

Connect for SAP® (Classic)



Getting Started

1	About this Document	3
2	About Connect for SAP® (Classic)	4
3	Architectural overview	5
3.1	RFC Function Architecture	6
3.1.1	Data Representation	6
3.1.2	Data Mapping	7
3.1.3	Early and Late Function Binding	10
3.2	Client Applications	12
3.2.1	Using Connection Aliases	13
3.2.2	Features for Transactional Calls	13
3.3	Server Applications	14
3.3.1	Specifying Registration Parameters	15
3.3.2	Features for Transactional Calls	15
4	Installation	16
4.1	System Requirements	16
4.2	Installing the RFC Library Binaries	16
4.2.1	SAPGUI installation	16
4.2.2	Compact Installation	16
4.2.3	Full Installation	18
4.3	Additional Requirements	19
4.4	Installing into Embarcadero Delphi or C++ Builder	20
4.4.1	Building Connect for SAP® Binaries	20
4.4.2	Installing Components	20
5	Connect for SAP® Explorer	21
5.1	Creating and Maintaining Aliases	21
5.1.1	Creating or Opening a Connection Alias File	21
5.1.2	Creating a New Connection Alias	22
5.1.3	Modifying a Connection Alias	22
5.2	Browsing SAP Dictionary Information of RFC Functions	24
5.2.1	Displaying RFC Functions	24
5.2.2	RFC Objects Definition Information	25
5.3	Executing RFC Functions	26
5.4	Generating Wrapping Code for RFC Function	26
5.4.1	Generating Wrapping Code	27
5.4.2	How to Use Generated Code	28
6	Troubleshooting	29
6.1	Issue report	29
6.2	Tracing	29
6.2.1	SAP RFC library tracing	29
6.2.2	“Connect for SAP”® tracing	29
Appendix A – Defining Server Parameters		30
Appendix B – Transaction Management in Connect for SAP® Server Application		31
Appendix C – Connect for SAP® Component List		32

1 About this Document

This document might be very useful for Embarcadero Delphi developers in:

- Building applications that are SAP system clients;
- Extending functionality of a SAP application server by creating external non-SAP server programs.

You can find in this guide a general overview of the Connect for SAP® software and its possible applications. This document helps to understand main architectural concepts of the Connect for SAP® work: information on the RFC function architecture, different types of the data mapping and the function binding. You will also learn general concepts of creating client and server applications based on Connect for SAP®. The guide provides the developer with necessary installation instructions and gives a brief overview of components installed.

If you need to get any additional information not mentioned in this guide do not hesitate to contact us:

Web: <http://www.gs-soft.com/CMS/en/products/connect-for-sap-sapx.html>

By email: sapx@gs-soft.com

2 About Connect for SAP® (Classic)

Connect for SAP® is an object-oriented software library. It has been specially designed for an access to SAP application servers using Embarcadero Delphi™ and C++ Builder™ for building partner server programs run in non-SAP systems. Connect for SAP® is a flexible and versatile tool for:

- An integration of existing Delphi™ applications with SAP systems. This feature allows corporations to use their own information systems and create superstructures with additional opportunities;
- A development of new systems and applications that have an access to a SAP application server as clients;
- An extension of SAP system functionality through Connect for SAP® by building external non-SAP servers. This feature gives the developer an opportunity to avoid costs connected with the ABAP training as all the functionality extensions are implemented in Delphi™ programs.

Connect for SAP® encapsulates a Remote Function Call (RFC) interface and offers high-level software components and classes.

RFC API is a set of C-language routines that perform certain end user's communication tasks and allow an execution of remote calls between two SAP Systems or between a SAP System and a non-SAP system.

RFC API supports a number of external systems, such as OS/2, Windows, as well as all of R/3-based UNIX platforms. This feature makes it possible to use the RFC functionality for an interaction of a SAP System with a C-program based on the platforms mentioned above (there exists a RFC SDK that includes a RFC library specific for each platform supported).

3 Architectural overview

On **Figure 1** you can see the way Delphi applications can interact with a SAP system through Connect for SAP®. Connect for SAP® can be used both in client and server applications.

In the first case, when the developer wants to call an ABAP function he has to use the Connect for SAP® object methods and properties. Connect for SAP® packs all the necessary data and transfers the call to the RFC library. In such a way the client request is sent to the SAP system. On receiving the request the SAP application server processes it and returns the result. Connect for SAP® gets the resulting data from the RFC library and the developer can have the access to it.

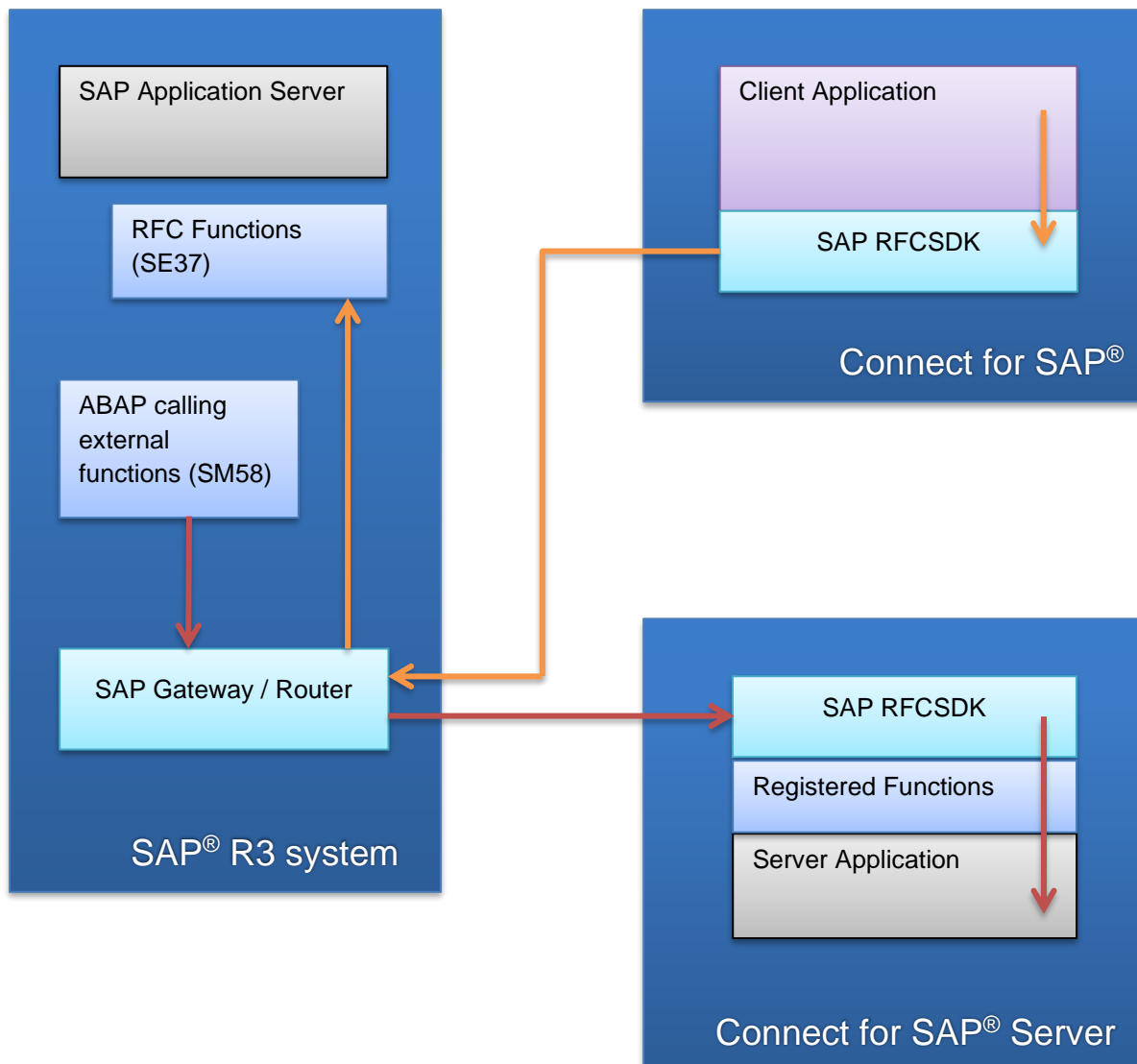
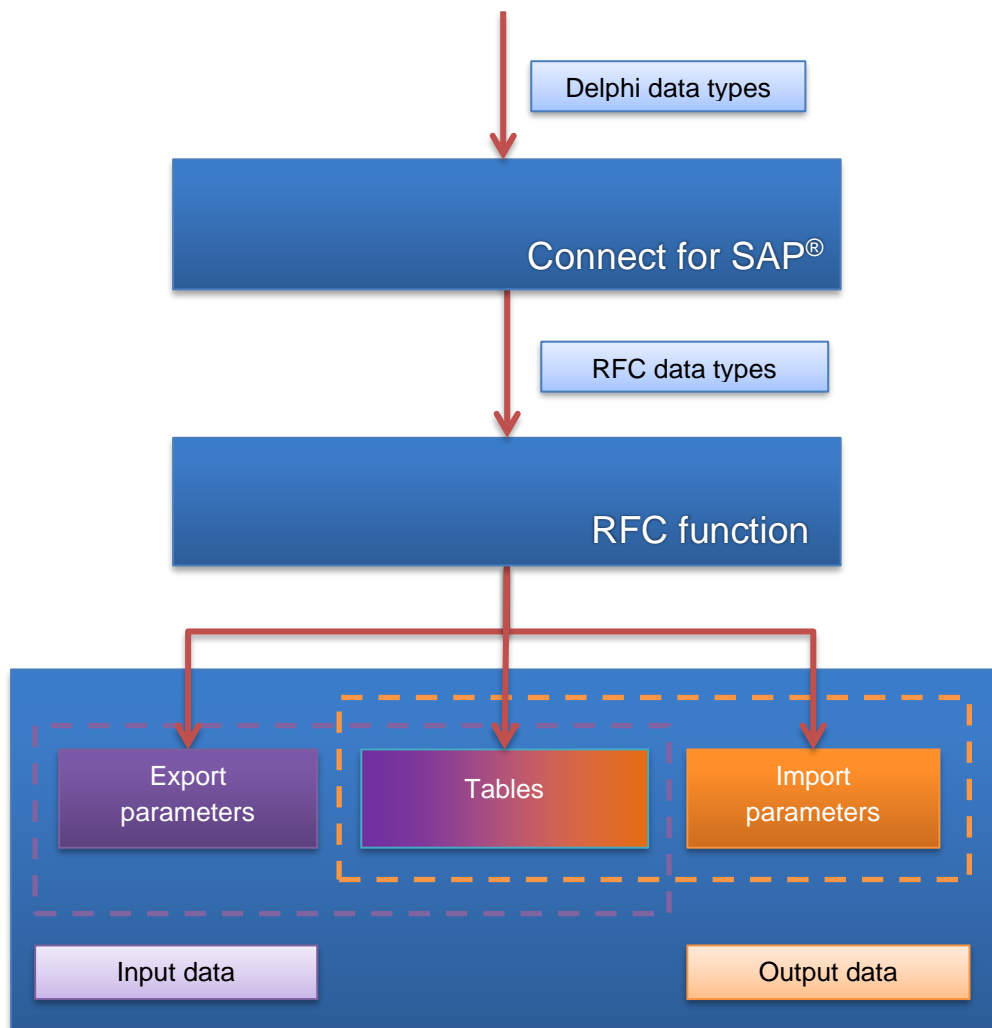


