHook(HANDLE hMr, void *ptr);
```

Parameters

hMr	Connection handle
ptr	The printing function to use during It_ListControls or ItEv_DumpEvent

Return Value

Returns TRUE on success.

See Also

It_ListControls [Page 124], ItEv_DumpEvent [Page 149], Listing Screen and Control Information [Page 67]

It_SetNewSessionHook

Use

Registers a callback function that monitors a new session being created.

Syntax

```
DWORD DLEX It_SetNewSessionHook(HANDLE hMr, void *ptr);
```

Parameters

hMr	Connection handle
ptr	Pointer to the callback function

Return Value

Returns TRUE on success, FALSE on error.

See Also

It_SetDelSessionHook [Page 143], Handling Multiple Sessions [Page 40]

It_StartSapGui

Use

Starts SAPGUI Front.

This function allows you to start SAPGUI anytime during the connection to the application server.

Note that you can request to start SAPGUI Front when establishing a connection, by using the SAPGUI_FRONT flag of the connection function. However, using the It_StartSapGui is useful when you wish to log on automatically through your program, and then display the SAPGUI only after the logon had succeeded.

Syntax

```
DWORD DLEX It_StartSapGui (HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE on success, FALSE on error.

See Also

It_New_Connection [Page 127], It_NewGroupConnection [Page 130], It_NewServerConnection [Page 131], Connection Functions Flags [Page 128], It_StorpSapGui [Page 147]

It_StopSapGui

Use

Stops the SAPGUI Front.

If the SAPGUI Front is running, you need ot stop it before you log off from the R/3 application server or before exiting your program.

One reason, for example, for stopping the SAPGUI Front is if your program starts SAPGUI Front but then needs to exit before SapGui Front actually appears.

Note that your program may have started SAPGUI by either:

- Using the SAPGUI_FRONT flag with any of the connection functions
- Calling the It_StartSAPGUI function

Syntax

```
DWORD DLEX It_StopSapGui (HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns TRUE on success, FALSE on error.

See Also

It_StartSapGui [Page 146], It_Logoff [Page 126]

ItEv_CustomizeTable

Use

Invokes the Table settings dialog, which allows you to customize a table, if the table is customizable.

Syntax

```
DWORD DLEX ItEv_CustomizeTable(PIT_EVENT pEvt, ITCCTRL pCtrl)
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	Refer to Specifying control [Page 151]

Return Value

Returns TRUE on success, FALSE on error.

See Also

ItEv_FindControl [Page 154].

ItEv_DumpEvent

Use

Reports information about the event pointed to by *pEvt* to standard output. This includes information about the menu, the toolbar, and the various controls on the screen associated with the current event.

To use this function you must first write a C function specifying the display or print format for the report. You then must point to your C function by using the It_SetDumpHook [Page 144] function. For more information, see Listing Screen and Control Information [Page 67].

Syntax

```
DWORD DLEX ItEv_DumpEvent(PIT_EVENT pEvt);
```

Parameters

pEvt	Pointer to the IT_EVENT structure

Return Value

Returns TRUE on success.

See Also

Using the It_ListControls [Page 124] function you can get a subset of this information: the information on just the controls that are on the screen.

Finding Controls

Specifying the Control Parameter

For functions that accept an ITCCTRL parameter, such as: ItEv_GetControlInfo [Page 161], ItEv_SetCheck [Page 164], ItEv_SetControlInfo [Page 165], ItEv_SetCurPosByCtrl [Page 167], ItEv_SetTableColumnPermutation [Page 174], ItEv_SetValue [Page 175], and ItEv_SetWidth [Page 176], you can specify the ITCCTRL parameter using one of the following:

- Field name of the control (this is the default)
- Value name of the control (this is the contents of the field)
- Zero-based index of the control in the list of controls on the screen

You specify the control as a string. The string is interpreted as the name of the control by default, unless you use one of the following two macros to specify that value name or index should be used:

- To specify a control by its value name (by its contents) use the ITVALUE_NAME macro.

  For example, the following code selects (sets) the radio button whose label is "Attributes".

  ```
  ItEv_SetCheck(pEvt, ITVALUE_NAME("Attributes"), 1);
  ```

- To specify a control by its index (0-based), use the ITCTRL_IDX macro.

  The following example sets control # 4 on the screen (assuming that it is a check box or a radio button).

  ```
  ItEv_SetCheck (pEvt, ITCTRL_IDX(4), 1);
  ```

Specifying Controls for the ItEv_FindControl* functions

You can use the above two macros with the ItEv_FindControl* functions, but you only need to use them if you do not specify the flags [Page 152] parameter of these functions. For example, the following code finds the first control with “Address” as a value.

```
ItEv_FindControl(pEvt, ITVALUE_NAME("Address"), 0);
```

Because the above example does not specify flags (that is, it specifies zero for the flags parameter), you need to specify how to interpret the specification of the control, as with the other ITEv_* functions.

However, if you specify flags with the ItEv_FindControl* functions, it is redundant to use either ITVALUE_NAME or ITCTRL_IDX, because the flags parameter allows you to use the macros FC_FIND_VALUE or FC_FIND_FIELD to specify how to interpret the ITCCTRL parameter. As the matter of fact, once you use one of these macros in the flags parameter, your specification at the ITCCTRL parameter is ignored (that is, specifying the FC_FIND_VALUE or the FC_FIND_FIELD in flags takes precedence). See the discussion in the topic Specifying ItEv_FindControl* Flags [Page 152]

Specifying the Flags Parameter

The *flags* parameter of the ItEv_FindControl and ItEv_FindControlEx functions is a set of bits creating a mask that controls how you search for controls on the screen. The *flags* parameter has two roles:

- Specifying how to conduct the search, for example which direction to go when searching
- Specifying how to interpret the ITCCTRL parameter of the ItEv_FindControl or ItEv_FindControlEx function.

Specifying How the Search is Performed

The GUI Library provides a set of macros for specifying the various aspects of the search. The following table describes these macros and how to use them:

Macro	Use to Specify
FC_FIND_STARTING_FROM(*x*)	The index of the control to start the search from. *x* is the zero-based index of the control on the screen. For example, if you wish to start searching **after** the third control on the screen, specify FC_FIND_STARTING_FROM(3).
FC_LEFT	Searching to the left
FC_RIGHT	Searching to the right
FC_DOWN	Searching down
FC_UP	Searching going up
FC_COUNT(*int*)	How many controls to search through. *Int* is the number of controls to search. For example if you specified FC_FIND_STARTING_FROM(2), and you specify FC_COUNT(*6*) the search starts at the third control and continues for six controls, stopping the search after the eighth control.
FC_FIND_TYPE(*typ*)	The type of control to search for. For example *typ* can be CTRL_EDIT for an edit box type of control. See dlgtype values [Page 90] for the values you can specify for control type in *typ*.

You do not have to specify any of the above parameters of the search.

Specifying How to Interpret the ITCCTRL Parameter

You can also use the *flags* parameter of the ItEv_FindControl and ItEv_FindControlEx functions to specify how to interpret the specification of the ITCCTRL parameter, that is whether the ITCCTRL parameter specified using the name of the control or the contents of the control.

Use one of the following macros to specify how the ITCCTRL parameter of the function is interpreted:

Macro	Specifies that..
FC_FIND_VALUE	the control specified in the ITCCTRL parameter of the function is using the value (contents) of the control.
FC_FIND_FIELD	the control specified in the ITCCTRL parameter of the function is using the technical name of the control.

You do not have to use either of these macros. Another way to specify how to interpret the ITCCTRL parameter of the ItEv_FindControl or ItEv_FindControlEx function is to do so directly in the ITCCTRL parameter (see the topic Specifying the Control Parameter [Page 151]).

However, if you do use either the FC_FIND_VALUE or the FC_FIND_FIELD macros as part of specifying the *flags* parameter, this specification takes precedence over how you specify the control parameter directly in the function.

Note that these two macros are mutually exclusive: you can only use one of them. If you specify both, they are both ignored.

ItEv_FindControl

Use

Finds a control on the screen.

You can find a control by its name, by its value name (the name of the control that contains the value of the field), or by its index (a zero-based index of the control in the list of the controls on the screen). See the topic of specifying control parameter for the ItEV_Find* functions [Page 151].

Syntax

```
DWORD DLEX ItEv_FindControl(PIT_EVENT pEvt, ITCCTRL *pCtrl,
                            DWORD flags);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	A string identifying the control to find. Refer to Specifying control [Page 151] for how to specify the control parameter.
flags	Flags that determine how to search for the control. For example, it allows you to specify where to start the search for the control and in which direction. Refer to Specifying control flags [Page 152].

Return Value

Returns the 0-based index of the found control on the screen, or -1 if no control is found.

ItEv_FindControlEx

Use

The definitions of the FC_FIND_STARTING_FROM and FC_COUNT (specified as part of the flags parameter of ItEv_FindControl [Page 152]) do not allow for effective searching when there are more than 255 controls on the screen.

The ItEv_FindControlEX performs similar operation as the ItEv_FindControl function [Page 154], but it provides two parameters that allow you to specify a starting index and how many controls to search without the limitations of the FC_FIND_STARTING_FROM and FC_COUNT macros.

If you use them, then the *startindex* parameter overrides FC_FIND_STARTING_FROM for specifying the starting point for the search. The *count* parameter overrides FC_COUNT for specifying the distance of the search.

Syntax

```
DWORD DLEX ItEv_FindControlEx(PIT_EVENT pEvt, ITCCTRL *pCtrl,
                              DWORD startIndex, DWORD count, DWORD flags);
```

Parameters

pEvt	Pointer to the IT_EVENT structure.
pCtrl	A string identifying the control to find. Refer to Specifying control [Page 151].
startIndex	Starting point for the search. This overrides FC_FIND_STARTING_FROM in *flags*.
count	How many controls to search. This overrides FC_COUNT in *flags*.
flags	Flags that determine how to search for the control. For example, it allows you to specify where to start the search for the control and in which direction. Refer to Specifing control flag [Page 152].

Return Value

Returns the 0-based index of the control found, or -1 if no control is found.

See Also

ItEv_FindControl [Page 154].

ItEv_FindControlByPos

Use

Finds a control on the screen based on its position.

Syntax

```
DWORD DLEX ItEv_FindControlByPos (PIT_EVENT pEvt, long top, long left);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
top	Row number for the position in character coordinates
left	Column number for the position in character coordinates

The top and left coordinates are relative to the parent control by default, unless you have used one of the connection flags [Page 128] to change the way coordinates are specified [Page 64].

Return Value

This call returns the control index for a given location (top and left). The index is zero-based. If the control is not found, then –1 will be returned.

See Also

ItEv_FindControl [Page 154].

ItEv_FindPFKeyID

Use

Finds the first key on the screen whose name starts with the substring you specify.

For example, if you specify "Execute", then both the keys called *Execute* and *Execute with var* qualify. However, this function returns the first key it finds.

