

The RFC Generator



HELP.BCFESDE3

Release 4.6C



Copyright

© Copyright 2001 SAP AG. Alle Rechte vorbehalten.

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

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

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

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

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

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

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

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

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

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

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

Symbole

Symbol	Bedeutung
	Achtung
	Beispiel
	Hinweis
	Empfehlung
	Syntax

Inhalt

The RFC Generator	5
The RFC Generator	6
Introduction	7
Creating RFC Stubs and Example Programs	9
Introduction.....	10
Generating an RFC Program	11
Displaying an RFC Program	13
Downloading an RFC Program	14
Clearing RFC Programs from the System	15
Setting Generation Options.....	16
Generator Example Programs	20
Complete ANSI C Client Example	21
Complete ANSI C Server Example	25
Complete Visual Basic Client Example.....	31
Complete Visual Basic Server Example	35
Complete OLE Automation Client Example	39
Calling RFC Stub Programs	43
Compiling and Linking on the Workstation.....	44
Tasks Performed by the RFC Stub	46
Tasks Performed by a Stub Caller	47

The RFC Generator

The RFC Generator

[Introduction \[Seite 7\]](#)

[Creating RFC Stubs and Example Programs \[Seite 9\]](#)

[Generator Example Programs \[Seite 20\]](#)

[Calling RFC Stub Programs \[Seite 43\]](#)

Introduction

The function library in R/3 provides a facility for generating and then downloading RFC programs to a workstation or PC. This facility is the RFC Interface Generator. With this tool, you can create RFC stub programs (that call SAP function modules) and example programs (that show how to call stub programs).



The RFC Generator is only available for and in R/3 Systems and **not** for R/2 Systems.

Generating RFC Stubs: RFC stub programs contain all the parameter-handling and communications necessary to call SAP function modules from a non-SAP System.

Once a stub has been exported to your machine, you can compile it as a library file or DLL (dynamic-link library) routine. DLL routines can be called without having been linked together with your program at compile time. You can call DLL routines from any programming language whose compiler offers DLL options. (This includes, for example, most recent C and BASIC compilers.)

Generating Example Programs: The RFC Interface Generator provides example programs for different programming languages. Both ANSI C and Visual Basic are supported. With Release 3.0, RFC client example programs as well as RFC server example programs are provided for these programming languages.

The list of example programs will be extended in later releases. The list at the end of this chapter shows in detail which example programs are provided in which release.



To view the list of example programs that you can actually generate use the F4 Help key or press the value button when editing the generation settings.

The procedure to generate an example program is always the same. Therefore, after having tested one example you should not come across any problems if you want to generate another one.



A text file (e.g. saprfcex.txt) is provided for each example program that you can generate. Read this text file to get the latest information about the example program you generated, i.e. its contents, how to compile it, etc.

You can generate example programs in C or Visual Basic that contain all the code needed to call RFC stubs.

Once an example program has been downloaded to your machine, you can edit it (to fit your own needs) and compile it as a normal application program.

The following topics are available:

- [Creating RFC Stubs and Example Programs \[Seite 9\]](#)
Describes how to use the RFC Interface Generator to create RFC programs (stubs and example stub-callers) in the SAP System and download them.
- [Calling RFC Stub Programs \[Seite 43\]](#)

Introduction

Describes what an RFC stub does, how to call it from a C or Visual Basic program, and how to use an example program to develop your own application.

List of RFC example programs that are available

Name	Description	Release
RFC_ANSIC	ANSI C client stub function	2.2
RFC_ANSIC_EXP	ANSI C client example	2.2
RFC_MSVB_EXP	ANSI C / Visual Basic client example	2.2
RFC_MSVB_EXT	Extended Visual Basic client stub	2.2
RFC_MSVB_EXT_EXP	Extended Visual Basic client example	2.2
RFC_ANSIC_CL_EXP	Complete ANSI C client example	3.0 *)
RFC_ANSIC_SR_EXP	Complete ANSI C server example	3.0 *)
RFC_MSVB_CL_EXP	Complete Visual Basic client example	3.0 *)
RFC_MSVB_SR_EXP	Complete Visual Basic server example	3.0 *)
RFC_OLEA_CL_EXP	Complete OLE Automation client example	3.0*)

- *) A complete example program means you have to generate only one example instead of using a certain combination of two example programs.



This list will change when generation templates for new example programs come out (e.g. C++).

As of R/3 Release 3.0, you can generate RFC example programs that let you record test RFC calls and later re-run (play back) them. For each example program, the system also generates a readme file containing an exact description of the program. To use these RFC example programs, you should have a basic familiarity with RFC and knowledge of C or Visual Basic.

Creating RFC Stubs and Example Programs

This section contains the following topics:

[Introduction \[Seite 10\]](#)

[Generating an RFC Program \[Seite 11\]](#)

[Displaying an RFC Program \[Seite 13\]](#)

[Downloading an RFC Program \[Seite 14\]](#)

[Clearing RFC Programs from the System \[Seite 15\]](#)

[Setting Generation Options \[Seite 16\]](#)

Introduction

Introduction

R/3 users can create new RFC stubs or example programs for any SAP function module registered as remotely callable. The basic steps for creating RFC stubs and example stub-callers are:

1. Set generation options, if necessary (in the R/3 System).
2. Generate the program (in the R/3 System).
3. Download the program (from R/3 to your workstation/PC).
4. Compile and link the program (on the workstation/PC).

Tools for performing the steps on the R/3 side are available in the function library's RFC Interface Generator. (Enter the function library by selecting *Tools* → *ABAP Workbench* → *Development* → *Function library*).