Figure 1: Interaction of a SAP system with Delphi application based on Connect for SAP®

In the second case the Connect for SAP® server application is constantly waiting for the SAP system client request. When the request is occurred Connect for SAP® receives and processes it. Connect for SAP® also undertakes to send the result to the SAP system in the correct format.

3.1 RFC Function Architecture

If you want to understand the way a RFC function can be called and how to work with the function parameters it is necessary to examine RFC function architecture.



As it is shown on the figure above, RFC function receives data from Export parameters; Import parameters contain resulting data; whereas Tables can contain both input and output data. All data imported from and exported to RFC function has its own format and internal order. These data formats, RFC data types, differ from Delphi ones. That is why one of the Connect for SAP® most important tasks is to map RFC data types to Delphi ones and backwards.

3.1.1 Data Representation

SAP R/3 servers are able to run on different types of computers. And they may have different than on WinTel representations of integer and float data. The data representation should be changed, when the data are received from / transmitted to a SAP R/3 server and data representations of the server and the client are different. Connect for SAP® performs that for you.

How Connect for SAP® will do that is controlled by the alias parameters `TSAPxRFCALiasGS.DataFormat.IntType` and `TSAPxRFCALiasGS.DataFormat.FloatType`. By default they have the values `itAutoDetect` and `ftAutoDetect`.

The alias parameter `TSAPxRFCALiasGS.DataFormat.BytesPerChar` specifies the server side character data representation – bytes per char. By default it has the value `bcAutoDetect`.

Using the default values, Connect for SAP® will automatically detect the server side data representation. In some special cases, you can decide to force Connect for SAP® to expect some specified data representation. It is not recommended, although.

3.1.2 Data Mapping

RFC data types can be divided into three groups with different mapping methods: simple data type, structured data type and tables.

3.1.2.1 Simple Data Types

Figure 2 shows concepts of a simple RFC data type mapping. If a data type has an ambiguous mapping, the developer can definitely indicate the target Delphi data type. Otherwise, Connect for SAP® maps this data type to the most appropriate Delphi data type.

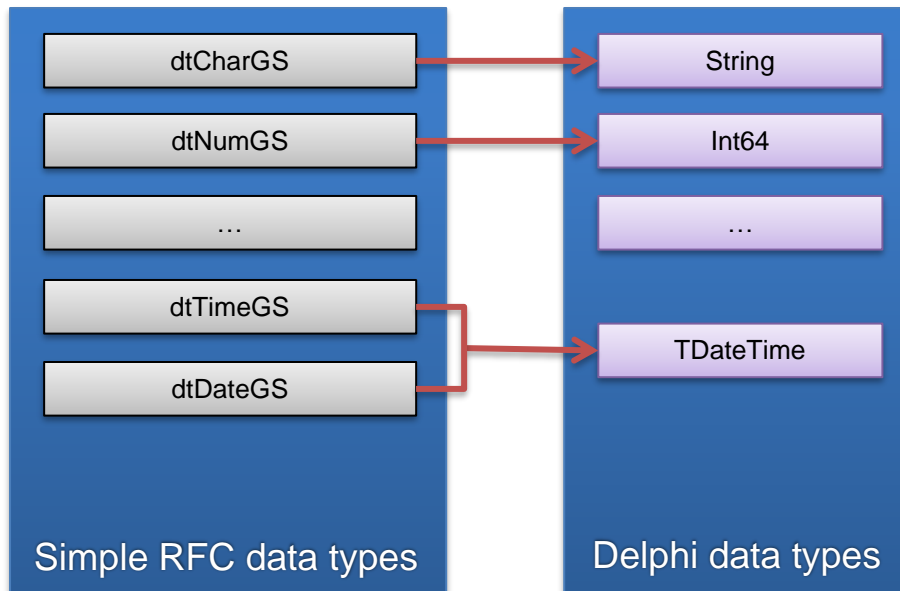


Figure 2: Simple RFC data type mapping

The next table shows the supported mapping of simple RFC data types to Delphi data and field types:

RFC data type	Delphi data type	Delphi field type
dtCharGS	<u>Pre Delphi 2009</u>	
	AnsiString	ftString (size <= 8192)
		ftMemo (size > 8192)
	<u>Delphi 2009 and higher</u>	
	UnicodeString	ftWideString (size <= 8192)
		ftWideMemo (size > 8192)
dtNumGS	Int64	ftSmallInt (size <= 4)
		ftInteger (size <= 9)
		ftLargeInt (size > 9)
dtByteGS	RawByteString	ftVarBytes (size <= 8192)
		ftBlob (size > 8192)
dtBCDGS	Integer	ftFloat
	Int64	
	Currency	
	Double	
	String	
dtIntGS	Integer	ftInteger
dtInt1GS	ShortInt	ftSmallInt
dtInt2GS	SmallInt	ftSmallInt
dtFloatGS	Double	ftFloat
dtDateGS	TDateTime	ftDate
dtTimeGS	TDateTime	ftTime
dtStringGS	UnicodeString	ftWideString (size <= 8192)
		ftWideMemo (size > 8192)
dtXMLDataGS	UnicodeString	ftWideMemo
dtXStringGS	RawByteString	ftBlob
dtLineTypeGS	Not supported	Not supported

3.1.2.2 Character Data Types

There are two character-based RFC types that used to be mixed sometimes: dtCharGS and dtStringGS.

Parameter (or field) of dtCharGS type has a fixed length. In case of a dynamical evaluation of parameters (see 3.1.3) the length is obtained from SAP System. In case of a static parameters definition (when Programmer uses RFC Wrapper or adds parameters/fields manually) the length must explicitly be defined by Programmer using property "CharacterSize". During transfer of the parameter, unused characters will be filled by "space" characters.

Parameter (or field) of dtStringGS type has a variable length but not exceeding 8192 bytes (or 4096 characters for Unicode SAP systems). Only filled characters will be transferred to SAP System (no padding with "spaces" as for dtCharGS type). In case of a static parameters definition, Programmer can additionally restrict the maximal length by setting the property "CharacterSize".

In case Programmer mixes these parameter types (e.g. manually defines a parameter as dtStringGS while it is actually defined on SAP System as dtCharGS), the data can be transferred incorrectly by the RFC Library.

So, please make sure you use parameter types as they defined on your SAP System.



Hint: Use Connect for SAP® Explorer to generate code wrappers for your RFC functions.

3.1.2.3 Structured Data Types

Unlike a simple data type, structured one, i.e. dtStructureGS, does not have Delphi analogues. **Figure 3** illustrates how the Connect for SAP® wraps the dtStructureGS type by means of the TSAPxRFCParameterGS class, which contains a field list of the TSAPxRFCFieldsListGS class. So the structure corresponds to the field list, where an individual field of the TSAPxRFCFieldGS class represents each structure item.

Connect for SAP® does not support nested structured data types. It means that each structure item should be of a simple data type.