Syntax

```
DWORD DLEX ItEv_FindPFKeyID(PIT_EVENT pEvt, const char *pfkey);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pfkey	Initial substring to use in the search of the key names. Case-insensitive

Return Value

Returns the 0-based index of key that matches the search., or -1 if no match.

See Also

ItEv_SetPFKeyID [Page 173].

ItEv_GetAccelerator

Use

Finds the accelerator key associated with the specified menu item.

Syntax

```
DWORD DLEX ItEv_GetAccelerator (PIT_EVENT pEvt, int iMenu, char *accel,
int BuffLen, int *AccelLen);
```

Parameters

pEvt	Pointer to the IT_EVENT structure to change
iMenu	Index to the menu item
Accel	Buffer you allocate for the accelerator key string.
BuffLen	Length of the buffer you allocate in your program for the accelerator key. Maximum length is 256.
AccelLen	Actual length of the accelerator key string. Allocated by the client program, and filled in by the call. Currently this length is always 1.

Return Value

Returns TRUE on success, FALSE on error.

ItEv_GetControlCode

Use

Finds the internal function code associated with the specified control. It is mainly used for a push button or a tab button.

Syntax

```
DWORD DLEX ItEv_GetControlCode (PIT_EVENT pEvt, ITCCTRL pCtrl, char
*code, int len);
```

Parameters

pEvt	Pointer to the IT_EVENT structure to change
pCtrl	A string identifying the control to find. Refer to Specifying control [Page 151]
Code	Function code of the control
Len	Length of the function code

Return Value

Returns TRUE on success, FALSE on error.

ItEv_GetControlCount

Use

Gets the number of controls in the current event.

Syntax

```
DWORD DLEX ItEv_GetControlCount(PIT_EVENT pEvt);
```

Parameters

pEvt	Pointer to the IT_EVENT structure

Return Value

Returns the number of controls in the current event.

Comments

Use this function only if the EVT_SCREEN flag in eventtype [Page 80] is set, indicating that there is screen and control data. EVT_SCREEN is not set when the user area of a screen is empty.

See Also

IT_SCREEN [Page 84].

ItEv_GetControlInfo

Use

This is a convenience routine for getting the control information for either a table control or a tabstrip control.

Syntax

```
DWORD DLEX ItEv_GetControlInfo(PIT_EVENT pEvt, ITCCTRL pCtrl,
LPVOID pBuffer, int len);
```

Parameters

hMr	Handle to an opened connection
pCtrl	Refer to Specifying control [Page 151]
pBuffer	Pointer to buffer for additional control information
len	Length of pBuffer in bytes

Return Value

Returns TRUE on success, FALSE on error.

Comments

This information can also be obtained by adding the value of dwOffset in the IT_CTRL structure to the IT_EVENT pointer value, and getting the control information directly using this pointer.

See Also

ItEv_FindControl [Page 154], ItEv_SetControlInfo [Page 165].

ItEv_GetControlTooltip

Use

Returns the string of the tooltip associated with the specified control (the tooltip is the text the user sees when holding the mouse over an icon). It is mainly used for button controls that have an icon.

Syntax

```
DWORD DLEX ItEv_GetControlTooltip (PIT_EVENT pEvt, ITCCTRL pCtrl, char *Tip, int BuffLen, int *TipLen);
```

Parameters

pEvt	Pointer to the IT_EVENT structure to change
pCtrl	A string identifying the control to find. Refer to Specifying control [Page 151]
Tip	Buffer you allocate for the tooltip string. The maximum length of the tooltip string is 256.
BuffLen	Length of the memory buffer you have allocated in your program for the tooltip
TipLen	Length of the actual tooltip string. Allocated by the client program, and filled in by the call.

Return Value

Returns TRUE on success, FALSE on error.

If you allocate a buffer smaller than the size of the tooltip string, the function returns FALSE. However, in this case the function still returns the actual length of the tooltip in *TipLen*. You can use it to issue another call to this function with the correct buffer size allocated.

ItEv_GetSessionCount

Use

Gets the number of sessions in the R/3 connection.

Syntax

```
DWORD DLEX It_Ev_GetSessionCount (HANDLE hMr);
```

Parameters

hMr	Connection handle

Return Value

Returns the number of sessions.

Example

In the following example *hndl* is the connection handle:

```
int m_TotalSessions = ItEv_GetSessionCount(hndl);
```

See Also

Handling Multiple Sessions [Page 40]

ItEv_SetCheck

Use

Selects and de-selects radio buttons and check boxes: a value of 1 selects the control; a value of 0 deselects it.

If selecting from a group of radio buttons, it is the calling program's responsibility to ensure that only one radio button from the group is selected at a time.

Syntax

```
DWORD DLEX ItEv_SetCheck(PIT_EVENT pEvt, ITCCTRL pCtrl, int bSelected);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	A string identifying the control. Refer to Specifying control [Page 151]
bSelected	1 to select the control; 0 to deselect it

Return Value

Returns the previous value of the selection, or -1 if error.

Comments

This function does not affect non-selectable controls.

See Also

ItEv_FindControl [Page 154].

ItEv_SetControlInfo

Use

This is a convenience routine for setting the control information for either a table control or a tabstrip control.

Syntax

```
DWORD DLEX ItEv_SetControlInfo(PIT_EVENT pEvt, ITCCTRL pCtrl, LPVOID
pBuffer, int len);
```

Parameters

hMr	Handle to an opened connection
pCtrl	A string identifying the control. Refer to Specifying control [Page 151]
pBuffer	Pointer to buffer for additional control information
len	Length of pBuffer in bytes

Return Value

Returns TRUE on success, FALSE on error.

Comment

This information can also be changed by adding the value of dwOffset in the IT_CTRL structure to the IT_EVENT pointer value, and setting the control information directly using this pointer.

See Also

ItEv_FindControl [Page 154], ItEv_GetControlInfo [Page 161].

ItEv_SetCurPos

Use

Positions the cursor to a specific row and column anywhere on the screen.

Syntax

```
DWORD DLEX ItEv_SetCurPos(PIT_EVENT pEvt, int row, int col);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
row	Row number for the new cursor position, in character coordinates
col	Column number for the new cursor position, in character coordinates

Return Value

Returns TRUE on success.

Comments

It is usually more convenient to use ItEv_SetCurPosByCtrl [Page 167], which positions the cursor to the top left corner of a given control. ItEv_SetCurPos provides the flexibility to position elsewhere on the screen if needed, but placing the cursor not on a control does not have any meaning.

See Also

ItEv_SetCurPosByCtrl [Page 167].

ItEv_SetCurPosByCtrl

Use

Positions the cursor to the top left corner of the control.

Use it, for example, to position the cursor over a push button (followed by an It_SendReturn), or over a matchcode (followed by an It_SendPFKeyID(pEvt, “Save”).

Syntax

```
DWORD DLEX ItEv_SetCurPosByCtrl(PIT_EVENT pEvt, ITCCTRL pCtrl);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	Refer to Specifying controls [Page 151]

Return Value

Returns TRUE on success.

Comments

No effect if the control specified by pCtrl is not found.

See Also

ItEv_FindControl [Page 154], ItEv_SetCurPos [Page 166].

ItEv_SetMenu

Use

Sets the menu entry to be sent by directly specifying the menu handle of the appropriate IT_MENU structure.

Also sets the MES_MENU bit of eventtype [Page 80].

Syntax

```
DWORD DLEX ItEv_SetMenu(PIT_EVENT pEvt, HANDLE menuhandle);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
menuhandle	Menu handle from the hMenu field in the IT_MENU structure

Return Value

Returns TRUE on success.

Comments

ItEv_SetMenuID, which selects the menu item by index number, internally calls ItEv_SetMenu to set the menu entry.

See Also

ItEv_SetMenuID [Page 169].

ItEv_SetMenuID

Use

Sets the menu entry to be sent by specifying the menu index within the current event's IT_MENUS structure.

Also sets the MES_MENU bit of eventtype [Page 80].

Syntax

```
DWORD DLEX ItEv_SetMenuID(PIT_EVENT pEvt, int menuidx);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
menuidx	Index to the menu item to be sent, based on the current event's IT_MENUS structure

Return Value

Returns TRUE on success.

Comments

Using ItEv_SetMenuID allows you to support the same menu in different languages.

ItEv_SetMenuID, which selects the menu item by index number, internally calls ItEv_SetMenu to set the menu entry.

See Also

ItEv_SetMenu [Page 168].

ItEv_SetMenuKey

Use

Sets the menu entry to be sent by specifying the menu key in the key member of the IT_EVENT structure.

Also sets the MES_MENU bit of eventtype [Page 80].

Syntax

```
DWORD DLEX ItEv_SetMenuKey(PIT_EVENT pEvt, int key);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
key	The key equivalent of the menu

Return Value

Returns TRUE on success.

ItEv_SetOKCode

Use

Sets the OK code to be sent when the event structure is sent. (The OK code is the contents of the SAGUI Command filed, which usually contains a transaction code).

Syntax

```
DWORD DLEX ItEv_SetOKCode(PIT_EVENT pEvt, const char *okcode);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
okcode	String containing the new OK code to send

Return Value

Returns TRUE on success.

Comments

Sending an OK code of a transaction immediately invokes that transaction, ignoring other control changes that may have been made in the current screen or event.

See Also

IT_EVENT [Page 76].

ItEv_SetPFKey

Use

Sets the PFKey to be sent when the event structure is sent.

Syntax

```
DWORD DLEX ItEv_SetPFKey(PIT_EVENT pEvt, int key);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
key	The value to set the PFKey to. Use Windows Virtual-Key values [Page 51] to specify the key.

Return Value

Returns TRUE on success, false otherwise.

See Also

ItEv_SetPFKeyID [Page 173].

ItEv_SetPFKeyID

Use

Sets the key to be sent to the event to be the first key with an initial substring matching the *pfkey* argument in a case-insensitive search.

Syntax

```
DWORD DLEX ItEv_SetPFKeyID(PIT_EVENT pEvt, const char *pfkey);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pfkey	Name of the key to be sent (initial substring, case-insensitive)

Return Value

Returns TRUE on success, FALSE on error.

See Also

ItEv_SetPFKey [Page 172], It_SendPFKeyID [Page 139].

ItEv_SetTableColumnPermutation

Use

Sets the order of table columns in a CTRL_TABLE control, by specifying the new position of each of the columns as values in an array.

Indices and values for the pCols array are both zero-based.

For example, specifying pCols[0] = 2 means that the first column now goes to the third position in the table.

Syntax

```
DWORD DLEX ItEv_SetTableColumnPermutation(PIT_EVENT pEvt, ITCCTRL
pCtrl, int* pCols, int nCols);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	The table control whose columns should be switched. Refer to Specifying control [Page 151]
pCols	Array of integers specifying new table column positions
nCols	Length of pCols array

Return Value

Returns TRUE on success, FALSE on error.

See Also

ItEv_FindControl [Page 154].

ItEv_SetValue

Use

Sets the values of any of the controls in the event structure (and its related structures) directly. This is useful for setting the value of an editing box or a checkbox, for example.

Syntax

```
DWORD DLEX ItEv_SetValue(PIT_EVENT pEvt, ITCCTRL pCtrl, const char
*val);
```

Parameters

pEvt	Pointer to the IT_EVENT structure to change
pCtrl	Specifies the control to change. Refer to Specifying control [Page 151]
val	New value of the control

Return Value

Returns TRUE on success, FALSE on error.

Comments

Below is the internal implementation of ItEv_SetValue:

```
DWORD DLEX ItEv_SetValue(PIT_EVENT pEvt, ITCCTRL pCtrl, char *val) {
     int ctrl;  // internal conversion of pCtrl to ctrl not shown here
     lstrcpy(pEvt->screen.pCtrl[ctrl].value, val);
     pEvt->screen.pCtrl[ctrl].bModified = 1;
     return TRUE;
     }
```

See Also

ItEv_FindControl [Page 154], IT_EVENT [Page 76], IT_CTRL [Page 87].

ItEv_SetWidth

Use

Resets the width of a CTRL_TABLE_COLUMN control, which represents the column header.

Syntax

```
DWORD DLEX ItEv_SetWidth(PIT_EVENT pEvt, ITCCTRL pCtrl, int width);
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	The control representing the column header. Refer to Specifying control [Page 151]
width	New table column width

Return Value

Returns TRUE on success, FALSE on error.