Once in the function library, enter the name of the desired function module and select *Display* or *Change*. In the resulting screen, access the RFC Interface Generator by selecting *Utilities* → *RFC-Interface*. The generator sub-menu offers the following functions:

- [Generating an RFC Program \[Seite 11\]](#)
- [Displaying an RFC Program \[Seite 13\]](#)
- [Downloading an RFC Program \[Seite 14\]](#)
- [Clearing RFC Programs from the System \[Seite 15\]](#)
- [Setting Generation Options \[Seite 16\]](#)

Generating an RFC Program

You can generate an RFC stub for one or more function modules at a time. When you want to create more than one, you generate each separately (that is, one after the other), but the new code is appended to existing code files.

All generated stubs remain in existence until you clear them from the system explicitly or leave the function library. (Generated code is also deleted when you change the values of certain generation options in the Settings window.)

You can also generate example programs for multiple function modules, but it is not recommended: names for generated sub-routines will not be unique. When generating example programs, download (transfer) or clear (delete) the code for one function module before going on to the next.

The RFC Interface Generator collects all code into specific files, no matter how many different function modules you generate for:

- for RFC stubs
 - file saprfcc.c
 - file saprfcc.h
 - file saprfcc.txt
- for example programs in C
 - file saprfcce.c
 - file saprfcce.h
 - file saprfcce.txt



As of Release 3.0B, C programs can be generated for several function modules, that is, one example program for several functions. In former releases, one example program was generated for each function. Now only one example program for an SAP Business API containing several functions has to be sent to an external developer, for instance.

- for example programs in Basic
 - file saprfcbe.txt (Info/Comments)
 - file saprfcbe.mak (Make file)
 - file saprfcbf.bas (Fctn decls)
 - file saprfcbg.bas (Globals decls)
 - file saprfcba.frm (Start win)
 - file saprfcbc.frm (Call win)
 - file saprfcbl.frm (Logon win)
 - file saprfcbo.frm (Logoff win)
 - file saprfcbr.frm (Error win)

Generating an RFC Program

- file saprfcbs.frm (Structures win)
- file saprfcvt.frm (Tables win)



As of Release 3.0B, an additional Visual Basic example program will be generated calling the SAP Automation Server (RFC Client) instead of the RFC API.

To generate a single program:

1. Get into the function library main window, enter the function module name, and select *Display* or *Change*.
2. In the resulting screen, use menu-item *Utilities* → *RFC-Interface* → *Generate* to create an RFC program.

The Settings window: The system generates code according to option settings in the Settings window.

This window, which appears by default each time you create a new program, lets you set generation options before generating. You can disable the appearance of the Settings window by setting the Change settings variable in the window.

If the window has already been disabled, and you want to set options before generating, select *Utilities* → *RFC-Interface* → *Settings*. (Using the Settings window is explained in [Setting Generation Options \[Seite 16\]](#).)

The Status window: The Status window appears by default during program generation. This window records progress by displaying the percent of the generation completed. You can cancel the generation in this window if desired.

Enable (or disable) the status window by setting the Display status field in the Settings window.

3. If you want to generate further programs, return to the main window (to enter the new function module name) and repeat the two steps above.

The system generates new code and appends it to the existing code files.

When a generation has finished, the new code appears (unless disabled in the Settings window).

If you are generating more than one program, it is helpful to disable the code display (in the Settings window) until the last generation is complete. The topic [Setting Generation Options \[Seite 16\]](#) explains how to do this.



You cannot generate RFC programs for function modules that are not registered as remotely callable. This registration occurs in the Administration screen of the function library. (See [RFC Programming in ABAP \[Extern\]](#) for more information.)

For further information on the generated example programs, see [Generator Example Programs \[Seite 20\]](#).

Displaying an RFC Program

The *RFC-Interface* → *Display* menu item displays already generated code, if any exists. This function is useful when the automatic display has been disabled in the Settings window. It is not available if no generated code files exist.

Downloading an RFC Program

Downloading an RFC Program

Select *Utilities* → *RFC-Interface* → *Transfer* to transfer the generated program to your workstation or PC.

If the Change settings option is set (in the Settings window), the Transfer function prompts you for the directories where the generated code should be placed. The directory names required are the same as those you specify in the Settings window. (See [Setting Generation Options \[Seite 16\]](#) for more information.)

Clearing RFC Programs from the System

The *RFC-Interface* → *Delete* menu item deletes all generated code files from the system. This function is important since you cannot re-generate a stub or example program for a function module without first deleting all code files (or exiting from the function module).

Setting Generation Options

Setting Generation Options

The Settings window lets you specify generation parameters for creating RFC stubs and example programs. Unless disabled, this window appears automatically when you generate the stub. You can also call the window up directly by selecting *Utilities*→*RFC-Interface*→*Settings*.

If you set options in this window, they are valid only for the duration of a function library session. You can save the settings beyond the length of a session using the *Save setting* button.



Changing the generation forms requested and then running the Generate function causes existing code files to be deleted. Thus, if you change the Generation Forms values while generating a series of stubs, you will clear the system of all code generated up to the point of change.

Settings window fields are as follows:

- Generate Forms
- Download Paths
- Generate Settings
- Function-Key Buttons

Generate Forms Fields

With these variables you specify the kind of program you want generated. You can request more than one form at a time. However, do not generate a stub and example program at the same time, unless you want them downloaded to the same directories.

- RFC_TEST
This form is a debugging aid: it generates a list of all parameters to the function module, with variable information (field type, field length, and so on) for each one.
- RFC_ANSIC
This form generates an RFC client stub in ANSI C.
- RFC_ANSIC_EXP
This form generates an ANSI C RFC client caller example program calling the ANSI C RFC client stub.
- RFC_MSVB_EXP
This form generates a Visual Basic client caller example program calling the ANSI C RFC client stub.
- RFC_MSVB_EXT
This form generates a Visual Basic RFC client stub. The function is implemented using the RFC API Extended functions.
- RFC_MSVB_EXT_EXP
This form generates a Visual Basic client caller example program calling the Visual Basic client stub.