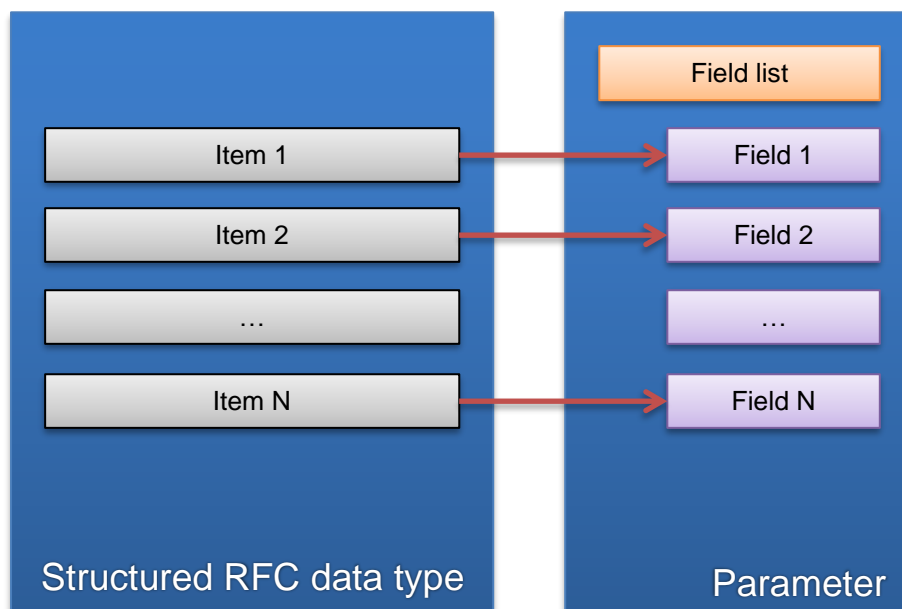


Figure 3: Wrapping structured RFC data type by Connect for SAP®

Connect for SAP® includes the TSAPxRFCvParamsGS component derived from the TDataSet that, on the one hand, offers a clear and easy interface for Delphi developers and, on the other hand, works with the RFC library using the RFC data types and formats. TSAPxRFCvParamsGS represents each function simple parameter by a single field, and each structured parameter by one top level field with subfields. You can choose which parameter types (input, output or both) TSAPxRFCvParamsGS includes by specifying ParamKinds.

The next table shows conversion of dtStructureGS data type to Delphi field type:

RFC data type	Delphi data type	Delphi field type
dtStructureGS	<pre> graph TD A[TSAPxRFCParameterGS] -- "1 ... 1" --> B[TSAPxRFCFieldsListGS] B -- "1 ... N" --> C[TSAPxRFCFieldGS] </pre>	ftADT

3.1.2.4 RFC Table Parameters

We should also pay more attention to the way Connect for SAP® works with function tables featuring their own format. SAP RFC table parameter is like a structured parameter – it has a field list, but may contain multiple rows of data.

Connect for SAP® includes the TSAPxRFCvTableGS component derived from the TDataSet that, on the one hand, offers a clear and easy interface for Delphi developers and, on the other hand, works with the RFC library using the RFC data types and formats. TSAPxRFCvTableGS corresponds to one table parameter with TableName name.

The next table shows converting RFC table parameters to Delphi data type:

RFC data type	Delphi data type	Delphi field type
RFC table parameter	TSAPxRFCTableGS	TSAPxRFCvTableGS

3.1.2.5 Unicode Character Data

The Unicode support in Connect for SAP® depends on the Delphi version:

- All Delphi versions prior Delphi 2009 does not support the Unicode version of the RFC dynamic library (see **Installing the RFC Library** for details). This means, that for the CHAR data type and for the object names, the library always uses the ANSI encoded character data. Even if a server is in the Unicode mode.
- Delphi 2009 and higher supports the Unicode version of the RFC dynamic library (see **Installing the RFC Library** for details). This means, that for the CHAR data type and for the object names, the library always uses the Unicode UCS-2 encoded character data.

The STRING and XMLDATA data types are always UTF8 encoded. For these data types, the library internally performs the UTF8 <-> UCS2 transformation and always returns the UCS2 encoded character data. Connect for SAP® supports the UTF8 encoded character data properly with the version 6 of the RFC library or higher.

3.1.3 Early and Late Function Binding

There are two types of binding ABAP RFC functions with Connect for SAP® function objects in Connect for SAP®, early and late binding.

The early binding means that an ABAP function name has been known at design time already. So the Connect for SAP® function object is statically defined. It is recommended to use the Connect for SAP® Explorer tool to generate a wrapping code for the ABAP functions. That will save you a lot of time and will help you to avoid lots of mistakes. See

Generating Wrapping Code for RFC Function for details and restrictions applied to the code generation.

The next listing shows an example of the generated wrapping code:

Listing 1: Early binding. Wrapping code generated for RFC_READ_TABLE ABAP function by Connect for SAP® Explorer.

```
TSAPxRFCRFC_READ_TABLEFuncGS = class(TSAPxRFCFunctionGS)
private
  procedure SetDELIMITER(const AValue: String);
  function GetDELIMITER: String;
  function GetDATA: TSAPxRFCTAB512TableGS;
public
  constructor Create; override;
  property DELIMITER: String read GetDELIMITER write SetDELIMITER;
  property DATA: TSAPxRFCTAB512TableGS read GetDATA;
end;
```

On the contrary, the late binding allows the developer to call an ABAP function at runtime dynamically. In this case Connect for SAP® automatically gets the necessary metadata. The next listing shows an example of the late binding:

Listing 2: Late binding. Using a dynamically prepared function.

```
procedure Execute;
begin
  // working with TSAPxRFCFunctionGS object interface
  with FCFUNCTION do begin
    ObjName := 'RFC_READ_TABLE';
    Prepared := True;
    InParams.ParameterByName('DELIMITER').AsString := '%';
    ExecFunction;
    with Tables.TableByName('DATA') do begin
      { do something with table 'DATA' }
    end;
  end;
end;
end;
```

The main differences between the early and the late binding:

- The early binding has a higher productivity as it excludes application round trip to the SAP system for metadata retrieval;
- The early binding allows to use the Code Insight feature and find some mistakes during compile process;
- The early binding is sensitive to the client and the server Unicode mode, due to the differences in structure layouts for different Unicode modes;
- The late binding has smaller performance but wider flexibility;

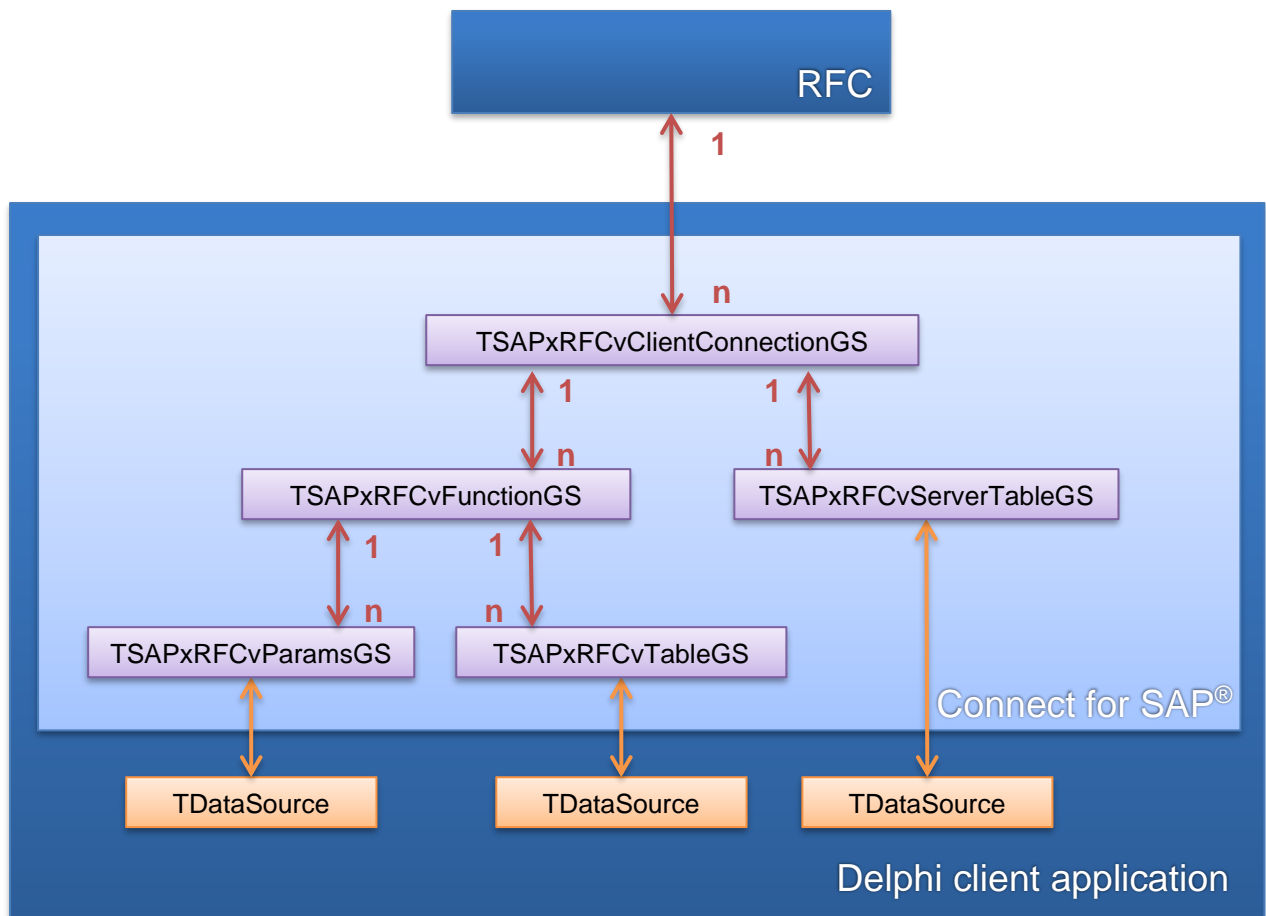
The developer can choose a binding mode depending on a specific task.