Comments

Setting the width is only applicable to a table control, and not to a step table control.

See Also

ItEv_FindControl [Page 154].

ItEv_SetTabButton

Use

Set the active tab in a tab strip control.

Syntax

```
DWORD DLEX ItEv_SetTabButton(PIT_EVENT pEvt, ITCCTRL pCtrl)
```

Parameters

pEvt	Pointer to the IT_EVENT structure
pCtrl	The tab to become active. Refer to Specifying control [Page 151]

Return Value

Returns TRUE on success, FALSE on error.

See Also

ItEv_FindControl [Page 154].

ItEv_SupportFlags

Use

Allows you to set flags for the connection.

Currently only one flag is available, and setting it enables the following features:

- Ability to handle graphics on the screens of the R/3 transaction
- Uploading and downloading of data to and from files on the client

Syntax

```
DWORD DLEX ItEv_SupportFlags(HANDLE hMr, DWORD flgs);
```

Parameters

hMr	Connection handle
flgs	Currently the only *flgs* value is: **SAPGUI_SUPPORT_GRAPHICS** Using this *flgs* value enables handling of graphics in an R/3 screen, and it enables the downloading and uploading of data between an R/3 screen and a local file. This *flgs* value is available with SAPGUI of releases 4.5B and higher only.

Return Value

Returns TRUE on success, FALSE on error.

Comments

SAPGUI Front must be running. Use this function after successfully logging in and after SAPGUI Front has started.

GUI Component: ActiveX/OLE Automation

The GUI Component interface allows SAP Automation GUI to be run from any application that works as an OLE Automation controller, including Office 97, Visual Basic, and Lotus Notes.

The GUI Component allows you to access the data stream sent between the R/3 application server and the R/3 SAPGUI in COM-compliant applications. Your client programs can either replace the SAPGUI, or they can work alongside the SAPGUI screens.

Relationship to the GUI Library

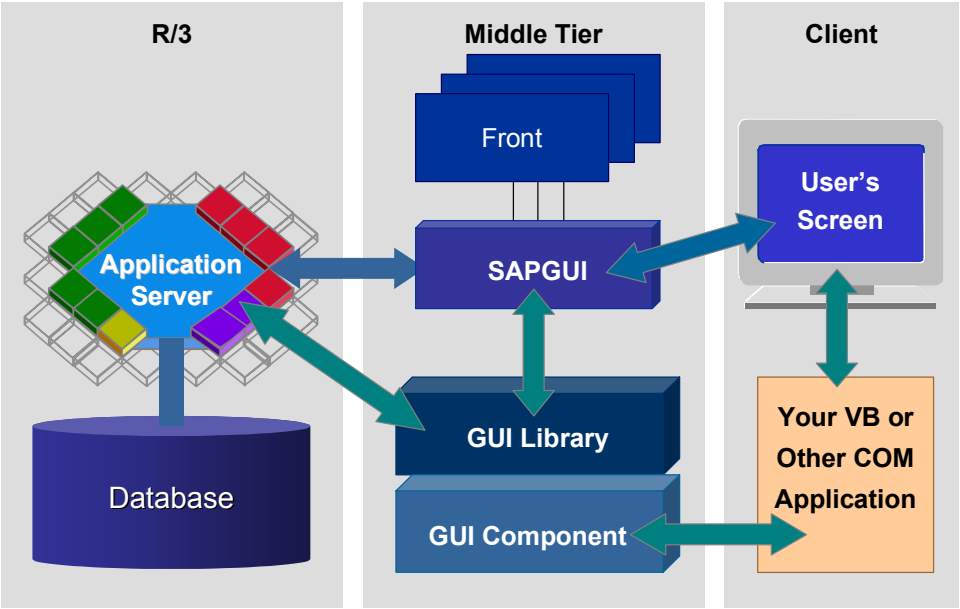
The GUI Component interface is layered on top of the SAP Automation GUI Library. It allows your VB or other COM-compliant applications to perform similar functions to that of the GUI Library.

The GUI Component allows a SAP system to be controlled directly from OLE Automation controller programs such as Office 97, Visual Basic, and Lotus Notes on Windows 95 and Windows NT systems.

Nearly all of the GUI Library calls have corresponding calls in the GUI Component. The GUI Component also provides greater error checking as well as many convenience routines for ease of programming within Visual Basic and similar environments.

The GUI Component interface is available both as an out-of-process EXE server and as an OLE control (OCX).

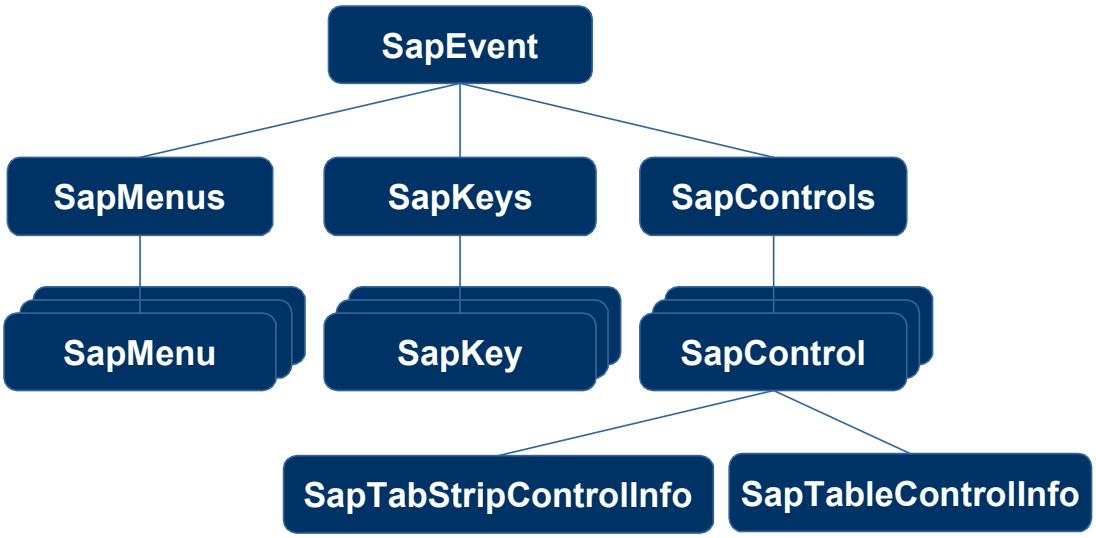
The following diagram shows the relationship of the GUI Component to the GUI Library and to your VB or other COM-compliant.



GUI Component Objects

GUI Component Object Hierarchy

The following diagram shows the hierarchy of the GUI Component objects:



Relationship to GUI Library Structures

The GUI Component objects correspond to the structures in the GUI Library [Page 30].

The top-level OLE Automation object is the SapEvent object. It is the one creatable object in the interface, and it corresponds to the GUI Library's IT_EVENT [Page 76] structure.

An Event contains a collection of Controls, a collection of Keys, and a collection of Menus. These collections correspond to the IT_SCREEN [Page 84], IT_PFKEYS [Page 94], and IT_MENUS [Page 97] structures.

Individual Control, Key, and Menu objects correspond to IT_CTRL [Page 87], IT_PFKEY [Page 95], and IT_MENU [Page 98] structures. Table controls have a TableControlInfo subtype that corresponds to the IT_TABLEINFO [Page 100] structure in the GUI Library. Tab strip controls have a TabStripControlInfo subtype that corresponds to the IT_TABSTRIPINFO [Page 103] structure.

Five enumerated types are also provided: SapControlType Enumeration [Page 272], SapDirection Enumeration [Page 273], SapGetType Enumeration [Page 274], SapGuiFlags Enumeration [Page 275], and SapSelectionType Enumeration [Page 276].

GUI Component Object Properties and Methods

The properties and methods for each of the OLE Automation objects are available from both the EXE and OCX unless otherwise noted.

All properties are read-only unless otherwise noted.

Referring to GUI Components Objects

The following table describes how to refer to the SapEvent object (the top-level object in the GUI Component hierarchy):

In:	Refer to SapEvent as:
EXE	“SapAutoGui.Event.”
OCX	"SapAutoGui.Control.1"

Connecting to R/3

Use

To use any functionality of the R/3 application server, you must first connect to R/3.

Procedure

3. Establish a connection to R/3, by using the Connect [Page 203], the NewGroupConnection [Page 220], or the NewServerConnection [Page 221] methods of SapEvent.

   You can optionally specify that the standard SAPGUI is to be displayed. In this case, a SAPGUI and one Front are invoked.

4. Log onto the SAP system, by using the Logon method [Page 219] of SapEvent.
5. Logoff from the R/3 system when done.

You can handle up to six sessions in the connection. See the details in the topic Using Multiple Sessions [Page 183].

You can handle multiple connections by calling the connection functions multiple times. You may use up to 99 connections.

See Also

Connecting to R/3 when using the GUI Library, [Page 39]Connection flags [Page 275], and the equivalent GUI Library connection flags [Page 128]

Using Multiple Sessions

Use

You can use up to six sessions for every R/3 connection you have. The first session is always created when you establish a connection to the R/3 application server.

You can only have one connection is SAPGUI Front is running.

The following procedure shows how to handle multiple R/3 sessions.

Procedure

1. Ask for a new connection to R/3, by using the Connect method [Page 203] of SapEvent.

   You can optionally specify that the standard SAPGUI is to be displayed. In this case, a SAPGUI and one Front are invoked.

2. Log onto the SAP system, by using the Logon method [Page 219] of SapEvent.

3. Handle the event of a new session being created, by assigning the new session object to a variable.

   The new session can be created as a result of an end user asking for it, by using the menu option *System→Create session*, for example.

Example

The following example handles two sessions.

The main program handles the connection to R/3 and handles the first session. The first session is in myEvt. In the first session, the program invokes a specific transaction, namely transaction *bibs*.