Setting Generation Options

- **RFC_ANSIC_CL_EXP**
This form generates a complete ANSI C client example program.
- **RFC_ANSI C_SR_EXP**
This form generates a complete ANSI C server example program.
- **RFC_MSVB_CL_EXP**
This form generates a complete Visual Basic RFC client example program.
- **RFC_MSVB_SR_EXP**
This form generates a complete Visual Basic RFC server example program.
- **RFC_OLEA_CL_EXP**
This form generates a complete OLE Automation client example program.

Download Paths Fields

- **Header files**
Enter the path name for the directory where RFC include files (any *.h files) are to be stored on your workstation. (This should be the directory containing the RFC API header files *saprfc.h* and *sapitab.h*).

The RFC Interface Generator will place any generated include files here, for example *saprfcc.h* (created for an RFC stub) and *saprfcce.h* (created for example programs).
- **Source files**
Enter the path name for the directory where generated C-language programs (any *.c files) are to be stored on your workstation.

If you are generating an RFC stub: This path should be the directory containing your project directory.

If you are generating an example program in C: This path should be the directory (or project) where source files for your application project should go.

If you are generating both stub and example in C: The directory you specify must contain both the source files for the DLL and the source files for the application. (If you are working with projects, both the DLL project and the application project must be in the same directory.)

If you are generating an example program in Basic: Use the *Other* field to specify the directory for Basic programs.
- **Other**
Enter the path name for all files other than those with .c or .h extensions. This includes generated Basic files (*.bas), all text (*.txt) files, and any others.

Generate Settings Fields

- **Change Settings**
This option enables or disables the *Settings* window during the generation process. When enabled, the window appears automatically with each request to generate. When disabled, it does not appear.

Setting Generation Options

This field also determines whether you are prompted for target directories, when you request a download.

- *Display source code*

This option causes the code to be displayed after it is generated.

- *Suppress ONCE elements*

Set this option when you are generating an RFC program to add to an existing group and the original code files for the group have already been cleared from the system. (Files are cleared when you explicitly delete the code, or when you leave the function library).

For example, if you generate and download a set of stubs and later decide you want to add more modules, use the *Suppress ONCE elements* option.

You need this option to tell the system that the current program will be an add-on. When generating, the system produces some code that recurs for each desired function module, and some code that only occurs once for the entire set. When you create add-ons, the once-only code is not needed a second time. However, the system has no way of knowing this, since the original code files have been deleted.

- *Download source code automatically*

This option causes the program to be downloaded to your machine automatically after generation.

If the *Change settings* field is also set, the system prompts you for the name of the directories where the code files should be placed. (Otherwise, target directories are assumed to be those currently specified in the *Path* fields in the *Settings* window.)

- *Delete source code automatically*

This option causes all code files to be cleared from the system automatically after a generation. If you request automatic download, the files are cleared after the download.

This option is useful when you request generation and download from a remote system and want to clean up files after you are finished. Code files disappear anyway when the remote user logs off, but for multiple generations in a single session, the files must be cleared explicitly.

- *Display status*

This option enables or disables the program indicator window. When enabled, this window appears automatically during the generation process, showing how much of the generation has been completed. When disabled, the window does not appear.

Function-Key Buttons

After you set the options that you want, exit the window with one of the following:

- *Continue*

If you entered the window by requesting a generation, the generation now proceeds. If you entered by requesting *Settings*, the window simply closes.

- *Get setting*

This button resets the options to the settings last saved with the *Save setting* button.

- *Save setting*

Setting Generation Options

This button saves the current option settings.

Normally, changes to option settings are valid as long as you stay in the function library, and are thrown away when you exit. However, if you click on Save setting, your current settings will be saved and made available again each time you re-enter the function library.

- *Get default*

This button resets the *Settings* window to its default (system-defined) settings.

- *Cancel*

This button closes the *Settings* window without saving your option settings. If you entered this window by selecting the *Generate* setting, *Cancel* also cancels the generation request.

Generator Example Programs

Generator Example Programs

The following example programs are explained in detail:

[Complete ANSI C Client Example \[Seite 21\]](#)

[Complete ANSI C Server Example \[Seite 25\]](#)

[Complete Visual Basic Client Example \[Seite 31\]](#)

[Complete Visual Basic Server Example \[Seite 35\]](#)

[Complete OLE Automation Client Example \[Seite 39\]](#)

Complete ANSI C Client Example

```
-----  
SAP AG - R/3 Remote Function Call Interface Generation  
RFC_ANSIC_CL_EXP E 30A 22.09.1995 15:22  
Complete ANSI C client example  
saprfccl.txt  
-----
```

Code has been generated for the following function module

- ABAP4_CALL_TRANSACTION
Executes the ABAP command CALL TRANSACTION.

RFC_ANSIC_CL_EXP : Complete ANSI C client example

Description

The generation form RFC_ANSIC_CL_EXP generates a complete ready-to-compile RFC client example program written in ANSI C. This program lets you test your remote function module without manual programming.

The example program prompts the user to input exporting parameters before the call and to display importing parameters after the call. Dialogs for setting values in and displaying internal table objects are also provided.

The example program uses ANSI C statements and standard library routines. The user interface provides standard input and output functionality. Help texts are available for each dialog step.

The example program contains type and variable definitions for all function module parameters. It also includes functions for logon, for init, input and display of export and import parameters, and for the handling of internal tables.