Hint: early binding can improve the execution speed significantly for methods sequentially called many times per second.

3.2 Client Applications

The next figure illustrates the architecture of a client application build with the Connect for SAP® components using Embarcadero Delphi.



The `TSAPxRFCvClientConnectionGS` component is responsible for connection to a SAP server. Use the `Params` property to specify connection parameters on the fly or the `AliasName` property to use a predefined connection alias. Set `Connected` to `True` to establish the connection. Such components as `TSAPxRFCvFunctionGS` and `TSAPxRFCvServerTableGS` use the `TSAPxRFCvClientConnectionGS` object to communicate with the RFC library.

The main Connect for SAP® client component is `TSAPxRFCvFunctionGS`. It is responsible for describing and executing of SAP functions using the SAP RFC library. Set the `Connection` property to a connection component. Use the `ObjName` property to specify a function name to call. The `OutParams` property represents an output parameter collection, the `InParams` – input ones, the `Table` – table ones. These properties are not accessible at design time.

The developer can use the `TSAPxRFCvParamsGS` and `TSAPxRFCvTableGS` components to operate with `TSAPxRFCvFunctionGS` function parameters and tables. Set the `Func` property to a function component. Use `TSAPxRFCvParamsGS.ParamKinds` to specify parameter types (input, output or both) to represent. Use `TSAPxRFCvTableGS.TableName` to specify table parameter to represent.

The components `TSAPxRFCvParamsGS`, `TSAPxRFCvTableGS`, and `TSAPxRFCvServerTableGS` inherited from `TDataSet` can be linked with any data aware controls.

To build a client application you can use components as well as objects encapsulated into these components, e.g. the `RFCFunction` property of `TSAPxRFCvFunctionGS`.

3.2.1 Using Connection Aliases

A client application establishes a communication with a SAP system through the SAP RFC library. The connection is defined by a set of parameters, which has to be specified before connecting.

For convenience developers may use Connect for SAP® aliases to define the connection parameters. An alias is a named stored set of the parameters. By default the aliases are stored in the `%Windows%\SAPxRFCAliases.ini` file.

The Connect for SAP® Explorer tool is used to maintain the aliases and test them (see **Connect for SAP® Explorer** for details).

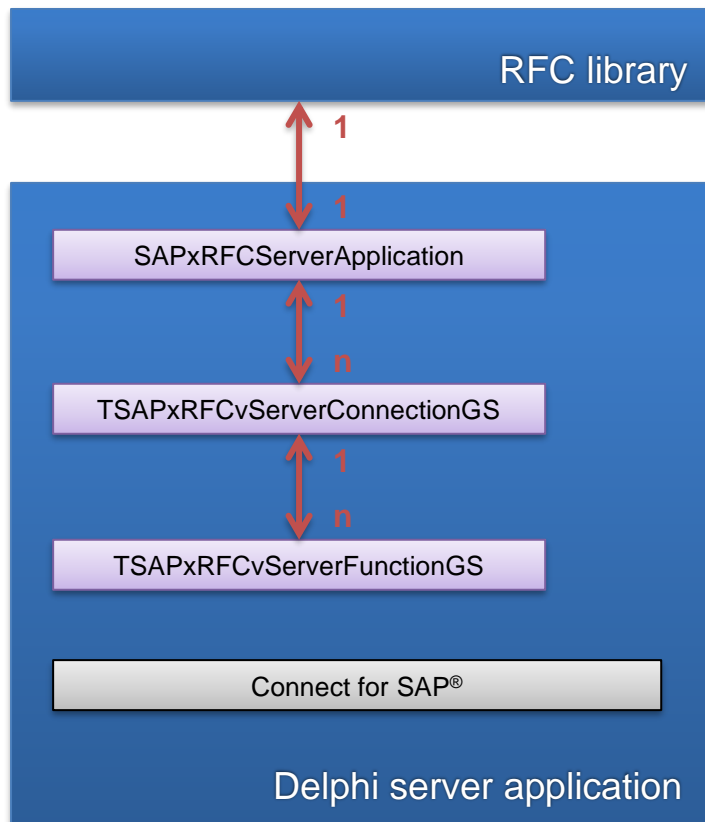
3.2.2 Features for Transactional Calls

Connect for SAP® supports transactional functions. The transactional function should be called only between starting and ending points of a transaction. These are distinctive features of transactional calls:

- CallType property of TSAPxRFCvFunctionGS should be set to ftTransactionalGS.
- TSAPxRFCvFunctionGS should have no import parameters.
- Only one function can be called within a singular transaction.

3.3 Server Applications

The next figure illustrates the architecture of a server application build with the Connect for SAP® components using Embarcadero Delphi.



The TSAPxRFCvServerConnectionGS component is responsible for a registration of a server on a gateway and a communication of the TSAPxRFCvServerFunctionGS components with a SAP system through the SAP RFC library. Use the `CommandLine` property to specify registration parameters. Set `Connected` to `True` to establish the registration. All requests to the functions registered with this connection are handled in a single thread in a serialized fashion.

The main Connect for SAP® server component is TSAPxRFCvServerFunctionGS. It is responsible for installing description and execution of requests to a custom RFC function using the SAP RFC library. Set the `Connection` property to a server connection component. Use the `ObjName` property to specify the function registration name. Create an `OnExecute` handler, which will handle the custom function request. The `OutParameters` property represents an output parameter collection, the `InParameters` – input ones, the `Tables` – table ones. These properties are not accessible at design time.

And the SAPxRFCServerApplication is a singleton, controlling the Connect for SAP® server application life cycle. All the TSAPxRFCvServerConnectionGS and TSAPxRFCvServerFunctionGS objects must be created and setup before calling `SAPxRFCServerApplication.Start`. The method creates a thread for each server connection. Then each thread registers its server connection at the gateway, installs a transaction control and installs all associated with this connection custom functions. Now the Connect for SAP® server application is able to handle requests from external SAP systems.

The `SAPxRFCServerApplication.Shutdown` method stops the Connect for SAP® server application.

Server applications as well as client ones can be built on both Connect for SAP® components and objects encapsulated into these components.

3.3.1 Specifying Registration Parameters

The server connection parameters can be specified in the command line when the server application is starting. In this case the command line parameters are automatically assigned to the `CommandLine` property of the `TSAPxRFCvServerConnectionGS` component. **Appendix A** shows the command line switches and their meaning. When the server starts it becomes possible to specify either `PROGRAM_ID`, `GWHOST`, `GWSERV` and `RFC_TRACE` parameters or just a `DESTINATION` parameter solely.

In the second case you have to define the entry named `DESTINATION` in the `saprfc.ini` file specifying all connection parameters (see example in **Appendix A**). This way to specify the server connection parameters is much more flexible than the first one.

While using the command line it is very important to remember that you cannot specify more than one set of the server connection parameters. So, for server applications with multiple connections the developer should definitely and explicitly specify the `CommandLine` property of the `TSAPxRFCvServerConnectionGS` component.

3.3.2 Features for Transactional Calls

Connect for SAP® supports transactional server functions. You can use the transactional RFC to bundle several remote functions into one logical unit of work (LUW) (with an automatic rollback mechanism in case of error). With the transactional RFC, generated LUW's are processed independently of each other. This means, the order in which they are processed is not always the order in which they are generated. Check [the SAP help page](#) for more details.

4 Installation

4.1 System Requirements

Before installing Connect for SAP® ensure that:

- The SAP RFC non-Unicode library version 6 or higher is installed on your PC. It is required for Delphi versions prior Delphi 2009.
- The SAP RFC Unicode library version 6 or higher is installed on your PC. It is required for **Delphi 2009** and higher versions.



The SAP RFC library may be installed as a part of the SAP GUI installation.

The SAP RFC library from SAP GUI v 6.x has a bug – each unload of librfc32.dll will result a memory loss around 6 Mb.

UTF8 character data is supported properly by the SAP RFC library from the SAP GUI v 6.x or higher.

- The SAP R/3 system you want to work with should be Release 2.1 or newer.
- Embarcadero Delphi™ 5 / 6 / 7 / 2005 / 2006 / 2009 / 2010 / XE... 10.4 Sydney or Embarcadero C++Builder™ 6 / 2006 / 2007 / 2009 / 2010 / XE... 11 is installed on your PC.