The second session is in sap2. In the second session, the program invokes another R/3 transaction, namely transaction *se38*.

```
Dim WithEvents myEvt As SapEvent
Public sap2 As SapEvent
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Private Sub ScreenCheck(ByVal ProgName As String, _
                        ByVal ScrName As String)
    If myEvt.ProgramName <> ProgName Or _
       (Len(ScrName) > 0 And myEvt.ScreenName <> ScrName) Then
        MsgBox "Unexpected screen " & myEvt.ProgramName & " " &
myEvt.ScreenName, vbCritical
        Stop
    End If
End Sub

Private Sub OKCheck(ByVal bIsOK As Boolean, ByVal sMsg As String)
  If Not bIsOK Then
    MsgBox sMsg, vbCritical
    Stop
  End If
```

```
End Sub

Sub Sessns()
  Dim iCtrl As Integer
  Dim bOK As Boolean
  'Create the GUI Component object:
  ' When using the OCX, use the following line
  Set myEvt = CreateObject("SapAutoGui.Control.1")

  'Connect to R/3
  '  Replace host name with the actual value:
  bOK = myEvt.Connect("myhost", "sysnum", SapGuiMerlin Or SapGuiFront
Or SapGuiFullMenu)
  OKCheck bOK, "Error in opening connection"
  'Set the dimensions of the SAPGUI window
  myEvt.RowDimension = 24
  myEvt.RowListDimension = 24
  myEvt.ColumnDimension = 80
  myEvt.ColumnListDimension = 80
  myEvt.SetSizeFlag = True
  OKCheck bOK, "Error in opening connection"
  ' Logon to the first session
  ' Replace client username and password with actual values
  bOK = myEvt.Logon("client", "user", "passwd", "en")
  OKCheck bOK, "Error in logon"

  myEvt.OKCode = "bibs" ' go to transaction bibs
  bOK = myEvt.SendEvent
  OKCheck bOK, "Error in sending default key"
  ScreenCheck "SAPMBIBS", "0100"
  ' Create a new session as a user would
  bOK = myEvt.SendMenuName("System")    ' index 24
  OKCheck bOK, "Error in sending menu"
  bOK = myEvt.SendMenuName("Create Session")    ' index 44
  OKCheck bOK, "Error in sending menu"

  'The second session
  Sleep (1500)
  If Not sap2 Is Nothing Then
    bOK = sap2.GetEvent()
    sap2.OKCode = "se38" 'go to transaction SE38
    bOK = sap2.SendEvent
  End If
End Sub
Private Sub myEvt_OnNewSession(ByVal sapEvt As Object)
    Set sap2 = sapEvt
End Sub
```

To log off the R/3 system, use the following code:

```
' Use the next line for standalone SAP program
' myEvt.Quit
bOK = myEvt.Logoff()
```

See Also

Handling multiple sessions when using the GUI Library [Page 40]

GUI Component Application Examples

The GUI Component makes it easy to use SAP Automation GUI from other Windows applications that can serve as OLE controllers. Here we show two examples:

- Driving the SAP R/3 System from a specially formatted Excel spreadsheet, using Visual Basic for Applications
- Re-creating R/3 screens within the Visual FoxPro environment

Spreadsheet Playback from Excel

Procedure

1. When starting work with the SAP Automation GUI Component, you would need to perform the following two preliminary operations:
   - Connecting to an SAP system
   - Logging onto an SAP system
2. After logging onto an R/3 system, there are only five basic operations for driving the SAP Automation GUI Component:
   - Setting the values of text or matchcode fields
   - Selecting and deselecting check box or radio button fields
   - Setting the OK code
   - Positioning the cursor over a user interface control (usually a push button or matchcode field)
   - Sending an event with an optional key, menu, or other event setting
3. To finish working with the GUI Component perform the following two closing operations:
   - Logging off from an SAP system
   - Quitting the SAP Automation GUI session

Example

The following example plays back Excel spreadsheets that have been created or recorded using a simple format.

The initial column in a row represents one of the nine operations listed above.

The additional columns in the row represent the data values being set.

For simplicity, controls are represented by their index on the screen. Keys are represented by their Windows Virtual-Key value.

```
Option Explicit

' Interpret an encoded Excel spreadsheet as an SAP session
' The first column in the row determines the operation, and
' the remaining columns are arguments.

' Operations:
' C: connect
' L: logon
' T: set text (or matchcode) fields
' X: set check boxes or radio buttons
' O: set OK code
' P: position cursor to a control
' K: send a key
' M: send a menu
```

Spreadsheet Playback from Excel

```
' F: logoff
' Q: quit

Sub SapSheet()
  Dim Sap As Object          ' Dim as SapEvent if using Excel 97
  Dim nRow As Integer
  Dim nCol As Integer
  Dim Ctrl As Integer
  Dim CtrlVal As String
  Dim OkCode As String
  Dim KeyNum As Integer
  Dim OnOff As Boolean
  Dim HostName As String     ' for C
  Dim SystemNumber As String ' for C
  Dim Client As String       ' for L
  Dim UserID As String       ' for L
  Dim Password As String     ' for L
  Dim Language As String     ' for L
  Dim MenuName As String     ' for M
  Dim OK As Boolean

  ' Start with the first cell in the speadsheet.

  nRow = 1
  nCol = 1

  ' Start the OLE Automation server.

  Set Sap = CreateObject("SapAutoGui.Event")

  ' Loop through the speadsheet until we find an empty entry
  ' in column 1.  Ignore rows with unrecognized letters or
  ' numbers in column 1.

  Do
    Cells(nRow, 1).Show
    Select Case Cells(nRow, 1)

      Case "C"    ' connect
        HostName = Cells(nRow, 2)
        SystemNumber = Cells(nRow, 3)
        OnOff = Cells(nRow, 4)
        If OnOff Then
          OK = Sap.Connect(HostName, SystemNumber, _
                           SapGuiFront)
        Else
          OK = Sap.Connect(HostName, SystemNumber, 0)
        End If
        If Not OK Then
          MsgBox "Could not connect to " & HostName & " " &
SystemNumber
        End If
```

```
Case "L"    ' logon
  Client = Cells(nRow, 2)
  UserID = Cells(nRow, 3)
  Password = Cells(nRow, 4)
  Language = Cells(nRow, 5)
  Sap.Logon Client, UserID, Password, Language

Case "T"    ' set text/matchcode data
  nCol = 2

  ' Loop through the row for pairs of control indices and
  ' new text values.  Stop when we find an empty column where
  ' we expect to find a control index (empty values are OK).

  Do
    If IsEmpty(Cells(nRow, nCol)) Then
      Exit Do
    End If
    Ctrl = Cells(nRow, nCol)
    If IsEmpty(Cells(nRow, nCol + 1)) Then
      CtrlVal = ""
    Else
      CtrlVal = Cells(nRow, nCol + 1)
    End If
    Sap.Controls(Ctrl) = CtrlVal
    nCol = nCol + 2
  Loop

Case "X"    ' set check box or radio button data
  nCol = 2

  ' Loop through the row for pairs of control indices and
  ' new button values (1 for on, 0 or anything else for off).
  ' Stop when we find an empty column.

  Do
    If IsEmpty(Cells(nRow, nCol)) Then
      Exit Do
    End If
    Ctrl = Cells(nRow, nCol)
    If IsEmpty(Cells(nRow, nCol + 1)) Then
      Exit Do
    End If
    OnOff = (Cells(nRow, nCol + 1) = 1)
    Sap.Controls(Ctrl).Selected = OnOff
    nCol = nCol + 2
  Loop

Case "K"    ' send key

  ' If column 2 has a value, use it as the virtual-key value to
  ' send.  Otherwise send the default key.
```

Spreadsheet Playback from Excel

```
        If Not IsEmpty(Cells(nRow, 2)) Then
          KeyNum = Cells(nRow, 2)
          Sap.SendKey (KeyNum)
        Else
          Sap.SendEvent
        End If

      Case "M"    ' send menu

        ' If column 2 has a value, use it as the menu name to
        ' send.  Otherwise ignore this line.

        If Not IsEmpty(Cells(nRow, 2)) Then
          MenuName = Cells(nRow, 2)
          Sap.SendMenuName (MenuName)
        End If

      Case "O"    ' set OK code
        OkCode = Cells(nRow, 2)
        Sap.OkCode = OkCode

      Case "P"    ' position cursor to a control
        Ctrl = Cells(nRow, 2)
        Sap.SetCursorByControl Ctrl

      Case "F"    ' logoff
        Sap.Logoff

      Case "Q"    ' quit
        Sap.Quit

      Case Empty  ' stop spreadsheet interpretation
        Exit Do

    End Select
    nRow = nRow + 1
    nCol = 1
    DoEvents
  Loop
End Sub
```

Screen Capture from Visual FoxPro

Use

One use of SAP Automation GUI is to be able to run R/3 transactions with redesigned user interfaces. Tools like Microsoft's Visual FoxPro and Visual Basic provide Windows 95 user interface development environments.

A starting point for such redesigns is to capture the original R/3 screen in a representation that can be used by the user interface development tool.