Files generated by this form

- saprfccl.h
Type definitions for field and structure parameters
- saprfccl.c
 - Help function
 - SET/GET macros and functions

Complete ANSI C Client Example

- Logon dialog
- RFC error function
- ANSI C client stub
- Main program
- Variable definitions for function module parameters
- Init, input, display of export and import parameters
- Handling of internal tables.
- saprfccl.def
 - .def file for C WINDOWS applications
 - saprfccl.mkx
 - Makefile for UNIX platforms.
 - saprfccl.txt
 - This text.

Generation

You can generate RFC_ANSIC_CL_EXP for one function module at a time. The generation of example programs for multiple function modules at a time is not recommended, because names for generated subroutines will not be unique. When generating example programs, download and clear the code for one function module before going on to the next.

MAKE

Under UNIX you can use the generated UNIX makefile. Change it to the needs of your UNIX development environment and start it as follows:

- make -f saprfccl.mkx



The generated UNIX makefile should run under HP-UX. It may not fit your UNIX platform and development environment. Please make appropriate adjustments, e.g. compiler options, in order to run the makefile correctly.

In many development systems (especially WINDOWS-based systems) you must define projects in which to develop your programs. Under WINDOWS, define an application project (for example in MS Visual C++, a QuickWin project), that contains the following files:

- saprfccl.c
- saprfccl.def

Complete ANSI C Client Example

- `librfc16.lib` or `librfc32.lib`

The following header files will be included:

- `saprfc.h`
- `sapitab.h`
- `saprfccl.h`

Under WINDOWS, set the following compiler option:

- `memory model: large.`

Dependencies

- Files you need from the RFC SDK:
 - `saprfc.h`
ANSI C type and function declarations for the RFC API functions.
 - `sapitab.h`
ANSI C type and function declarations of the RFC API functions that support the handling of internal tables.
 - `librfc.a`
Static library that provides the RFC API functions (UNIX only).
 - `librfc16.lib`
Import library file used at link time. `librfc16.lib` is provided with the RFC SDK, but can also be created from a `librfc16.dll` file in WINDOWS development systems (WINDOWS 16-bit only).
 - `librfc16.dll`, `librfc2.dll`, `nidll.dll`, `dptrcerr.dll`
Dynamic Link Libraries that provide the RFC API functions (WINDOWS 16-bit only).
 - `librfc32.lib`
Import library file used at link time. `librfc32.lib` is provided with the RFC SDK, but can also be created from a `librfc32.dll` file in WINDOWS development systems (WINDOWS 32-bit only).
 - `librfc32.dll`
Dynamic Link Library that provides the RFC API functions (WINDOWS 32-bit only).

Complete ANSI C Client Example



These files should be in the same directory as your generated example program. The dynamic link libraries may also be stored in another directory, but this directory must be defined in the WINDOWS path variable.

Test support

The example program provides support to automate tests. Use program argument `-o <file name>` to record a test, edit the recorded file if necessary and play it back by using program argument `-i <file name>`. You can use `-o` and `-i` at a time. Verify your test results by comparing the input and output file. Use program argument `-?` to get help on all supported program arguments.

Tested development systems

- Microsoft Visual C++ version 1.0, 1.5

Warnings

- The RFC example programs written in ANSI C use the original length for RFC_CHAR parameters or parameters that are based on the RFC_CHAR data type. No null character is used, to mark the end of such strings.
- Using the generated macros and functions for type conversion (e.g. SETCHAR, GETCHAR) the length of parameter values are checked. There are no further checks or conversions (e.g. upper case) provided. The caller program is responsible for transferred values!

Changes

- 11/1/94 Definition of preprocessor symbol SAPonWindows in saprfc.h.
- 5/2/95 Definition of import parameters after RfcCall and before RfcReceive.
- 8/11/95 Multi-function support. This feature allows to generate one RFC client example program for SAP Business API's that contain more than one function module. You can generate a RFC client example program for more than one function module - one after another. The example program provides a line menu to select one of the included functions.

Complete ANSI C Server Example

```
SAP AG - R/3 Remote Function Call Interface Generation
RFC_ANSIC_SR_EXP E 30A 22.09.1995 15:22
Complete ANSI C server example
saprfcsr.txt
```

Code has been generated for the following external function

- ABAP4_CALL_TRANSACTION
Executes the ABAP command CALL TRANSACTION.

RFC_ANSIC_SR_EXP : Complete ANSI C server example

Description

The generation form RFC_ANSIC_SR_EXP generates a complete ready-to-compile external RFC server example program written in ANSI C. This program lets you test external functions without manual programming.

You can start your external RFC server from the R/3 function library testbed by specifying an appropriate RFC destination. Once called the RFC server installs external functions and waits for function calls until you leave the function library.

The example program displays importing parameters and prompts the user to raise an exception or to input exporting parameters. Dialogs for displaying and setting values in internal table objects are also provided.



Under UNIX you must start the SAPGUI manually from the shell to get the output and input of the external server displayed. Starting the SAPGUI you must specify the name of the application server, the name of the R/3 system and the R/3 system number:

- e.g. /sapmnt/C11/exe/saptemu hs0001 name=C11 nr=00

Complete ANSI C Server Example

The example program uses ANSI C statements and standard library routines. The user interface provides standard input and output functionality. Help texts are available for each dialog step.

The example program contains type and variable definitions for all function module parameters. It also includes functions for init, input and display of importing and exporting parameters, and for the handling of internal tables.



Your external server will be started by the first call. After performing the first call the server waits for further calls by calling the 'RfcDispatch' function.

To close the connection from the ABAP/4 side, call function module 'RFC_CONNECTION_CLOSE'. Closing the connection will terminate your external server. When leaving the R/3 function library testbed 'RFC_CONNECTION_CLOSE' is called by default.