Only Delphi 2009 and higher offers full Unicode support.

Native code is supported for both 32-bit and 64-bit Windows.

4.2 Installing the RFC Library Binaries

Internally Connect for SAP® uses SAP Classic RFC library. There are 3 approaches to installing the RFC library files on a workstation:

- The full installation of the **SAPGUI** contains already the necessary SAP RFCSDK files. No further actions are necessary (beside the potential deployment of the x64 environment).
- Compact installation: All the required SAP RFC runtime DLL's are copied from one workstation, where they are already installed, to another one.
- Full installation: The SAP RFC runtime DLLs installer and Microsoft VC++ Runtime installer are used to guarantee, that all the required SAP RFC runtime files are correctly deployed and installed on a workstation.

4.2.1 SAPGUI installation

This is the normal way that Workstations are prepared within SAP environments. If such a SAPGUI installation is up-to-date then your “Connect for SAP application” should run out of the box (with the exception of x64 executes).

4.2.2 Compact Installation

In general, you will need to copy the few SAP RFC runtime DLL's to the system folder (see **Table 1** for details). Also, it may be required to install the Microsoft VC++ Runtime DLL's. The approach assumes that they are rather already installed a workstation.

There are non-Unicode and Unicode versions of the Classic RFC library. To install the non-Unicode library it is need to copy only the single file

- librfc32.dll

The Unicode version requires that the following set of files to be copied to the system folder according to **Table 1**

- librfc32u.dll
- libsapu<versionVC>.dll
- libsapucum.dll
- icudt<version>.dll
- icuin<version>.dll
- icuuc<version>.dll

where <version> and <versionVC> variable parts, which can differ depending on the version of RFC library. E.g. the files have the next names icudt30.dll, icuin30.dll, icuuc30.dll, libsapu16vc80.dll.



Hint: To correctly work with Connect for SAP® the RFC library has to have version 6 or higher.

Since the version 7.20 of the RFC library, SAP provides the RFC library files for both 32-bit and 64-bit environments. The platform version of these files must correspond to the version of Connect for SAP® your application works with. Connect for SAP® searches for the SAP RFC library DLLs of the required version using the next rules:

- Search for the SAP RFC DLLs of the required version by the paths defined by the Environment Variable *PATH* starting from the first path in the list. As soon as the appropriate DLL is found, it will be loaded by Connect for SAP®.
- If the SAP RFC DLLs of the required version are not found in the previous step, Connect for SAP® tries to find them in standard locations in dependence of OS and the application platform version as shown in the next table.

Application version	32-bit Windows	64-bit Windows
32-bit application (use 32-bit RFC library)	%Windows%\System32	%Windows%\SysWOW64
64-bit application (use 64-bit RFC library)	Not supported	%Windows%\System32

Table 1: Standard locations of RFC library files

E.g. it means that if you plan running your 32-bit application on 64-bit Windows then the RFC library files should be 32-bit and be located in the %Windows%\SysWOW64 folder.



Hint: If you work in 64-bit Windows and use a 32-bit file manager to copy files, then you should know a specific of 32-bit mode emulation on 64-bit Windows (WOW64). The %Windows%\System32 folder inside the application actually is an alias for the %Windows%\SysWOW64 in the host 64-bit system.

There are cases then the application using Connect for SAP® is abnormally terminated without any errors when the application tries to initialize the RFC library. In such a case it makes sense to double check whether all the required files are present at the right location.

4.2.3 Full Installation

To install either non-Unicode or Unicode version of the RFC library (or both), you need to download the software from the SAP software download center (<http://service.sap.com/swdc>). For that you should have an account at SAP.

To install the non-Unicode version:

- Search for “SAP RFC SDK”, select the latest SAP RFC SDK version, Unicode, and required platform (32-bit or 64-bit). For example, “SAP RFC SDK, 7.20”, then “Windows server on IA32 32bit”. And download it.
- Execute from the command prompt
SAPCAR -xvf <downloaded SAR file path>
- Copy the required files to the folder according to **Table 1**:
 - bin\librfc32.dll

To install the Unicode version:

- Search there for “SAP Classic RFC SDK”, select the latest SAP Classic RFC SDK version, Unicode, 32-bit option. For example, “SAP RFC SDK UNICODE, 7.20”, then “Windows server on IA32 32bit”. And download it.
- From the command prompt execute SAPCAR -xvf <downloaded SAR file path>
- Copy the set of required files to the folder according to **Table 1**:
 - bin\librfc32u.dll
 - lib\libsapu<versionVC>.dll
 - lib\libsapucum.dll
 - lib\icudt<version>.dll
 - lib\icuin<version>.dll
 - lib\icuuc<version>.dll

where <version> and <versionVC> are variable parts, which can differ depending on the version of RFC library. E.g. the files have the next names icudt30.dll, icuin30.dll, icuuc30.dll, libsapu16vc80.dll.

Read SAP Note 684106 and optionally:

- download and setup R3DLLINS.exe for SAP release 6.40 and 7.20
- download and setup vcredist_<platform>.exe (32-bit) for SAP release 4.6D EX2, 6.40 EX2, 7.20 from <http://www.microsoft.com/downloads/details.aspx?FamilyID=200B2FD9-AE1A-4A14-984D-389C36F85647&displaylang=en>

4.3 Additional Requirements

SAP user accounts used by a Connect for SAP® client application should have all required privileges to execute the following RFC functions:

Function	Purpose	Used by
RFC_GET_FUNCTION_INTERFACE	To dynamically obtain the function interface from non-Unicode servers.	TSAPxRFCFunctionGS, TSAPxRFCvFunctionGS
RFC_GET_FUNCTION_INTERFACE_US	To dynamically obtain the function interface from Unicode servers.	TSAPxRFCFunctionGS, TSAPxRFCvFunctionGS
RFC_GET_STRUCTURE_DEFINITION	To dynamically obtain the record data type layout from non-Unicode servers.	TSAPxRFCFunctionGS, TSAPxRFCvFunctionGS, TSAPxRFCTableGS, TSAPxRFCvTableGS
RFC_GET_UNICODE_STRUCTURE	To dynamically obtain the record data type layout from Unicode servers.	TSAPxRFCFunctionGS, TSAPxRFCvFunctionGS, TSAPxRFCTableGS, TSAPxRFCvTableGS
RFC_SYSTEM_INFO	To get the server representation of integer and float data types.	TSAPxRFCClientConnectionGS, TSAPxRFCvClientConnectionGS
RFC_READ_TABLE	To read SAP server tables data.	TSAPxRFCEasyDataMoveGS, TSAPxRFCvServerTableGS
RFC_FUNCTION_SEARCH	To show a list of accessible RFC functions	Connect for SAP® Explorer
RFC_GROUP_SEARCH	To show a list of accessible RFC function groups.	Connect for SAP® Explorer

4.4 Installing into Embarcadero Delphi or C++ Builder

4.4.1 Building Connect for SAP® Binaries

The Connect for SAP® software includes a set of BAT command files. They may be used to build Connect for SAP® binary files from the command line without running IDE.

The naming of the BAT files in dependence of IDE version is shown in the following table:

File name	Delphi / C++ Builder version
compileD<Delphi>.bat	Delphi: 5, 6, 7
compileBCB6.bat	C++ Builder 6
compileD<RADstudio>.bat	RADstudio: Delphi / C++ Builder 2005, 2006, 2007, 2009, 2010, XE...11

All the BAT files will automatically detect a location where the tool is installed to. If the tool is not installed you will get an error message about that.

The compiled binary files will be put into the <SAPx>\Lib\<Tool> folder. For example, the Delphi 2010 files will be put into <SAPx>\Lib\Delphi14.

4.4.2 Installing Components

Installation of new components has become very easy due to the Delphi package system. To install Connect for SAP®:

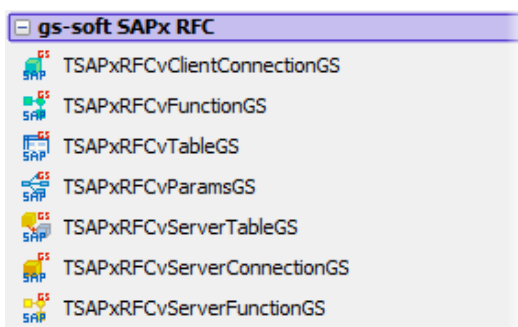
Run Delphi IDE.

Choose File -> Open. Set Files of type to Delphi package (*.dpk) and open the appropriate Package Project files in the Connect for SAP® installation directory. Naming of the runtime and the design time packages are correspondently gsSAPxRFC<Suffix>.dpk and gsSAPxRFC<Suffix>D.dpk. Values of the *Suffix* parameters are specified in following table:

Suffix	Delphi / C++ Builder version
d5...d7	Delphi 5...7
d9...d11	Delphi / C++ Builder 2005...2007
d12	Delphi / C++ Builder 2009
d14	Delphi / C++ Builder 2010
d15...d28	Delphi / C++ Builder XE...11

Click on the Compile button in the "Package" window. To install the design time package after compilation press the Install button.

The new components should appear in the Delphi Component Palette as shown on the next figure.



You can find the list of all the Connect for SAP® components in **Appendix C**.