Example

The following example shows how to capture a particular screen as a Visual FoxPro form. For simplicity we have coded in the system name, the login ID, and the transaction code.

```
* Read a transaction and create a Visual FoxPro form.

* Values are hardcoded for a simple example of
* programmatic form construction.

* Define control types that we will be using - taken from guilib.h

#DEFINE CTRL_STATIC            1
#DEFINE CTRL_EDIT              2
#DEFINE CTRL_PASSWORD          3
#DEFINE CTRL_PUSHBUTTON        4
#DEFINE CTRL_RADIOBUTTON       5
#DEFINE CTRL_CHECKBOX          6
#DEFINE CTRL_FRAMEBOX          7
#DEFINE CTRL_MATCH             9
#DEFINE CTRL_LISTSTATIC       10
#DEFINE CTRL_MATCHFIX         12

* Define vertical and horizontal scales and margins.

#DEFINE SAPVSCALE 1.25
#DEFINE SAPHSCALE 1.25
#DEFINE SAPVMARG 0.5
#DEFINE SAPHMARG 0.5
#DEFINE SAPTB 1.5

CLEAR

* Start the GUI Component server.

Sap = CREATEOBJECT ("SapAutoGui.Event")

* Connect to the SAP system and logon.  Change the arguments
* to those of a real system to have this work.

Sap.Connect ("mysys.sfo.sap-ag.de", "00", 0)
Sap.Logon ("000", "myuser", "mypass", "e")
```

```
* For this example, call the FBL1 transaction.

Sap.Transaction ("FBL1")

* Now build the form.

SapForm = CREATEOBJECT ("Form") && Create the form
SapForm.ScaleMode = 0           && Set units to Foxels
SapForm.Caption = Sap.Title     && Set form title

* Set width and height using dimensions of the current event.  Scale
* for Visual FoxPro and include room for margins and the FoxPro menu.

SapForm.Height = Sap.RowDimension * SAPVSCALE + (2 * SAPVMARG) + SAPTB
SapForm.Width = Sap.ColumnDimension * SAPHSCALE + (2 * SAPHMARG)

* Comment the next line out if you do not want to see the form as it
* is being built.

SapForm.Visible =.T.

* Loop through each control in the screen and add it to the form if
* we recognize the control type.  Skip any unrecognized controls.

CtrlLast = Sap.Controls.Count - 1
For i = 0 to CtrlLast

  && Cache values so we avoid redundant OLE Automation calls.

  Ctrl = Sap.Controls(i)
  CtrlType = Ctrl.Type
  CtrlTop = Ctrl.Top
  CtrlLeft = Ctrl.Left
  CtrlHeight = Ctrl.Height
  CtrlWidth = Ctrl.Width
  CtrlFieldName = ""

  && Provide unique names for the objects in the form.  We need
  && them to identify the controls as we build the form.

  SapName = "SapObj" + LTRIM(STR(i))
  SapLabel = ""  && Used as part of simulating frame boxes
  SapIgnore =.F.

  Do Case

    Case CtrlType = CTRL_STATIC
      SapForm.AddObject ((SapName), "Label")
      SapForm.&SapName..Caption = Ctrl.Value
      SapForm.&SapName..AutoSize =.T.  && Size to fit the label

    Case CtrlType = CTRL_EDIT
```

```
  SapForm.AddObject ((SapName), "TextBox")
  SapForm.&SapName..Value = LTRIM(Ctrl.Value)
  SapForm.&SapName..SpecialEffect = 0  && 3D

  && For edit and some other fields, set the tooltip text
  && to the table and field names.

  If Ctrl.HasFieldNames Then
    CtrlFieldName = RTRIM(Ctrl.Name)
    SapForm.&SapName..ToolTipText = CtrlFieldName
  EndIf

Case CtrlType = CTRL_PASSWORD
  SapForm.AddObject ((SapName), "TextBox")
  SapForm.&SapName..Value = LTRIM(Ctrl.Value)
  SapForm.&SapName..PasswordChar = "*"  && For password text
  SapForm.&SapName..SpecialEffect = 0   && 3D
  If Ctrl.HasFieldNames Then
    CtrlFieldName = RTRIM(Ctrl.Name)
    SapForm.&SapName..ToolTipText = CtrlFieldName
  EndIf

Case CtrlType = CTRL_PUSHBUTTON
  SapForm.AddObject ((SapName), "CommandButton")
  SapForm.&SapName..Caption = Ctrl.Value

Case CtrlType = CTRL_RADIOBUTTON
  SapForm.AddObject ((SapName), "OptionButton")
  SapForm.&SapName..Caption = Ctrl.Value
  SapForm.&SapName..SpecialEffect = 0  && 3D
  SapForm.&SapName..Value = IIF (Ctrl.Selected,.T.,.F.)

Case CtrlType = CTRL_CHECKBOX
  SapForm.AddObject ((SapName), "CheckBox")
  SapForm.&SapName..Caption = Ctrl.Value
  SapForm.&SapName..SpecialEffect = 0  && 3D
  SapForm.&SapName..Value = IIF (Ctrl.Selected,.T.,.F.)

Case CtrlType = CTRL_FRAMEBOX
  && Visual FoxPro does not seem to have a frame box, so combine
  && a box and label to simulate it.
  SapForm.AddObject ((SapName), "Shape")
  SapForm.&SapName..Curvature = 0      && Rectangular
  SapForm.&SapName..SpecialEffect = 0  && 3D
  SapLabel = "SapLabel" + LTRIM(STR(i))
  SapForm.AddObject ((SapLabel), "Label")
  SapForm.&SapLabel..Caption = RTRIM(Ctrl.Value)
  SapForm.&SapLabel..AutoSize =.T.

Case CtrlType = CTRL_MATCH
  SapForm.AddObject ((SapName), "ComboBox")
  SapForm.&SapName..AddItem (LTRIM(Ctrl.Value))
  SapForm.&SapName..ListIndex = 1
```

```
      SapForm.&SapName..Style = 0  && Dropdown combo allows text entry
      SapForm.&SapName..SpecialEffect = 0  && 3D
      CtrlWidth = CtrlWidth + 2    && Allow room for the arrow
      If Ctrl.HasFieldNames Then
        CtrlFieldName = RTRIM(Ctrl.Name)
        SapForm.&SapName..ToolTipText = CtrlFieldName
      EndIf

    Case CtrlType = CTRL_MATCHFIX
      SapForm.AddObject ((SapName), "ComboBox")
      SapForm.&SapName..AddItem (LTRIM(Ctrl.Value))
      SapForm.&SapName..ListIndex = 1
      SapForm.&SapName..Style = 2  && Dropdown list - no text entry
      SapForm.&SapName..SpecialEffect = 0  && 3D
      CtrlWidth = CtrlWidth + 2    && Allow room for the arrow
      If Ctrl.HasFieldNames Then
        CtrlFieldName = RTRIM(Ctrl.Name)
        SapForm.&SapName..ToolTipText = CtrlFieldName
      EndIf

    Case CtrlType = CTRL_LISTSTATIC
      SapForm.AddObject ((SapName), "Label")
      SapForm.&SapName..Caption = Ctrl.Value

    Otherwise
      SapIgnore =.T.  && ignore for now

  EndCase

  && Set location, size, and visibility for all control types.
  && Controls use scales and margins in the same way we used them
  && to set the overall size of the form.

  If !SapIgnore

    && When simulating frame boxes, shift the box down and the label
    && to the right.

    If LEN(SapLabel) > 0
      SapForm.&SapName..Top = (CtrlTop + 0.33) * SAPVSCALE + SAPVMARG
      SapForm.&SapName..Left = CtrlLeft * SAPHSCALE + SAPHMARG
      SapForm.&SapName..Height = CtrlHeight * SAPVSCALE
      SapForm.&SapName..Width = CtrlWidth * SAPHSCALE
      SapForm.&SapName..Visible =.T.
      SapForm.&SapLabel..Top = CtrlTop * SAPVSCALE + SAPVMARG
      SapForm.&SapLabel..Left = (CtrlLeft + 1) * SAPHSCALE + SAPHMARG
      SapForm.&SapLabel..Height = SAPVSCALE  && Just 1 unit
      SapForm.&SapLabel..Visible =.T.
    Else
      SapForm.&SapName..Top = CtrlTop * SAPVSCALE + SAPVMARG
      SapForm.&SapName..Left = CtrlLeft * SAPHSCALE + SAPHMARG
      SapForm.&SapName..Height = CtrlHeight * SAPVSCALE
      SapForm.&SapName..Width = CtrlWidth * SAPHSCALE
```

```
      SapForm.&SapName..Visible =.T.
    EndIf
  EndIf

EndFor

SapForm.Visible =.T.  && Make form visible when done

* Wait for a click to log off from the SAP System.

Wait Window "Click to logoff from SAP"
Sap.Logoff
Sap.Quit

* Wait for another click to finish the program, saving
* to a predefined file name.

Wait Window "Click to finish program"
SapForm.SaveAs ("SAP.SCX")
```

GUI Component Reference

SapEvent Object

SapEvent Properties

The following table summarizes the properties of the SapEvent object.

Property	Type	Description
Application	SapEvent	Returns this event object. (EXE only).
Client	BSTR	SAP client number for this session.
ColumnDimension	short	Number of columns in a dynpro screen. Read/write.
ColumnListDimension	short	Number of columns in a list screen. Read/write (currently write-only).
Connected	BOOL	TRUE if connected to an SAP server; FALSE otherwise.
Controls	SapControls [Page 263]	Controls collection for this event.
CPU	BSTR	Name of CPU running current session.
CursorColumn	short	Column of current cursor position. Read/write.
CursorRow	short	Row of current cursor position. Read/write.
Database	BSTR	Name of database of current session.
DataColumns	short	Columns of underlying data in step loop or list.
DataColumnScreenSize	short	Number of columns of data that are shown on screen in step loop or list.
DataColumnStart	short	Starting data column in step loop or list.
DataRows	short	Rows of underlying data in step loop or list.
DataRowScreenSize	short	Number of rows of data that are shown on screen in step loop or list.
DataRowStart	short	Starting data row in step loop or list.
DiagVersion	short	Version of DIAG software.
DialogDismissedFlag	BOOL	TRUE if dialog dismissed flag is set.
DynproInfoFlag	BOOL	TRUE if ProgramName and ScreenName properties are available for this event.
EndOfSessionFlag	BOOL	TRUE if end of session flag is set, indicating that session is being disconnected.
FullName	BSTR	File specification of the application. EXE only.
GetAllMenus	BOOL	TRUE if all submenus are fetched at once. Read/write.

GetNotSend	BOOL	When doing manual GetEvent calls, GetEvent and SendEvent calls generally must alternate. GetNotSend is TRUE if the server is expecting a GetEvent next; FALSE if it is expecting a SendEvent.
HowToGet	SapGetType [Page 274]	Set automatic or manual GetEvent calls. Read/write.
KeyToSend	long	The Windows Virtual-Key value for the key to be sent. Read/write.
Keys	LPDISPATCH	Keys collection for this event.
MenuEntriesFlag	BOOL	TRUE if this event is a menu message.
MenuFlag	BOOL	TRUE if menu structure is active.
Menus	SapMenus [Page 253]	Menus collection for this event.
MenuToSend	long	Menu number to be sent. Read/write.
Message	BSTR	SAP message. In the SAPGUI, this is displayed in the status line. Use of messages differs across R/3 applications, as does the use of indications for informational, warning, or error messages. Sometimes messages will appear in modal dialog boxes, not just here.
MessageFlag	BOOL	TRUE if event contains a message.
MessageHelpFlag	BOOL	TRUE if message help will be requested. Read/write.
ModalHeight	short	Height of modal dialog box, in character units. 0 if not modal.
ModalLeft	short	Leftmost position of modal dialog box, in character units. 0 if not modal.
ModalTop	short	Topmost position of modal dialog box, in character units. 0 if not modal.
ModalWidth	short	Width of modal dialog box, in character units. 0 if not modal.
ModeNumber	short	Mode or session number.
Name	BSTR	Name of the application. EXE only. Default property for EXE.
OKCode	BSTR	OK Code to be sent to the application. Read/write.
OKCodeFlag	BOOL	TRUE if received event contains an OK Code.

SapEvent Properties

OKHelpFlag	BOOL	TRUE if help for OK Code box will be requested. Read/write.
Parent	LPDISPATCH	For EXE, returns this SapEvent. For OCX, returns OLE control container.
PFKeyFlag	BOOL	TRUE if event contains keys information.
ProgramName	BSTR	Name of current SAP program. Only available if DynproInfoFlag property is TRUE.
R2	BOOL	TRUE if connected to an R/2 system via the CUA gateway.
RfcFlag	BOOL	TRUE if RFC flag is on.
RowDimension	short	Number of rows in a dynpro screen. Read/write.
RowListDimension	short	Number of rows in a list screen. Read/write (currently write-only).
ScreenFlag	BOOL	True if the event contains a screen with controls information.
ScreenModal	long	1 if screen is a modal dialog box; 0 otherwise.
ScrollColumnToSend	long	New starting data column for horizontal scroll. Read/write.
ScrollRowToSend	long	New starting data row for vertical scroll. Read/write.
ScreenName	BSTR	Name of current SAP screen. Only available if DynproInfoFlag property is TRUE.
SendMenuFlag	BOOL	TRUE if menu event will be sent. Automatically set when MenuToSend property is set, but may also be set manually. Read/write.
SendOKCodeFlag	BOOL	TRUE if OK code will be sent. Automatically set when OKCode property is set, but may also be set manually. Read/write.
SendPFKeyFlag	BOOL	TRUE if PF key will be sent. Automatically set when KeyToSend property is set, but may also be set manually. Read/write.
SetCursorPositionFlag	BOOL	TRUE if new cursor position will be sent. Automatically set when CursorColumn or CursorRow property is set, but may also be set manually. Read/write.
SetSizeFlag	BOOL	TRUE if screen size will be reset. Automatically set when RowDimension, RowListDimension, ColumnDimension, or ColumnListDimension properties are set, but may also be set manually. Read/write.
Size	long	Size of current event in bytes.

StaticInfoFlag	BOOL	TRUE if static information (CPU, Database, DiagVersion, ModeNumber) is available.
TerminalClearFlag	BOOL	TRUE if terminal clear flag is set.
Title	BSTR	Title of screen as it appears in the Windows title bar.
TitleFlag	BOOL	TRUE if Title property contains screen title.
Username	BSTR	SAP user name for this session.
V3	BOOL	TRUE if session is connected to an SAP System running version 3.0 or later.
Visible	BOOL	TRUE if the OLE Automation server user interface is visible on the screen. Read/write. EXE only.