Manually termination of the external server causes a runtime error on the ABAP/4 side when calling the terminated server. Use 'RFC_CONNECTION_CLOSE' to reinitialize the connection. After reinitialization the external server is started again by the next call.

Files generated by this form

- saprfcsr.bas
Type definitions for field and structure parameters
- saprfcsr.c
 - Help function
 - SET/GET macros and functions
 - RFC error function
 - External function
 - Init, input, display of import and export parameters
 - Handling of internal tables
 - Server stub
 - Variable definitions for external function parameters
 - Main program.
- saprfcsr.def
.def file for C WINDOWS applications
- saprfcsr.mkx
Makefile for UNIX platforms.

Complete ANSI C Server Example

- saprfcsr.txt
This text.

Generation

You can generate RFC_ANSIC_SR_EXP for one function module at a time. The generation of example programs for multiple function modules at a time is not recommended, because names for generated subroutines will not be unique. When generating example programs, download and clear the code for one function module before going on to the next.

MAKE

Under UNIX you can use the generated UNIX makefile. Change it to the needs of your UNIX development environment and start it as follows:

- make -f saprfcsr.mkx



The generated UNIX makefile should run under HP-UX. It may not fit your UNIX platform and development environment. Please make appropriate adjustments, e.g. compiler options, in order to run the makefile correctly.

In many development systems (especially WINDOWS-based systems) you must define projects in which to develop your programs. Under WINDOWS, define an application project (for example in MS Visual C++, a QuickWin project), that contains the following files:

- saprfcsr.c
- saprfcsr.def
- librfc16.lib or librfc32.lib

The following header files will be included:

- saprfc.h
- sapitab.h
- saprfcsr.h

Under WINDOWS, set the following compiler option:

- memory model: large.

Dependencies

- Files you need from the RFC SDK:
 - saprfc.h

Complete ANSI C Server Example

- ANSI C type and function declarations for the RFC API functions.
- sapitab.h
ANSI C type and function declarations of the RFC API functions that support the handling of internal tables.
- librffc.a
Static library that provides the RFC API functions (UNIX only).
- librffc16.lib
Import library file used at link time. librffc16.lib is provided with the RFC SDK, but can also be created from a librffc16.dll file in WINDOWS development systems (WINDOWS 16-bit only).
- librffc16.dll, librffc2.dll, nidll.dll, dptrcrr.dll
Dynamic Link Libraries that provide the RFC API functions (WINDOWS 16-bit only).
- librffc32.lib
Import library file used at link time. librffc32.lib is provided with the RFC SDK, but can also be created from a librffc32.dll file in WINDOWS development systems (WINDOWS 32-bit only).
- librffc32.dll
Dynamic Link Library that provides the RFC API functions (WINDOWS 32-bit only).



These files should be in the same directory as your generated example program. The dynamic link libraries may also be stored in another directory, but this directory must be defined in the WINDOWS path variable.

Test support

The example program provides support to automate tests. Use program argument `-o <file name>` to record a test, edit the recorded file if necessary and play it back by using program argument `-i <file name>`. You can use `-o` and `-i` at a time. Verify your test results by comparing the input and output file. Use program argument `-?` to get help on all supported program arguments.

Complete ANSI C Server Example



When defining a RFC destination for an external RFC server (SM59) in the current R/3 release you cannot specify additional program arguments like `-o` or `-i` for the RFC server program. You have to use a shell script or procedure to add additional program arguments.

Tested development systems

- Microsoft Visual C++ version 1.0, 1.5

Warnings

- The RFC example programs written in ANSI C use the original length for RFC_CHAR parameters or parameters that are based on the RFC_CHAR data type. No null character is used, to mark the end of such strings.
- Using the generated macros and functions for type conversion (e.g. SETCHAR, GETCHAR) the length of parameter values are checked. There are no further checks or conversions (e.g. upper case) provided. The caller program is responsible for transferred values!

Changes

- 6/26/94 Support for the following data types:
 - RFC_BCD : packed numbers
 - RFC_BYTE: hexadecimal fields
- 11/1/94 Definition of preprocessor symbol SAPonWindows in saprfc.h.
- 11/1/94 You can write screen output to a file by using the `-o` program argument:
 - e.g. `-o saprfcsr.out`
- 11/1/94 You can read input data from a file by using the `-i` program argument. You can use output files created by `-o` as input files.
 - e.g. `-i saprfcsr.in`
- 5/2/95 Definition of export parameters after RfcGetData and before RfcSendData.
- 8/11/95 Multi-function support. This feature allows to generate one RFC server example program for SAP Business API's that contain more than one function module. You can generate a RFC

Complete ANSI C Server Example

server example program for more than one function module - one after another.

Complete Visual Basic Client Example

SAP AG - R/3 Remote Function Call Interface Generation

RFC_MSVB_CL_EXP E 30A 22.09.1995 15:22

Complete Vis.Basic client exa.

saprfcze.txt

RFC_MSVB_CL_EXP : Complete Vis.Basic client exa.

Description

The generation form RFC_MSVB_CL_EXP generates a complete ready-to-run RFC client example program written in Visual Basic. This program lets you test your remote function module without manual programming.

The example program prompts the user to input exporting parameters before the call and to display importing parameters after the call. Dialogs for setting values in and displaying internal table objects are also provided.

The example program works with Microsoft Visual Basic (MSVB). It provides both a graphical user interface using Visual Basic screens and controls and a character user interface that supports automated tests for example. Help texts are available for each dialog step.

The example program contains type and variable definitions for all function module parameters. It also includes functions for logon, for init, input and display of export and import parameters, and for the handling of internal tables.