5 Connect for SAP® Explorer

Connect for SAP® Explorer is a tool giving an access to any SAP R/3 function supporting RFC (Remote Function Call) from the outside of a SAP system. The Explorer carries out the following functions

- Creating and maintaining the Connect for SAP® connection aliases
- Getting a metadata for the SAP RFC objects (functions, structures, tables)
- Executing RFC functions
- Generating Delphi wrapping source code for using RFC functions

5.1 Creating and Maintaining Aliases

A connection alias is a named persistent set of client connection parameters. Connect for SAP® applications may use the aliases to connect to SAP R/3 systems. The Explorer allows creating, viewing and modifying the aliases.

Aliases are stored in an INI file. The Explorer allows defining a default alias file which used by Connect for SAP® applications in cases when no alias file is explicitly specified at the application level. The full path to default alias file is stored under the registry key `HKCU\Software\gs-soft\SAPxRFC\CfgFile`.

In Connect for SAP® applications, a connection to a SAP server can be configured either at runtime or design time stages. For each of these stages, there are 2 approaches how to define connection parameters:

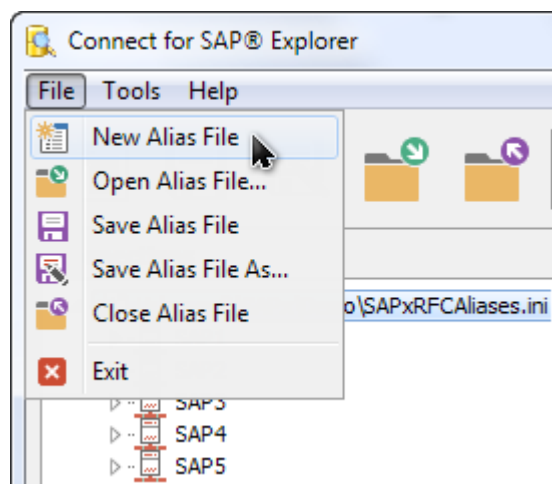
- Specify an alias providing the required connection parameters.
- Explicitly setting values of connection parameters.

The next rules show priorities of combined usage of these approaches:

1. Parameters defined at runtime have a higher priority than ones defined at design time. It means if both of these kinds of parameters are specified – runtime parameters override design time ones.
2. If the connection parameters are explicitly specified – they override the corresponding parameters from an alias (if one is defined).
3. If an alias file is explicitly specified – alias name defines an alias in the file. Otherwise, the default alias file is used.
4. If the default alias file is not defined - Connect for SAP® raises an error.

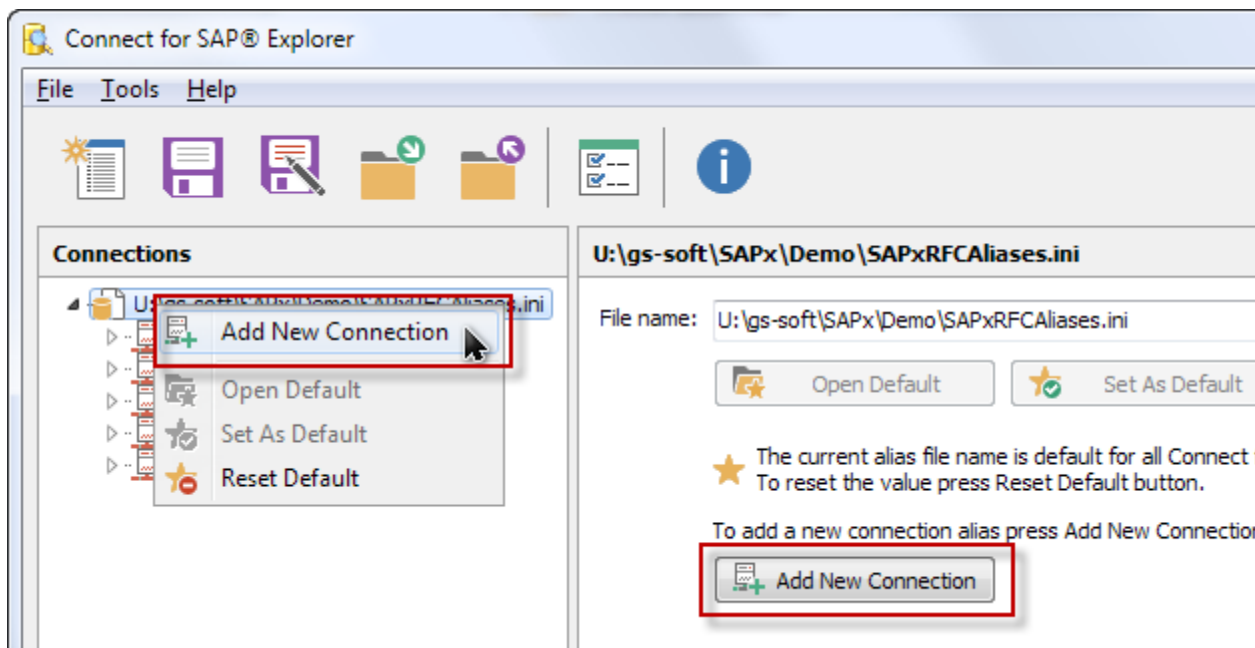
5.1.1 Creating or Opening a Connection Alias File

To start working with an alias file you should use File menu items New Alias File or Open Alias File... and create a new or open already existing alias file. If default alias file has been specified in your system the Explorer opens the file on starting.



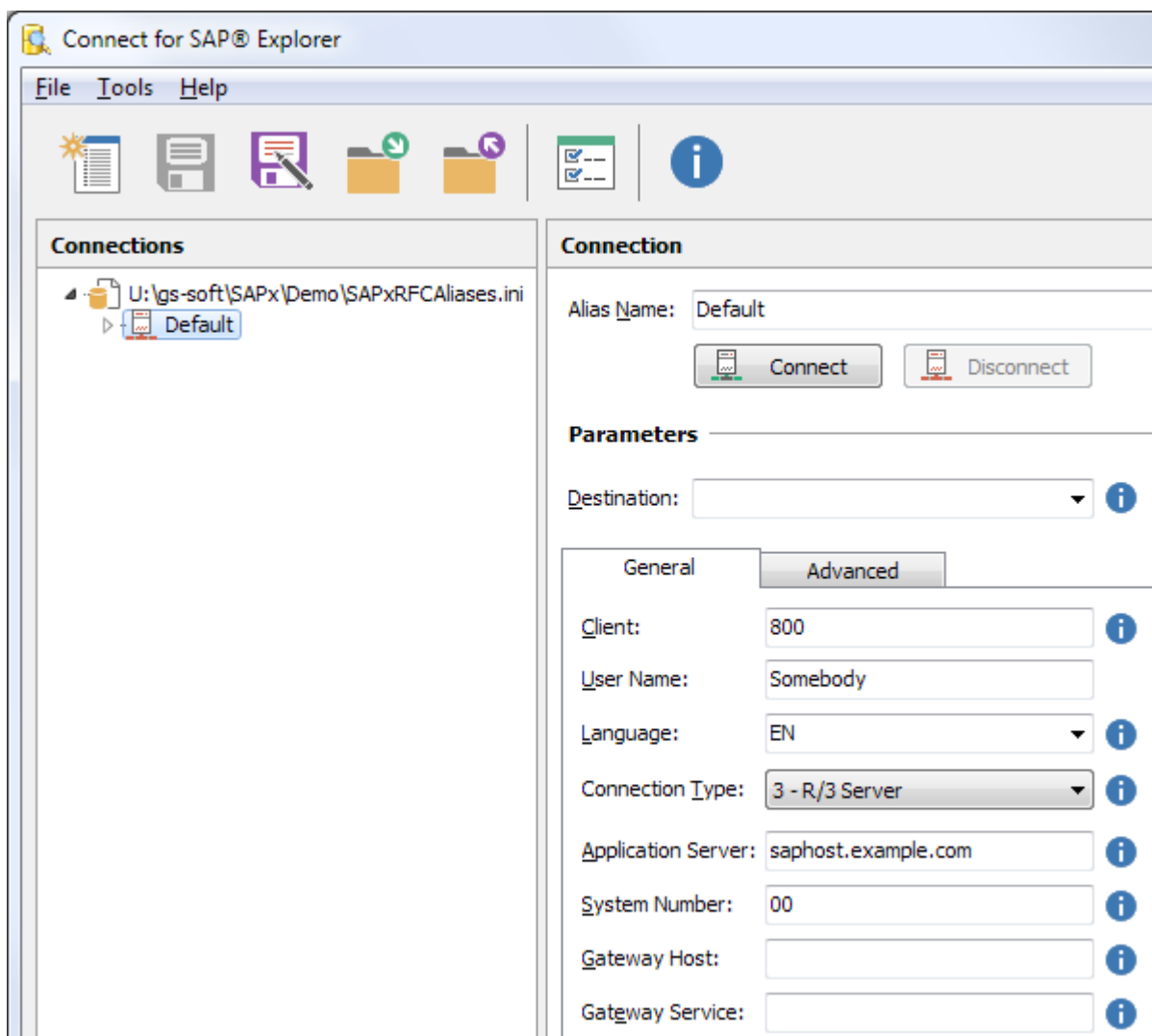
5.1.2 Creating a New Connection Alias

To create a new connection alias you may use either popup menu of the root item in Connections tree or click Add New Connection on the right page.