See Also

SapEvent Methods [Page 202]

SapEvent Events [Page 247]

SapEvent Methods

Connect

Syntax

```
BOOL Connect (LPCTSTR HostName, LPCTSTR SystemNumber,
      SapGuiFlags Flags)
```

Parameters

HostName	Name of SAP System host; may be TCP/IP network identifier
SystemNumber	SAP System number
Flags [Page 275]	Flags to control various aspects of the connection operation. The same flags are used in all of the connection methods. Each of the flags is based on a GUI Library connection flag [Page 128].

Return Value

Returns TRUE on success, FALSE on error.

Comments

Connects to an SAP system using the given parameters. Can then be followed by Logon [Page 219] or Logoff [Page 218] methods. Supersedes OpenConnection [Page 223] method.

DisplaySapGui

```
BOOL DisplaySapGui (BOOL DisplayWindow)
```

Parameters

DisplayWindow	TRUE to display the SAPGUI window; FALSE to hide it

Return Value

Returns TRUE if the SAPGUI window is found; FALSE otherwise.

Comments

Hides or displays the current SAP GUI window, if it can be found. The ID of the SAPGUI window may change, so there is no guarantee that the window will remain hidden or displayed across events. It is generally better to hide or display the SAPGUI window at connect time by setting or clearing the SapGuiFront flag in the Flags parameter of the Connect [Page 203] method.

FindByField

```
long FindByField (LPCTSTR Field, long Start, SapControlType Type)
```

Parameters

Field	String containing the batch input name to look for (table name - field name).
Start	Index of control where search begins; 0 to search all controls.
Type	Type of control to search for. (See SapControlType Enumeration [Page 272].)

Return Value

Index of the first control satisfying the search criteria; -1 if not found.

Comments

FindByField allows controls to be searched based on the underlying field in the SAP R/3 repository. Both the table name and field name are used. This provides for language-independent searches.

Starting with R/3 3.0C, field names and table names are provided by default with events. Prior to 3.0C, the GetEventFull [Page 215] method must be used to get field and table names. This requires an additional two server round trips for each edit and matchcode field. With 3.0C, field names are also available for other control types, such as check boxes and radio buttons.

FindByValue

```
long FindByValue (LPCTSTR Value, long Start,
      SapControlType Type, SapDirection Direction)
```

Parameters

Value	String containing the value of the field to search for.
Start	Index of control where search begins; 0 to search all controls.
Type	Type of control to search for. (See SapControlType Enumeration [Page 272].)
Direction	If non-zero, returns the control immediately to the right, left, up, or down from the control. (See SapDirection Enumeration [Page 273].) If used with the type field, the search will match a value of any control type, and then search in the given direction for the specified type.

Return Value

Index of the first control satisfying the search criteria; -1 if not found.

Comments

FindByValue searches based on the value names of the fields. This is the quickest method of searching for fields in R/3 Systems through version 3.0B, but the resulting code is language-dependent.

FindExtended

```
long FindExtended (LPCTSTR Value, long Start, long Amount, long Flags)
```

Parameters

Value	String containing the control value to search for.
Flags	Flags controlling the search. See Specifying the Flags parameter [Page 152] topic in the GUI Library.
Start	Starting point for the search. This overrides FC_FIND_STARTING_FROM in *flags*.
Amount	How many controls to search. This overrides FC_COUNT in *flags*.

Return Value

Index of the first control satisfying the search criteria; -1 if not found.

Comments

Searches are for the initial substring, and are not case-sensitive. The flags controlling the search are described in the ItEv_FindControl GUI Library call. FindExtended finds a string value, using either a value or field name. FindFromExtended finds controls from a certain starting point without a string value (i.e., it uses ITCTRL_IDX macro for the value argument to ItEv_FindControl).

See Also

FindFromExtended [Page 209], ItEv_FindControl [Page 154].

FindFrom

```
long FindFrom (long Start, SapControlType Type,
      SapDirection Direction, short Amount)
```

Parameters

Start	Index of starting control.
Type	Type of control to search for. (See SapControlType Enumeration [Page 272].)
Direction	Direction to search. (See SapDirection Enumeration [Page 273].)
Amount	Number of controls to search in given direction.

Return Value

Returns the index to the control located the specified amount away from the starting control index, searching in the given direction. Returns -1 if no such control exists.

Comments

FindFrom is commonly used to move through controls in step-loops that have neither distinct string values nor field names associated with them.

FindFromExtended

```
long FindFromExtended (long Start, long Amount, long Flags)
```

Parameters

Start	Index of starting control
Amount	How many controls to search. This overrides FC_COUNT in *flags*. See Specifying the Flags parameter [Page 152] topic in the GUI Library.
Flags	Flags to control search

Return Value

Index of the first control satisfying the search criteria; -1 if not found.

Comments

The flags controlling the search are described in the ItEv_FindControl GUI Library call. FindFromExtended finds controls from a certain starting point without searching for a value or field name (i.e., it uses ITCTRL_IDX macro for the value argument to ItEv_FindControl).

See Also

FindExtended [Page 207] ; ItEv_FindControl [Page 154].

FindKey

```
long FindKey (LPCTSTR Name)
```

Parameters

Name	Name of the key to search for

Return Value

Index (**not** Virtual-Key Value) of the first key with this name; -1 if not found.

Comments

Searches are for the initial substring, and are not case-sensitive.

FindMenu

```
long FindMenu (LPCTSTR Name)
```

Parameters

Name	Name of the menu item to search for

Return Value

Index of the first menu item with this name; -1 if not found.

Comments

Searches are not case-sensitive. All symbols, including ampersands and ellipses, are included in the search.

GetEvent

```
BOOL GetEvent ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Gets the current event. GetEvent checks the GetNotSend flag to avoid getting two events in a row without an intervening SendEvent [Page 225]. Done automatically after each SendKey [Page 228] or SendEvent when HowToGet = SapGetAutomatic (the default value).

GetEventAlways

```
BOOL GetEventAlways (long Flags)
```

Parameters

Flags	Flags to control how the event is retrieved. See the description of the *flgs* parameter to It_GetEventEx [Page 115] in the GUI Library for possible flag values.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Gets the current event, without checking the GetNotSend flag. In most cases, two consecutive GetEvent [Page 212] calls without an intervening SendKey [Page 228] or SendEvent [Page 225] can cause the system to hang waiting for a second event that will not come. In some cases, such as when monitoring rather than driving GUI usage with SAP Automation GUI, multiple consecutive GetEvent calls can be required. GetEventAlways is used then.

See Also

SendEventAlways [Page 226].

GetEventExtended

```
BOOL GetEventExtended (long Flags)
```

Parameters

Flags	Flags to control how the event is retrieved. See the description of the *flgs* parameter to It_GetEventEx [Page 115] in the GUI Library for possible flag values.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Allows for several options when getting an event manually. See It_GetEventEx in the GUI Library for further details. HowToGet must be set to SapGetManual in order to use GetEventExtended.

GetEventFull

```
BOOL GetEventFull ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Gets an event including extended screen information (program, screen, table, and field names, plus data element and data element supplement). Automatically done after each SendKey [Page 228] or SendEvent [Page 225] if HowToGet is set to 1.

GetEventFront

```
BOOL GetEventFront ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Gets an event from the SAPGUI FRONT.EXE program. Use this function to monitor changes in FRONT.EXE.

HasEvent

```
BOOL HasEvent ()
```

Parameters

None.

Return Value

TRUE if there is an event; FALSE otherwise.

Comments

When the HowToGet is set to SapGetManual, HasEvent() will return FALSE after a SendEvent [Page 225] or SendKey [Page 228] call until the next GetEvent* call is made.

Logoff

```
BOOL Logoff ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Logs off and disconnects from the current SAP session.

Logon

```
BOOL Logon (LPCTSTR Client, LPCTSTR UserID, LPCTSTR Password,
LPCTSTR Language)
```

Parameters

Client	String representing the client number (up to 3 characters long).
UserID	User ID for login.
Password	Password for login.
Language	Single-character string representing language to be used during the session (e.g., E for English, D for Deutsch (German).

Return Value

Returns TRUE on success, FALSE on error.

Comments

Logs onto the system using the given parameters. Dismisses the copyright screen and a "System Messages" screen, if any. Works for R/3 3.0C and later, or earlier R/3 logon screens in English or German.

NewGroupConnection

```
BOOL NewGroupConnection(LPCTSTR Id, LPCTSTR HostName, LPCTSTR Router,
LPCTSTR group, long Flags);
```

Parameters

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
hostname	Hostname of the message server. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one Message Server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the message server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
group	A group from a list of defined Groups in the system. This group is used to logon to an arbitrary application server in this Group. The group is based on load balancing. For example, group can be PUBLIC or JAPANESE, If the user does not know which group he wants to logon, he could use an empty string for this argument. The system will pick one automatically to make the connection.
Flags [Page 275]	Flags to control various aspects of the connection operation. The same flags are used in all of the connection methods. Each of the flags is based on a GUI Library connection flag [Page 128].

Return Value

Returns the handle to an SAP Automation GUI session if successful, otherwise returns FALSE.

Comments

This is used to connect with R3 using GROUP information (load balancing).

NewServerConnection

```
BOOL NewServerConnection (LPCTSTR Id, LPCTSTR HostName, LPCTSTR Router,
long Flags);
```

Parameters

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
hostname	Hostname of the message server. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one message server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the message server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
Flags [Page 275]	Flags to control various aspects of the connection operation. The same flags are used in all of the connection methods. Each of the flags is based on a GUI Library connection flag [Page 128].

Return Value

Returns the handle to an SAP Automation GUI session if successful, otherwise returns FALSE.

Comments

Connects to R/3 application server. This method connects to the first application server in server list.

NewServerConnectionEx

```
BOOL NewServerConnectionEx(LPCTSTR Id, LPCTSTR HostName, LPCTSTR
Router, LPCTSTR Server, long Flags);
```

id	SAP System ID that identifies the SAP System. The list of SAP Systems is retrieved from the file SAPMSG.INI.
hostname	Hostname of the message server. This message server will provide a list of the currently available application servers that are running on the selected system. Each SAP System provides one Message Server. This information is stored in the file SAPMSG.INI.
router	Destination router used to connect to the message server as well as to the listed application servers. The list of available SAP routers is retrieved from the file SAPROUTE.INI.
server	The IP address of server to connect to.
Flags [Page 275]	Flags to control various aspects of the connection operation. The same flags are used in all of the connection methods. Each of the flags is based on a GUI Library connection flag [Page 128].

Return Value

Returns the handle to an SAP Automation GUI session if successful, otherwise returns FALSE.

Comments

Allows you to connect to a specific R/3 application server.

OpenConnection

```
BOOL OpenConnection (LPCTSTR HostName, LPCTSTR SystemNumber,
BOOL RunFront, BOOL RunMerlin)
```

Parameters

HostName	Name of SAP system host; may be TCP/IP network identifier.
SystemNumber	SAP system number.
RunFront	TRUE to run FRONT.EXE and see the SAP GUI; FALSE to hide the SAP GUI.
RunMerlin	TRUE to run an SAP Automation GUI session; FALSE otherwise.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Retained for compatibility with older version of SAP Automation GUI. Use the Connect [Page 203] method instead.