Files generated by this form

- saprfcze.mak
Visual Basic project file.
- saprfcze.frm
Output screen.
- saprfcze.bas
 - Function declarations for RFC API functions
 - Type definitions for structure parameters
 - SET/GET functions

Complete Visual Basic Client Example

- Logon dialog
 - RFC error function
 - Visual Basic client stub
 - Main program
 - Variable definitions for function module parameters
 - Init, input, display of export and import parameters
 - Handling of internal tables.
- saprfczc.frm
Start and call screen.
 - saprfczh.frm
Help, error and end screen.
 - saprfczl.frm
Logon and logoff screen.
 - saprfczs.frm
Structure screen.
 - saprfczt.frm
Table screen.
 - saprfcze.txt
This text.

Generation

You can generate RFC_MSVB_CL_EXP for one function module at a time. The generation of example programs for multiple function modules at a time is not recommended, because names for generated subroutines will not be unique. When generating example programs, download and clear the code for one function module before going on to the next.

Making an EXE file

You can load the generated Visual Basic example program by opening the makefile saprfcze.mak from inside Visual Basic. Then you can run the example within Visual Basic or you make an EXE file by choosing the 'Make EXE file' menu item from the 'File' menu.

Dependencies

- Files you need from the RFC SDK for WINDOWS

Complete Visual Basic Client Example

- librfc16.dll, librfc2.dll, nidll.dll, dptrcerr.dll

Dynamic Link Libraries that provide the RFC API functions
(WINDOWS 16-bit)

Note:

These files should be in the same directory as your generated example program. The dynamic link libraries may also be stored in another directory, but this directory must be defined in the WINDOWS path variable.

Test support

The example program provides support to automate tests. Use program argument `-o <file name>` to record a test, edit the recorded file if necessary and play it back by using program argument `-i <file name>`. You can use `-o` and `-i` at a time. Verify your test results by comparing the input and output file. Use program argument `-?` to get help on all supported program arguments.

Tested development systems

- Microsoft Visual Basic 3.0

Warnings

- The length of generated source code lines is limited to 255 characters. A large number of parameters can cause errors, because MSVB 3.0 has no line-continue character.
- Statements longer than 255 characters are split into two or more lines. You can concatenate splitted statements again by removing the return character.



Repair function declarations carefully. When removing the return character of a function declarations the function header is recognized, but the function body remains in the declaration window.

To solve this problem, cut the function body from the declaration window, repair the function declaration, then paste the function body into the function declaration window.

- To convert the input string to an integer parameter, `var = CInt(s)` is used. The `CInt` function always rounds to the nearest even number. For example, 0.5 rounds to 0, and 1.5 rounds to 2 (see MSVB online help also).

Complete Visual Basic Client Example

- Having a field named 'Line' in a structure or table causes a Visual Basic syntax error. To solve this problem rename this field in all Visual Basic modules to 'XLine' for example.
- Don't use string variables with a certain length in MSVB 3.0: e.g. Dim myString * 10. Using string variables with a certain length can cause address errors when calling a DLL function.
- Compatibility issues for Visual Basic for Application
- MS Word 6.0 lacks support for structures known as VB types. You must use strings for structures and table rows.

Changes

- 11/1/94 Support for the following data types:
 - RFC_BCD : packed numbers
 - RFC_BYTE: hexadecimal fields
- 2/10/95 The complete Visual Basic client example now provides both a character user interface and a graphical user interface.
- 2/10/95 The graphical user interface comes up when starting the example without program arguments. You can switch from within the graphical user interface to character mode by pressing the 'Options' button on the start screen.
- 2/10/95 The character user interface comes up automatically when starting the example with program argument -y and/or any other allowed program argument.
- 5/2/95 Adding parameters immediately after parameter definition. Definition of import parameters after RfcCallExt and before RfcReceiveExt.
- 5/2/95 Support for new naming convention of WINDOWS RFC libraries:
librfc16.dll used instead of librfc.dll.

Code has been generated for the following function module

- ABAP4_CALL_TRANSACTION
Executes the ABAP command CALL TRANSACTION.

Complete Visual Basic Server Example

```
SAP AG - R/3 Remote Function Call Interface Generation
RFC_MSVB_SR_EXP E 30A 22.09.1995 15:22
Complete Vis.Basic server exa.
saprfcvr.txt
```

RFC_MSVB_SR_EXP : Complete Vis.Basic server exa.

Description

The generation form RFC_MSVB_SR_EXP generates a complete ready-to-run external RFC server example program written in Visual Basic. This program lets you test external functions without manual programming.

You can start your external RFC server from the R/3 function library testbed by specifying an appropriate RFC destination.

The example program displays importing parameters and prompts the user to raise an exception or to input exporting parameters. Dialogs for displaying and setting values in internal table objects are also provided.

The example program works with Microsoft Visual Basic (MSVB). The user interface provides a Visual Basic screen for output. Help texts are available for each dialog step.

The example program contains type and variable definitions for all function module parameters. It also includes functions for init, input and display of importing and exporting parameters, and for the handling of internal tables.

Files generated by this form

- saprfcvr.mak
Visual Basic project file.
- saprfcvr.frm
Output screen.
- saprfcvr.bas
 - Type definitions for field and structure parameters
 - Help function

Complete Visual Basic Server Example

- SET/GET macros and functions
 - RFC error function
 - External function
 - Init, input, display of import and export parameters
 - Handling of internal tables
 - Server stub
 - Variable definitions for external function parameters
 - Main program.
- saprfcvr.txt
This text.

Generation

You can generate RFC_MSVB_SR_EXP for one function module at a time. The generation of example programs for multiple function modules at a time is not recommended, because names for generated subroutines will not be unique. When generating example programs, download and clear the code for one function module before going on to the next.