5.1.3 Modifying a Connection Alias

To edit parameters of an already existing alias you should select it in the Connections tree. On the right you can edit the connection parameters.



Usually to connect to a SAP R/3 server the following parameters are required:

- Client
- User Name
- Language
- Connection Type (normally it should be set to "R/3 Server")
- Application Server
- System Number

The rest of parameters are required in more complex cases and should be specified according to the SAP documentation (see [SAP RFC Parameter Overview](#)).

To define data representations of specific types by the RFC library you should use *Advanced >>Data Format*.

Field	Description
Integer As	Specifies an integer data type representation expected by the RFC library for the data sent between the server and the client
Float As	Specifies a float data type representation expected by the RFC library for the data sent between the server and the client
Byte Per Char	Specifies a number of bytes that RFC library expects the server uses for character based data types when communicating between the server and the client
Map BCD To	Specifies a Delphi data type to represent RFC_BCD data type in Delphi code

See more detail in the **Data Representation** topic.

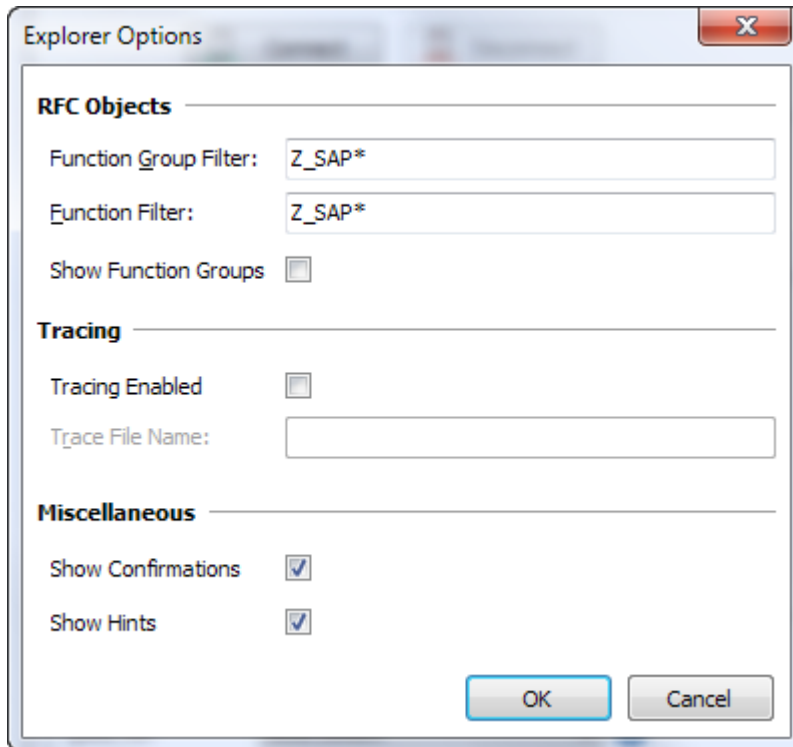
As soon as you finish editing the connection alias press *Save* and the modified alias can be used in any Connect for SAP® application installed on this computer.

5.2 Browsing SAP Dictionary Information of RFC Functions

SAP system dictionary contains a lot of metadata. Using the Connect for SAP® Explorer you can access to the definition of the RFC functions.

5.2.1 Displaying RFC Functions

Use the Connect for SAP® Explorer options to define a set of RFC functions to be displayed. Function Group Filter, Function Filter and Show Function Groups options controls that. You can modify them through the Explorer Options form by clicking the Tools >> Options menu item.



The screenshot shows the 'Explorer Options' dialog box. It is divided into three sections: 'RFC Objects', 'Tracing', and 'Miscellaneous'. In the 'RFC Objects' section, the 'Function Group Filter' and 'Function Filter' text boxes both contain the text 'Z_SAP*'. The 'Show Function Groups' checkbox is unchecked. In the 'Tracing' section, the 'Tracing Enabled' checkbox is unchecked, and the 'Trace File Name' text box is empty. In the 'Miscellaneous' section, both the 'Show Confirmations' and 'Show Hints' checkboxes are checked. At the bottom of the dialog, there are 'OK' and 'Cancel' buttons.

The two first options define what function groups and functions should be shown finally. The last option indicates if function modules will be grouped by groups they belong to.

The SAP dictionary contains too much information. To avoid downloading useless data it is recommended to use the Function Group Filter and Function Filter options. These options will help you to reduce the execution time considerably, especially in slow networks.

5.2.2 RFC Objects Definition Information

To get the RFC objects information you need

1. Specify the Explorer's options.
2. Choose a connection alias to connect to a SAP system.
3. Open the alias by clicking the Connect button (or popup menu item of Connection tree).

Connection

Alias Name: Default

Connect **Disconnect**

Parameters

Destination: ⓘ

General **Advanced**

Client: ⓘ

User Name:

Language: ⓘ

Connection Type: ⓘ

4. Specify user credentials on the “SAP R/3 Login” form.
5. Select the required Function group and Function in the tree.

Connect for SAP® Explorer

File Tools Help

Connections

U:\gs-soft\SAPx\Demo\SAPxRFCAliases.ini

- SAP1
 - RFC Function List
 - Z_ACSIS_GETCAUFV
 - Z_ACSIS_GETMAKT
 - Z_COPY_DATE
 - Z_FSCM_INPUT_PDF
 - Z_SAFYR_RFCSDDEXPV6
 - Z_SAPXRFC_INFO**
 - Z_SAPX_CALL_EXT_SEXTXTABLE
 - Z_SAPX_CALL_SLEEP
 - Z_SAPX_DU_CLIENT_BCD
 - Z_SAPX_DU_CLIENT_TRANSITION
 - Z_SAPX_DU_SERVER_TRANSITION
 - Z_SAPX_GETCOUNTRYDATA
 - Z_SAPX_TESTCREATION
 - Z_SAPX_TEST_INTERNAL_TAB
 - Z_SAPX_TEST_INT_ALIGNMENT
 - Z_SAPX_TEST_LONGSTRING

RFC Function

Name: Z_SAPXRFC_INFO

Description: test

Group: Z_SAPX

Host:

Application:

Execute... **Generate...**

Interface

Parameters

	Class	Name	Type	Table	Field	Position
1	Import	ALOOPNUM	I	ZSAPXRFC_STFC	I10	3
2	Export	AINFO	C	ZSAPXRFC_STFC	TABNAME	1

Definition of the RFC function consists of a description of its parameters, tables and exceptions. On the print screen above you can see the RFC function information.

5.3 Executing RFC Functions

The Connect for SAP® Explorer can execute any RFC function from outside of a SAP system. To perform this task you need

1. Select an RFC function in the tree.
2. Press Execute... button on the right panel to show the Execute RFC Function form.
3. Specify the necessary Import Parameters and Tables fields.
4. Press Execute button to run the function.
5. Check the Export parameters and the Table fields as soon as the function is executed.

5.4 Generating Wrapping Code for RFC Function

There are two main goals in generation of a wrapping code for an RFC function:

- A generation of the classes encapsulating the set of parameters and tables defined in functions. This turns the SAP RFC functions into natural extension of Object Pascal language. Usage of the statically defined objects allows avoiding round trip to SAP system for getting metadata during object preparing and to increase a processing speed.
- The Connect for SAP® users may use the IDE Code Insight feature. That makes a coding more efficient and helps to avoid potential coding mistakes.

The main drawback of generated code – it is sensitive to changes in Function Module on the SAP server. Check **How to Use Generated Code** for details. If the restrictions are not acceptable for you then use the dynamic function execution.

5.4.1 Generating Wrapping Code

To generate an Object Pascal code you should:

1. Select an RFC function in the tree.
2. Press Generate... button on the right panel to show the Generate Source Code form.
3. Specify Output Directory where result generated unit will be located.
4. Check the rest of parameters and correct them if it's required.
5. Generate the Object Pascal unit by clicking on the Generate button.

The screenshot shows a dialog box titled "Generate Source Code - Z_SAPX_DU_CLIENT_TRANSITION". It contains several sections for configuring the code generation process:

- Output Directory:** C:\Temp\SAPx
- Name Templates:**
 - Function: TSAPxRFC%sFunctionGS
 - Table: TSAPxRFC%sTableGS
 - Structure: TSAPxRFC%sStructureGS
- Base Classes:**
 - Function: TSAPxRFCFunctionGS
 - Table: TSAPxRFCTableGS
 - Structure: TSAPxRFCParameterGS
- Structured Data Types:** A table with columns: Object, Type, Class Name, Base Class Name, and Unit.

Object	Type	Class Name	Base Class Name	Unit
T000	Str	TSAPxRFCT000StructureGS	TSAPxRFCParameterGS	gs
T000	Tab	TSAPxRFCT000TableGS	TSAPxRFCTableGS	gs
T042	Tab	TSAPxRFCT042TableGS	TSAPxRFCTableGS	gs
Z_SAPX_DU_CLIENT_TRANSITION	Fnc	TSAPxRFCZ_SAPX_DU_CLIENT_	TSAPxRFCFunctionGS	gs

At the bottom, there are three buttons: "Refresh Structured Data Types", "Generate" (highlighted in blue), and "Cancel".