Quit

```
void Quit ()
```

Parameters

None.

Return Value

None.

Comments

Quits the OLE Automation EXE server. If still connected, logs off and disconnects from the current session. If using the OLE control, only the logoff and disconnection occurs.

SendEvent

```
BOOL SendEvent ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server.

SendEventAlways

```
BOOL SendEventAlways ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server, without checking the GetNotSend flag. In most cases, two consecutive SendEvent [Page 225] calls without an intervening GetEvent [Page 212] can cause system synchronization errors. In some cases, though, multiple consecutive SendEvent calls can be required, and SendEventAlways is used then.

See Also

GetEventAlways [Page 213].

SendEventToFront

```
BOOL SendEventTOFront ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the SAPGUI Front.

SendKey

```
BOOL SendKey (long VKValue)
```

Parameters

VKValue	Windows Virtual-Key code for key to send

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server, after setting the KeyToSend property to the specified virtual-key value.

SendKeyName

```
BOOL SendKeyName (LPCTSTR KeyName)
```

Parameters

KeyName	Key name to search for

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server, after setting the KeyToSend property to the first key with the name specified by the KeyName parameter. Searches are for the initial substring, and are not case-sensitive.

SendMenu

```
BOOL SendMenu (long MenuIndex)
```

Parameters

MenuIndex	Index of menu to send

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server, after setting the MenuToSend property to the specified menu index.

SendMenuName

```
BOOL SendMenuName (LPCTSTR MenuName)
```

Parameters

MenuName	Menu name to search for

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server, after setting the MenuToSend property to the first menu item with the name specified by the MenuName parameter. Searches are not case-sensitive. All symbols, including ampersands and ellipses, are included in the search.

SendMessageHelp

```
BOOL SendMessageHelp ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server after setting the MessageHelpFlag property to TRUE.

SendOKHelp

```
BOOL SendOKHelp ()
```

Parameters

None.

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends an event to the application server after setting the OKHelpFlag property to TRUE.

SendScrollColumn

```
BOOL SendScrollColumn (long Column)
```

Parameters

Column	New starting column for horizontal scroll

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends a horizontal scroll event to the application server, after setting the ScrollColumnToSend property to the specified column.

SendScrollRow

```
BOOL SendScrollRow (long Row)
```

Parameters

Row	New starting row for vertical scroll

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sends a vertical scroll event to the application server, after setting the ScrollRowToSend property to the specified row.

SendTabButton

```
BOOL SendTabButton (long Index)
```

Parameters

Index	Index of a tab button

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sets the tab button indexed by the Index parameter to active, sends the event to R/3 server and refreshes the event (if the tab button is a server tab button, a new tab strip page is loaded).

SetControlSelected

```
BOOL SetControlSelected (long Index, BOOL Selected)
```

Parameters

Index	Index of control that is being selected or de-selected.
Selected	TRUE if the control (a radio button or check box) is to be selected, FALSE if it is to be de-selected.

Return Value

Returns TRUE on success, FALSE on error.

Comments

A shortcut for Controls.Item(Index).Selected = Selected, for optimizing OLE Automation usage.

SetControlValue

```
BOOL SetControlValue (long Index, LPCSTR Value)
```

Parameters

Index	Index of control that is having its value set
Value	The new value of the control, represented as a text string

Return Value

Returns TRUE on success, FALSE on error.

Comments

A shortcut for Controls.Item(Index).Value = Value, for optimizing OLE Automation usage.

SetCursorByControl

```
BOOL SetCursorByControl (long Index)
```

Parameters

Index	Index of control used to set the current cursor position

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sets the current cursor position to the top left corner of the control specified by the Index parameter.

SetKeyByName

```
BOOL SetKeyByName (LPCTSTR KeyName)
```

Parameters

KeyName	Key name to search for

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sets the KeyToSend property to the first key with the name specified by the KeyName parameter. Searches are for the initial substring, and are not case-sensitive.

SetMenuByName

```
BOOL SetMenuByName (LPCTSTR MenuName)
```

Parameters

MenuName	Menu name to search for

Return Value

Returns TRUE on success, FALSE on error.

Comments

Sets the MenuToSend property to the first menu item with the name specified by the MenuName parameter. Searches are not case-sensitive. All symbols, including ampersands and ellipses, are included in the search.

StartGui

```
BOOL StartGui ()
```

Return Value

Returns TRUE on success, FALSE on error.

Comments

Starts the SAPGUI FRONT.EXE program and displays the current event.

StopGui

```
BOOL StopGui ()
```

Return Value

Returns TRUE on success, FALSE on error.

Comments

Stops the SAPGUI FRONT.EXE program.

SupportFlags

```
Bool SupportFlags (long Flags)
```

Parameters

Flags	The only value that is currently available is SapGraphicsSupport. Using this *Flags* value enables handling of graphics in an R/3 screen by the GUI Component, and it enables the downloading and uploading of data between an R/3 screen and a local file.

Return Value

Returns TRUE on success, FALSE on error.

Comments

This is similar to using the ItEv_SupportFlags [Page 178] in the GUI Library.

This feature is available with SAPGUI of releases 4.5B and higher only.

TableEntry

```
long TableEntry (long Table, long Column, long Row)
```

Parameters

Table	Index of SapTable control containing table entry.
Column	Column number of desired table entry. This applies to row selector and fixed columns.
Row	Row number of desired table entry. The number of the first data row in a table is 0.

Return Value

Returns the index of the control that is in the appropriate column and row within the given SapTable control. Returns -1 if no such control found.

Comments

Column and row numbers are based on the order in which the SAP application sends table data to the SAP presentation server, and may not match the layout that appears on the screen if data has been rearranged.

Transaction

```
BOOL Transaction (LPCTSTR Tcode)
```

Parameters

Tcode	Transaction code to start

Return Value

Returns TRUE on success, FALSE on error.

Comments

Starts a transaction with the given transaction code.

SapEvent Events

The OLE control version of the SAP Automation GUI Component provides the following OLE events to its control container.

OnLogoff [Page 249]

OnNewEvent [Page 250]

OnNewSession [Page 251]

OnDelSession [Page 248]

These events allow the container to respond when a new event has been received in response to a GUI Component call.

This is useful, for instance, in allowing a container program to replay macros and drive a reconstructed version of the SAP interface based on the incoming SAP event data.

OnDelSession

This event is fired when an event that closes a SAP GUI session is received.

It is equivalent to the GUI Library's It_SetDelSessionHook [Page 143] request.

This event is applicable for both the OCX and the EXE versions.

OnLogoff

This OLE event is fired when an SAP event is received with the End of Session flag turned on. Thus it will be fired both after the Logoff [Page 218] method is called, and after sending an event that results in logging off from the SAP system.

OnNewEvent

This OLE event is fired after an SAP event method has received the last SAP event associated with the method.

For instance, a call to the Logon [Page 219] method will fire this event only once. The Logon method will not fire this event when it receives the SAP events for the copyright and system messages screens, which it automatically dismisses as part of the method call.

Other methods that may receive multiple events in a single method call (e.g., GetEventExtended [Page 214]) work in the same way.

This event is also not fired when an End of Session event is received. In that case, the OnLogoff [Page 249] event is fired instead.

OnNewSession

This event is fired when a new SAP GUI session event is received.

It is equivalent to the GUI Library's It_SetNewSessionHook [Page 145] request.

This event is applicable for both the OCX and the EXE versions.

SapMenus Object Collection

SapMenus Properties

Application	SapEvent [Page 198]	Returns top-level event object. (EXE only).
Count	long	Number of menus in the collection.
Parent	SapEvent	Returns parent event object.

SapMenus Methods

```
SapMenu Item (long Index)
```

Parameters

Index	Index of the desired menu, ranging from 0 to Count - 1

Return Value

The SapMenu [Page 256] object associated with the index; NULL if no such key exists.

Comments

This is the default method for the SapMenus collection.

SapMenu Object

SapMenu Properties

Active	BOOL	TRUE if menu item is active; FALSE if disabled.
Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
ChildIndex	short	Index of first child in submenu.
Expanded	BOOL	TRUE if submenu has been expanded.
Flags	short	Menu status flags.
Name	BSTR	Name of menu item.
NextIndex	short	Index of next peer menu item.
Parent	SapMenus [Page 253]	Returns parent menus collection.
ParentIndex	short	Index of parent menu item.
Popup	BOOL	TRUE if menu item has a submenu.
PopupRequested	BOOL	TRUE if menu item's submenu is being expanded.

SapKeys Object Collection

SapKeys Properties

Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
Count	long	Number of keys in the collection.
Parent	SapEvent	Returns parent event object.

SapKeys Methods

```
SapKey Item (long Index)
```

Parameters

Index	Index of the desired key, ranging from 0 to Count - 1

Return Value

The SapKey [Page 261] object associated with the index; NULL if no such key exists.

Comments

This is the default method for the SapKeys collection.

SapKey Object

SapKey Properties

Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
Info	BSTR	Quick info or tooltip associated with the toolbar's icon.
Name	BSTR	Name of the key.
Order	short	Order in which key appears in the application toolbar, or -1 if not in toolbar.
Parent	SapKeys [Page 258]	Returns parent keys collection.
SAPValue	long	SAP PFKEY value (0 to 99).
ToolbarHasIcon	BOOL	TRUE if toolbar string contains an icon.
ToolbarIconCode	BSTR	Icon code that appears between @ characters in toolbar string.
ToolbarIconName	BSTR	Six-character icon internal format name for toolbar icon.
ToolbarIconText	BSTR	Text that follows the icon in the toolbar.
VKValue	long	Value of the key, expressed in Windows Virtual-Key codes. This is the value that must be used when setting the KeyToSend property in the Event object. Default property.

SapControls Object Collection

SapControls Properties

Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
Count	long	Number of controls in the collection.
Parent	SapEvent	Returns parent event object.

SapControls Methods

```
SapControl Item (long Index)
```

Parameters

Index	Index of the desired control, ranging from 0 to Count - 1

Return Value

The SapControl [Page 266] object associated with the index; NULL if no such control exists.

Comments

This is the default method for the SapControls collection.

SapControl Object

SapControl Properties

Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
Area	short	Screen area ID.
Block	short	Block ID within a step loop.
Bottom	long	Bottom position of the control on the screen, in character units.
ChildIndex	short	Index of first child of this control.
ColorNumber	short	Number of control color. Used with lists together with the Intensive and Inverse properties to determine the background colors of controls. Also used to specify an icon background color.
Container	short	Container ID for a manager. For non-manager controls, the container ID for the control's parent manager.
DataElement	BSTR	Data element; available only after GetEventFull().
DataElement		
Supplement	BSTR	Data element supplement; available only after GetEventFull method.
FieldName	BSTR	Name of field associated with this control; available only when the HasFieldNames property is TRUE.
Flags	short	Flags indicated whether control is selected and/or modified. The same information is also available using the Modified, Selected, and HasFieldNames properties. Read/write.
Group	short	Group ID for radio buttons. Radio buttons with the same area, block, and group IDs are in the same group and are mutually exclusive.
Has3D	BOOL	TRUE if control is marked with a 3D style.
HasFieldNames	BOOL	TRUE if the control has its FieldName and TableName properties set.
HasIcon	BOOL	TRUE if the control contains an icon.
Height	long	Height of the control on the screen, in character units.
HotSpot	BOOL	TRUE if control is a hotspot, allowing single- rather than double-clicking.