Making an EXE file

You can load the generated Visual Basic example program by opening the makefile saprfcvr.mak from inside Visual Basic. Then you can run the example within Visual Basic or you make an EXE file by choosing the 'Make EXE file' menu item from the 'File' menu.

Dependencies

- Files you need from the RFC SDK for WINDOWS
 - librfc16.dll, librfc2.dll, nidll.dll, dptrcrr.dll
- Dynamic Link Libraries that provide the RFC API functions (WINDOWS 16-bit)



These files should be in the same directory as your generated example program. The dynamic link libraries may also be stored in another directory, but this directory must be defined in the WINDOWS path variable.

Complete Visual Basic Server Example**Test support**

The example program provides support to automate tests. Use program argument `-o <file name>` to record a test, edit the recorded file if necessary and play it back by using program argument `-i <file name>`. You can use `-o` and `-i` at a time. Verify your test results by comparing the input and output file. Use program argument `-?` to get help on all supported program arguments.



When defining a RFC destination for an external RFC server (SM59) in the current R/3 release you cannot specify additional program arguments like `-o` or `-i` for the RFC server program. You have to use a shell script or procedure to add additional program arguments.

Tested development systems

- Microsoft Visual Basic 3.0

Warnings

- The length of generated source code lines is limited to 255 characters. A large number of parameters can cause errors, because MSVB 3.0 has no line-continue character.

Statements longer than 255 characters are splitted into two or more lines. You can concatenate splitted statements again by removing the return character.



Repair function declarations carefully. When removing the return character of a function declarations the function header is recognized, but the function body remains in the declaration window.

To solve this problem, cut the function body from the declaration window, repair the function declaration, then paste the function body into the function declaration window.

- To convert the input string to an integer parameter, `var = CInt(s)` is used. The `CInt` function always rounds to the nearest even number. For example, 0.5 rounds to 0, and 1.5 rounds to 2 (see MSVB online help also).
- Having a field named 'Line' in a structure or table causes a Visual Basic syntax error. To solve this problem rename this field in all Visual Basic modules to 'XLine' for example.

Complete Visual Basic Server Example

- Don't use string variables with a certain length in MSVB 3.0:
e.g. Dim myString * 10. Using string variables with a certain length can cause address errors when calling a DLL function.
- Compatibility issues for Visual Basic for Application
- MS Word 6.0 lacks support for structures known as VB types. You must use strings for structures and table rows.

Changes

- 11/1/94 Support for the following data types:
 - RFC_BCD : packed numbers
 - RFC_BYTE: hexadecimal fields
- 5/2/95 Adding parameters immediately after parameter definition. Definition of export parameters after RfcGetDataExt and before RfcSendDataExt.
- 5/2/95 Support for new naming convention of WINDOWS RFC libraries:
librfc16.dll used instead of librfc.dll.

Code has been generated for the following external function

- ABAP4_CALL_TRANSACTION
Executes the ABAP command CALL TRANSACTION.

Complete OLE Automation Client Example

```
SAP AG - R/3 Remote Function Call Interface Generation
RFC_OLEA_CL_EXP E 30A 22.09.1995 15:22
Complete OLE Auto. client exa.
saprfcae.txt
```

RFC_OLEA_CL_EXP : Complete OLE Auto. client exa.

Description

The generation form RFC_OLEA_CL_EXP generates a complete ready-to-run RFC client example program written in Visual Basic that uses the SAP Automation Server to perform function calls. This program lets you test your remote function module without manual programming.

The example program prompts the user to input exporting parameters before the call and to display importing parameters after the call. Dialogs for setting values in and displaying internal table objects are also provided.

The example program works with Microsoft Visual Basic (MSVB). It provides both a graphical user interface using Visual Basic screens and controls and a character user interface that supports automated tests for example. Help texts are available for each dialog step.

The example program contains type and variable definitions for all function module parameters. It also includes functions for logon, for init, input and display of export and import parameters, and for the handling of internal tables.



This Visual Basic example program uses the SAP Automation Server to perform remote function calls. So, to run this example you should have installed both the RFC libraries and the SAP Automation Server. See the user manual or the help file for more information about the SAP Automation Server.



The SAP Automation Server offers a named-argument convention and a non-named-argument convention for calling a function

Complete OLE Automation Client Example

module remotely. Because Visual Basic 3.0 does not support named arguments, the non-named-argument convention is used here.

Files generated by this form

- saprfcae.mak
Visual Basic project file.
- saprfcae.frm
Output screen.
- saprfcae.bas
 - Main program
 - Variable definitions for function module parameters
 - Init, input, display of export and import parameters
 - Handling of internal tables.
- saprfcac.frm
Start and call screen.
- saprfcah.frm
Help, error and end screen.
- saprfcas.frm
Structure screen.
- saprfcat.frm
Table screen.
- saprfcae.txt
This text.

Generation

You can generate RFC_OLEA_CL_EXP for one function module at a time. The generation of example programs for multiple function modules at a time is not recommended, because names for generated subroutines will not be unique. When generating example programs, download and clear the code for one function module before going on to the next.

Making an EXE file

You can load the generated Visual Basic example program by opening the makefile saprfcae.mak from inside Visual Basic. Then you can run the example within Visual Basic or you make an EXE file by choosing the 'Make EXE file' menu item from the 'File' menu.

Complete OLE Automation Client Example**Dependencies**

- Files you need from the RFC SDK for WINDOWS
 - librfc16.dll, librfc2.dll, nidll.dll, dptrcerr.dll
- Dynamic Link Libraries that provide the RFC API functions (WINDOWS 16-bit)



These files should be in the same directory as your generated example program. The dynamic link libraries may also be stored in another directory, but this directory must be defined in the WINDOWS path variable.