Now you can use the generated classes equally to other classes. Complex parameter types, such as SAP structures, are converted to the corresponding Connect for SAP® classes. This feature simplifies their usage, allowing you to take an advantage of the IDE Code Insight feature and be sure that your code can be run if it compiles.

5.4.2 How to Use Generated Code

Generated wrapping code may be used only at runtime, because the generated classes are not inherited from the TComponent class.

Please note that after any changes in an ABAP Function Module its wrapping code must be regenerated.

To use a generated code for a SAP function, you should:

- Include the generated units into your code by uses clause;
- Create an object of the SAP function generated class;
- Set the Connection property to the RFC connection object. If you use components, then assign TSAPxRFCvClientConnectionGS.RfcConnection.

6 Troubleshooting

6.1 Issue report

If you encounter any issue when working with the Connect for SAP® library then you may raise an issue report for the technical support. To make the process of gathering necessary information easier you may use a document template for such a report. It's located in the file `<SAPx>\Docu\Connect for SAP - Issue Report.docx`.

6.2 Tracing

To debug a problem in an application using the Connect for SAP® library two tracing systems can be used. One of them is SAP RFC library tracing system and another one is own Connect for SAP® tracer.

6.2.1 SAP RFC library tracing

This system allows getting information about internals of RFC library calls. To handle SAP tracing you need to the RFC_TRACE flag within the client connection parameters to the value "True" (see `TSAPxRFCClientConnectionGS.Params`). The resulting trace files are located in the working directory of your application and have names `dev_rfc.trc` and `rfcxxxxx_xxxx.trc`.

6.2.2 "Connect for SAP"® tracing

This tracing system allows logging detailed information about internal work within "Connect for SAP" library. To turn on the tracing in your application the `gsSAPxRFCStdObj.SAPxRFCSetTracing` call is used. If you do not specify exact file name for the trace file name it will have a name like `SAPxRFC<xxxxxxx>` and be located in the working directory of your application.



Important Note: To have a possibility to activate the tracing system in the application it has to be built with the **SAPX_TRACE compiler directive**. This directive is by default enabled (see `gsSAPx.inc`).

We also recommend developers to have a possibility to enable/disable tracing in their application. This will allow activating the tracing on the customer site without rebuilding and reinstalling the software.

Appendix A – Defining Server Parameters

The following table shows the command line switches and their meanings:

Command line switch	Description
-a<PROGRAM_ID>	Identifier of server connection registered on SAP gateway
-g<GWHOST>	Host name of SAP gateway
-x<GWSERV>	Service name on SAP gateway
-t<RFC_TRACE>	Indicator of tracing
-D<DESTINATION>	Destination name in saprfc.ini file

Listing 3: Example of a destination list in saprfc.ini file

```
// -----  
// SAPx server destinations  
// -----  
DEST=S1  
TYPE=R  
PROGID=SAPXRFCTEST_PROGID1  
GWHOST=myserver.mydomain.com  
GWSERV=sapgw00
```

Appendix B – Transaction Management in Connect for SAP® Server Application

Connect for SAP® supports transactional server functions. SAP R/3, the RFC library and Connect for SAP® server connection communicate in two phases (see **Figure 4: Scheme of calling a transactional function**):

- The first phase (**F1**) – Function transfer
- The second phase (**F2**) – Confirmation

Function transfer phase is initiated in ABAP program and is divided into three parts:

- **T1** – OnCheckTID event handler has to check the TID status, update it and return the corresponding check result.
- **T2** – OnExecute event handler should contain the required RFC server function implementation.
- **T3 (T3')** – OnCommit (OnRollback) event handler updates the TID status and commits (rolls back) database (non-SAP database) transaction(s).

Confirmation phase starts as soon as the RFC library informs the SAP system about successful **T3** (not **T3'**). The TSAPxRFCvServerConnectionGS component receives a confirmation of the current transaction. In the OnConfirm event handler the developer should update the TID status (delete). After this phase is over the current transaction is successfully completed on both sides.

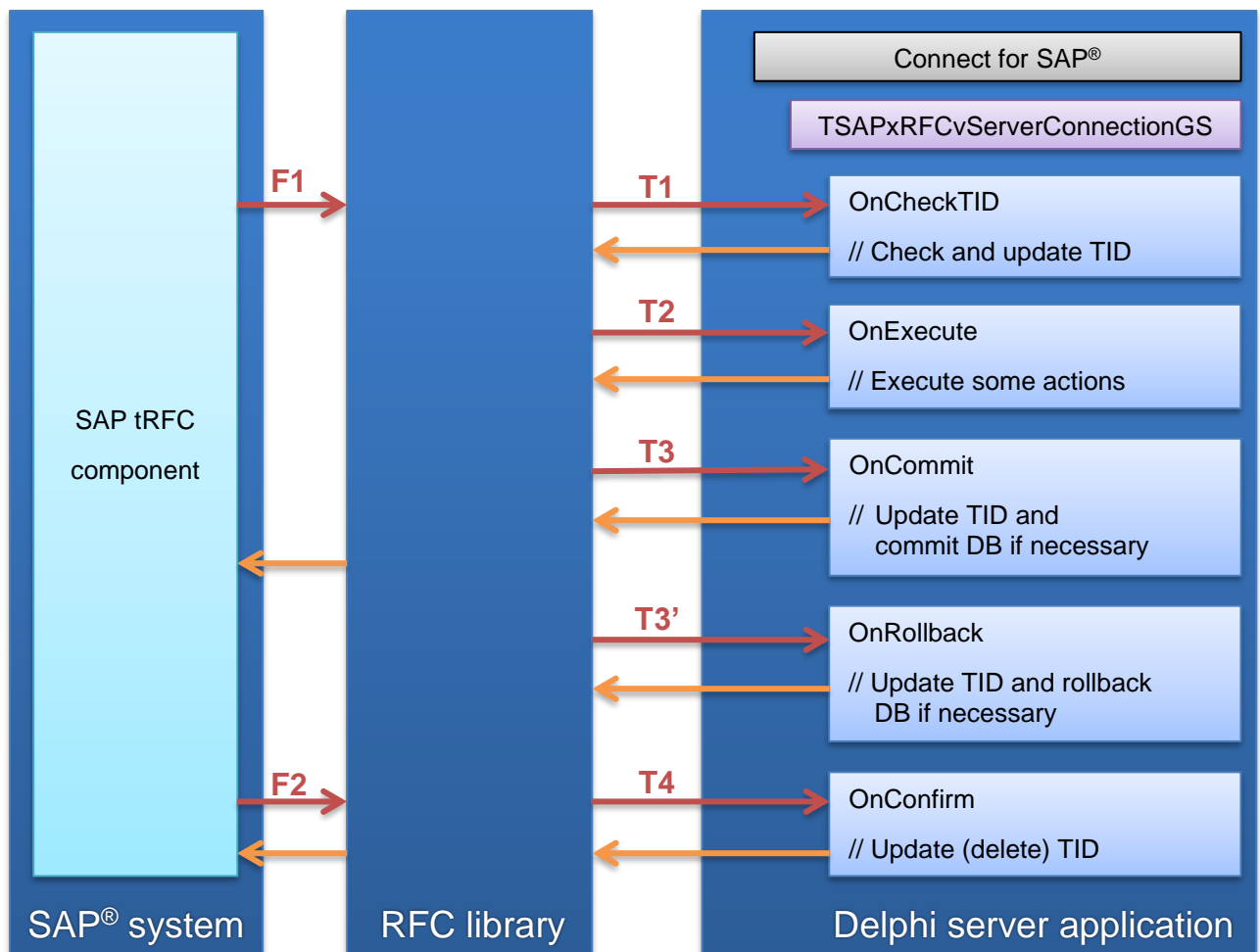


Figure 4: Scheme of calling a transactional function

Appendix C – Connect for SAP® Component List

The Connect for SAP® components are divided into two groups: components for client programs and for non-SAP server programs. In the following sections you can find a description of each of these groups.

Client components



TSAPxRFCvClientConnectionGS

The main client component. It connects to the specified SAP system and supports the data exchange between a client program and the SAP system.



TSAPxRFCvTableGS

A descendant of TDataSet component. Can be used by data aware controls. It allows the access to the specified table from the TSAPxRFCvFunctionGS table list.



TSAPxRFCvParamsGS

Corresponds to a set of function parameters. It allows editing and displaying a set of parameters using data aware controls.



TSAPxRFCvFunctionGS

Executes an ABAP RFC function. It contains sets of input and output parameters and table lists that are used for access to the function data.



TSAPxRFCvServerTableGS

A descendant of the TDataSet component. Can be used by data aware controls. It allows getting dictionary information on a specified SAP DB table (fields description) and data stored within this table.

Server components



TSAPxRFCvServerFunctionGS

Implements a certain part of the server functionality. Every such a server function component belongs to the specified server connection and can receive client requests only from it.



TSAPxRFCvServerConnectionGS

The main component for non-SAP server programs. It registers all supported server functions on a SAP gateway, processes client requests and dispatches them.