IconCode	BSTR	The icon code contained between @ characters, not including the quick info string. Usually a two-character code, ranging from “00” to “5C” in 3.1G, but may also be the 6-character icon name.
IconName	BSTR	Root name of the bitmap file for the icon. For example, the OK icon, with a value of “@01@”, returns “S_OKAY”. This is the same as the “internal format” listed by the details button with the ICON transaction. Empty if the control has no icon.
IconQuickInfo	BSTR	Quick info (i.e., tooltip) associated with the icon. Empty if the control has no icon.
IconText	BSTR	Text that follows the icon in the control. Empty if the control has no icon.
Intensive	BOOL	TRUE if control value is displayed in an intensive color. In R/3, the choice of intensive color is user-selectable and can differ among control types.
Inverse	BOOL	TRUE if control value is displayed in an inverse (or dimmed) color. In R/3, the choice of inverse color is user-selectable, and is restricted to use within lists.
Left	long	Leftmost position of the control on the screen, in character units.
Modified	BOOL	TRUE if the control has been modified. This property is set by the other read/write properties, but may also be set manually. Read/write.
Name	BSTR	Text name of the control.
NextIndex	short	Index of the next peer of this control.
Parent	SapControls [Page 263]	Returns parent controls collection.
ParentIndex	short	Index of the parent of this control. -1 if no parents.
ProportionalFont	BOOL	TRUE if the control value is displayed using a proportional font; FALSE if displayed using a fixed-width font.
ReadOnly	BOOL	TRUE if the control is marked with a read-only style. Valid for control types that can be edited.
Right	long	Rightmost position of the control on the screen, in character units.
Selected	BOOL	TRUE if the control (a radio button or check box) has been selected. Read/write.

SapControl Properties

Style	long	Flags indicating the visual style of the control. See the description of the *dwStyle* member of the IT_CTRL [Page 87] data structure in the GUI Library for a list of flag values. Currently, only a limited set of flags are available directly through separate properties.
SymbolFont	BOOL	TRUE if the control value is displayed using the symbol font; FALSE if displayed using a text font.
TableControlInfo	SapTableControlInfo [Page 269]	Returns table control info object for SapTable control; NULL if control is not a SapTable.
TableName	BSTR	Name of the table associated with this control; available only when the HasFieldNames property is TRUE.
Top	long	Top position of the control on the screen, in character units.
Type	SapControlType [Page 272]	Control type.
UppercaseInput	BOOL	TRUE if an editable control converts all input into uppercase.
Value	BSTR	The value of the control, represented as a text string. Read/write. Default property.
Visible	BOOL	TRUE if this control is visible on the screen.
Width	long	Width of the control on the screen, in character units.

SapTableControlInfo Properties

Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
Columns	short	Total number of data columns in table.
ColumnSelectionType	SapSelectionType Enumeration [Page 276]	Indicates if able to select none, one, or multiple columns.
ControlOKCode	BSTR	OK code needed to access the table's modal control box.
DataRows	short	Total number of data rows in table.
FixedColumns	short	Number of fixed, non-scrolling columns in table.
Flags	long	Table control style flags. See the IT_TABLEINFO [Page 100] section in the GUI Library for more information.
Parent	SapControl [Page 266]	Returns parent control object.
Rows	short	Number of rows appearing on screen. Does not include table captions.
RowSelectionType	SapSelectionType Enumeration [Page 276]	Indicates if able to select none, one, or multiple rows.
ScrollOKCode	BSTR	OK code needed to scroll this particular table.
StartColumn	short	Starting column for scrolling. Read/write.
StartRow	short	Starting data row for scrolling. Read/write.

SapTabStripControlInfo Properties

Application	SapEvent [Page 198]	Returns top-level event object. EXE only.
GetNumButtons	short	Total number of tab buttons in a tab strip.
GetNumLocalButtons	short	Number of local buttons
GetLeftButton	short	Leftmost tab button given by the server.
GetActiveButton	short	Active tab button given by the server.
SetActiveButton	BOOL	Set an active tab button.
GetNumButtonRows	short	Reserved for future use
Parent	SapControl [Page 266]	Returns parent control object.
GetButtonHeight	short	Get the height of button, reserve for future use.
GetTabOrientation	[Page 276]short	Tab Orientation:
		ITOLE_TABSTRIP_TOP
		ITOLE_TABSTRIP_BUTTOM
		ITOLE_TABSTRIP_LEFT
		ITOLE_TABSTRIP_RIGHT.
GetScrollArrowPos	short	Scroll arrow position for tab buttons:
		ITOLE_TABSTRIP_SCROLL_LL
		ITOLE_TABSTRIP_SCROLL_LR
		ITOLE_TABSTRIP_SCROLL_RR
GetTextOrientation	short	Tab strip text orientation:
		ITOLE_TABSTRIP_TEXT_VERTICAL
		ITOLE_TABSTRIP_TEXT_HORIZONTAL
GetTabStyle	long	Tab style:
		ITOLE_TABSTRIP_TAB_AS_TAB
		ITOLE_TABSTRIP_TAB_AS_BUT

Enumerated Types

SapControlType Enumeration

SapAnyType is a general control type that can be used for any control type. It is equivalent to 0.

The following table describes the SAP control types, and their equivalent GUI Library control types.

SAP Control Type	Equivalent to GUI Library Type
SapStatic	CTRL_STATIC
SapEdit	CTRL_EDIT
SapPassword	CTRL_PASSWORD
SapPushButton	CTRL_PUSHBUTTON
SapRadioButton	CTRL_RADIOBUTTON
SapCheckBox	CTRL_CHECKBOX
SapFrameBox	CTRL_FRAMEBOX
SapLine	CTRL_LINE
SapMatch	CTRL_MATCH
SapListStatic	CTRL_LISTSTATIC
SapGraphStatic	CTRL_GRAPHSTATIC
SapMatchFix	CTRL_MATCHFIX
SapIcon	CTRL_ICON
SapListCheckBox	CTRL_LISTCHECKBOX
SapTable	CTRL_TABLE
SapTableCaption	CTRL_TABLE_CAPTION
SapTableColumn	CTRL_TABLE_COLUMN
SapManager	CTRL_MANAGER
SapTabButton	CTRL_TABBUTTON
SapTabStrip	CTRL_TABSTRIP

See Also

IT_CTRL [Page 87].

SapDirection Enumeration

SapHere	Stay here. Equivalent to 0.
SapLeft	Look to the left. Equivalent to GUI Library's FC_LEFT.
SapRight	Look to the right. Equivalent to GUI Library's FC_RIGHT.
SapDown	Look down. Equivalent to GUI Library's FC_DOWN.
SapUp	Look up. Equivalent to GUI Library's FC_UP.

See Also

ItEv_FindControl [Page 154].

SapGetType Enumeration

SapGetAutomatic	Automatically do a GetEvent [Page 212] after a SendEvent [Page 225] (default).
SapGetExtendedAutomatic	Automatically do a GetEventFull [Page 215] after a SendEvent.
SapGetManual	Requires manual GetEvent* call after a SendEvent.

SapGuiFlags Enumeration

The SapGuiFlags enumeration type are equivalent to the GUI Library's connection flags [Page 128]. See the discussion of the GUI Library flags for details of the role and behavior of these flags.

GUI Component Flag	Equivalent to GUI Library's Flag
SapGuiFront	SAPGUI_FRONT
SapGuiR2	SAPGUI_R2
SapGuiFullmenu	SAPGUI_FULLMENU
SapGuiAbsoluteCoord	SAPGUI_ABSOLUTE_COORD
SapGui45ACoord	SAPGUI_45A_COORD
SapGuiActiveX	SAPGUI_ACTIVEX

SapSelectionType Enumeration

SapSelectUnknown	Do not know what type of unit (e.g., row or column) selection is supported.
SapSelectNone	Does not support selection by this unit (e.g., row or column).
SapSelectSingle	Supports selection of a single unit (e.g., row or column).
SapSelectMultiple	Support selection of multiple units (e.g., rows or columns).

See Also

IT_TABLEINFO [Page 100].

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

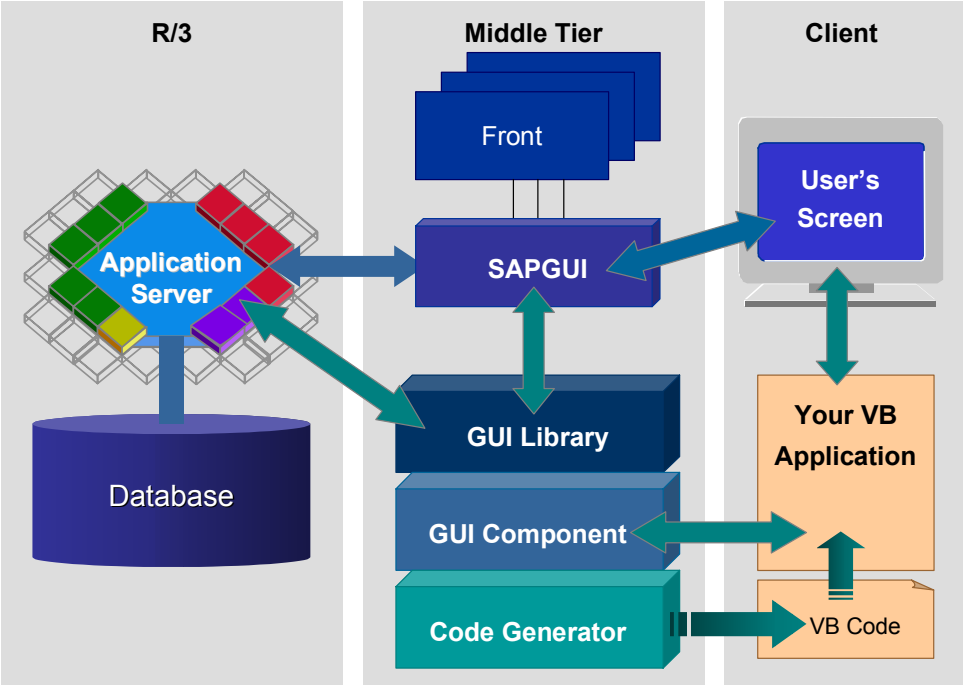
The SAP Automation Code Generator lets you program SAP alternate user interfaces by example. You can create Visual Basic programs simply by interacting with the SAP system through the code generator. The code generator:

- 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

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 [Page 280].
2. Open a connection [Page 281].
3. Log on to the R/3 System [Page 282].
4. Use the Generator GUI [Page 283] to perform all your R/3 activities.
5. Log off again [Page 289].

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 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 [Page 282].

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 [Page 284]

Avoiding Menus [Page 297]

Requesting Generator Code

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

- variable declarations [Page 285]
- message checks [Page 287]
- subroutines [Page 286]

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 [Page 288]

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 SystemCreate 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

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

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 [Page 285]

Creating a New Subroutine [Page 286]

Inserting a Message Check [Page 287]

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

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

Understanding Generated Code [Page 291]

Avoiding Menus [Page 297]

Finding Controls [Page 298]

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
'------------------------------------------------------------------------
```

```
-
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.
```

```
'-------------------------------------------------------------------------
-
    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"
```

```
'------------------------------------------------------------------------
-
'  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"
```

```
    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

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