- The Visual Basic example program generated by RFC_OLEA_CL_EXP uses the SAP Automation Server to perform remote function calls.

Test support

The example program provides support to automate tests. Use program argument `-o <file name>` to record a test, edit the recorded file if necessary and play it back by using program argument `-i <file name>`. You can use `-o` and `-i` at a time. Verify your test results by comparing the input and output file. Use program argument `-?` to get help on all supported program arguments.

Tested development systems

- Microsoft Visual Basic 3.0

Warnings

- The length of generated source code lines is limited to 255 characters. A large number of parameters can cause errors, because MSVB 3.0 has no line-continue character.
- Statements that are longer than 255 characters are splitted into two lines or more than two lines if necessary. Concatenate splitted statements again by removing the return character.
- Having a field named 'Line' in a structure or table will raise a Visual Basic syntax error. To solve this problem rename this field in all Visual Basic modules to 'XLine' for example.

Changes

None

Complete OLE Automation Client Example

Code has been generated for the following function module

- ABAP4_CALL_TRANSACTION

Executes the ABAP command CALL TRANSACTION.

Calling RFC Stub Programs

Once you have downloaded the desired RFC stubs to the workstation or PC, you can call them from programs of your own. This section describes how to use these stubs, that is, how to compile them, what they do, and what your program must do in order to call them correctly. If you have generated example programs for the stubs, you can use these as a basis for programming your own application.

Compiling and Linking on the Workstation

Compiling and Linking on the Workstation

You must compile and link the downloaded programs before you can use them:

- For RFC stubs, compile as a new library or DLL file.
- For example programs, compile as an ANSI C application program.

In general, you have to use an ANSI C compatible C-compiler and you have to set the include and library search path to the installed RFC SDK include and lib directory.

At some platforms you also have to link the TCP/IP socket libraries explicitly.

Assume.../rfcsdk is the root directory of the unpacked RFC SDK. The following compile/link calls should be done for the program sapinfo.c on different platforms.

- HP-UX
`cc -Ae -I.../rfcsdk/include -L.../rfcsdk/lib sapinfo.c librfc.a`
- AIX (RS/6000)
`ccc -I.../rfcsdk/include -L.../rfcsdk/lib sapinfo.c librfc.a`
- SINIX (RM600)
`/opt/C/bin/cc -I.../rfcsdk/include -L.../rfcsdk/lib sapinfo.c librfc.a -lsocket -lnsl -lusc`

(only this compiler is supported.)

- DEC Alpha AXP
`cc -std1 -unsigne -DA_OSF -I.../rfcsdk/include -L.../rfcsdk/lib sapinfo.c librfc.a`
- SUN (SunPro)
`/opt/SUNWspro/bin/cc -Xc -xcg92 -I.../rfcsdk/include -L.../rfcsdk/lib sapinfo.c librfc.a -lsocket -lnsl`
 (only this compiler is supported.)
- WINDOWS 3.1 and higher
`cl /nologo /Gs /G2 /W4 /AL /D "_DOS" /Od /D "_DEBUG" /Mq /Fesapinfo.exe sapinfo.c librfc16.lib /link`
- WINDOWS NT 3.5 and higher
`cl -nologo -Od -G5 -Z7 -Gs -W3 -J -D_X86_ -DWIN32 /MT /FR -Fesapinfo.exe sapinfo.c librfc32.lib`
- OS/2 2.1 and higher
`icc -c -Gs- -Kb+ -Sm -Q -Gm+ -Ss+ -O- -Ti+ -Gt- -DOS2 sapinfo.c link386 /NOI /CO /PM:VIO /ST:16384 /PACKD %1,,librfc;`



With many Windows compilers, you must set certain compiler options. For the following options, specify the values:

- Memory model: Large

Compiling and Linking on the Workstation

- Preprocessor: SAPonWINDOWS

For DLL projects, a large memory model is the default. However, for application projects, the default memory model is sometimes "medium", and in these cases, it must be changed to "large".

With UNIX systems, the memory model is not important, and the macro SAPonWINDOWS should not be specified as the preprocessor option.

Tasks Performed by the RFC Stub

Tasks Performed by the RFC Stub

The RFC stub generator creates two files for each stub:

- the stub program *saprfcc.c*
- header file *saprfcc.h*

All RFC stub programs perform the same basic functions:

- define the types needed for passing parameters
- fill the parameter structures with the individual importing, exporting and tables parameters
- make the RFC call needed (*RfcCall*) to call the function module
- make the RFC call needed (*RfcReceive*) to get return parameter values

The *saprfcc.h* header file contains standard RFC definitions, as well as those needed by the specific function modules.



The function module declaration itself contains the macro calls, *win_export*, *win_far*, *win_loadds*, *win_pascal*. These macros insert code needed only in Windows-based systems; in UNIX systems, the macros resolve to blanks.

To see sample stub code, generate an RFC stub function and display the code.

Tasks Performed by a Stub Caller

To call an RFC stub successfully, an application program must perform specific tasks. The example programs provided by the RFC Interface Generator show how to program these tasks, and can be created in C or in Visual Basic.

You can use these example programs as a basis for developing your own application, or you can write stub callers from scratch. In either case, your program must perform the following specific tasks:

- call *RfcOpen* to establish a connection with the SAP System
- call *ItCreate* (if you are passing table parameters) to create a control structure for each table to be passed
- call the actual stub
- call *RfcClose* to terminate the connection



RFC does not use null-terminated strings ('\0' at the end). Parameters are passed with their original length.

To see sample caller code, generate an example program and display the code (described in [Creating RFC Stubs and Example Programs \[Seite 9\]](#)).