

# Connect for SAP® (NetWeaver)



## Getting Started

<b>1</b>	<b>About this Document</b>	<b>3</b>
<b>2</b>	<b>About Connect for SAP® (NetWeaver)</b>	<b>4</b>
<b>3</b>	<b>Architectural overview</b>	<b>5</b>
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	9
3.2	Client Applications	11
3.2.1	Using Connection Aliases	12
3.2.2	Features for Transactional Calls	12
3.3	Server Applications	13
3.3.1	Specifying Registration Parameters	14
3.3.2	Features for Transactional Calls	14
<b>4</b>	<b>Installation</b>	<b>15</b>
4.1	System Requirements	15
4.2	Installing RFC Library Using SAPGUI	15
4.3	Manual Installation of RFC Library Binaries	17
4.4	Requirements for SAP Server	19
4.5	Installing into Embarcadero Delphi or C++ Builder	20
4.5.1	Building Connect for SAP® Binaries	20
4.5.2	Installing Components	20
<b>5</b>	<b>Connectivity to SAP</b>	<b>21</b>
5.1	Client Connection	21
5.1.1	Connection Alias	21
5.1.2	Connection Parameters Priority	21
5.2	Server Connection	21
<b>6</b>	<b>Connect for SAP® Explorer</b>	<b>22</b>
6.1	Creating and Maintaining Connection Aliases	22
6.1.1	Creating or Opening a Connection Alias File	22
6.1.2	Creating a New Connection Alias	22
6.1.3	Modifying a Connection Alias	23
6.2	Browsing SAP Dictionary Information of RFC Functions	25
6.2.1	Displaying RFC Functions	25
6.2.2	RFC Objects Definition Information	26
6.3	Executing RFC Functions	27
6.4	Generating Wrapping Code for RFC Function	27
6.4.1	Generating Wrapping Code	28
6.4.2	How to Use Generated Code	29
<b>7</b>	<b>Migrate Code to New Connect for SAP®</b>	<b>30</b>
<b>8</b>	<b>Troubleshooting</b>	<b>32</b>
8.1	Issue Report	32
8.2	Tracing	32
8.2.1	SAP RFC Library Tracing	32
8.2.2	“Connect for SAP”® Tracing	32
<b>Appendix A.</b>	<b>Using sapnrwfc.ini</b>	<b>33</b>
<b>Appendix B.</b>	<b>Transaction Management in Connect for SAP® Server Application</b>	<b>34</b>
<b>Appendix C.</b>	<b>Connect for SAP® Component List</b>	<b>35</b>
<b>Appendix D.</b>	<b>Changes of Connect for SAP® API between 3.x and 4.x</b>	<b>36</b>
<b>Appendix E.</b>	<b>Changes of Applications to be Applied Manually</b>	<b>41</b>

## 1 About this Document

This document might be especially 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: <https://www.gs-soft.com/CMS/en/products/connect-for-sap-sapx>

By email: [sapx@gs-soft.com](mailto:sapx@gs-soft.com)

## 2 About Connect for SAP® (NetWeaver)

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 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 several 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 an RFC SDK that includes an 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 you need to call an ABAP function you should 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.

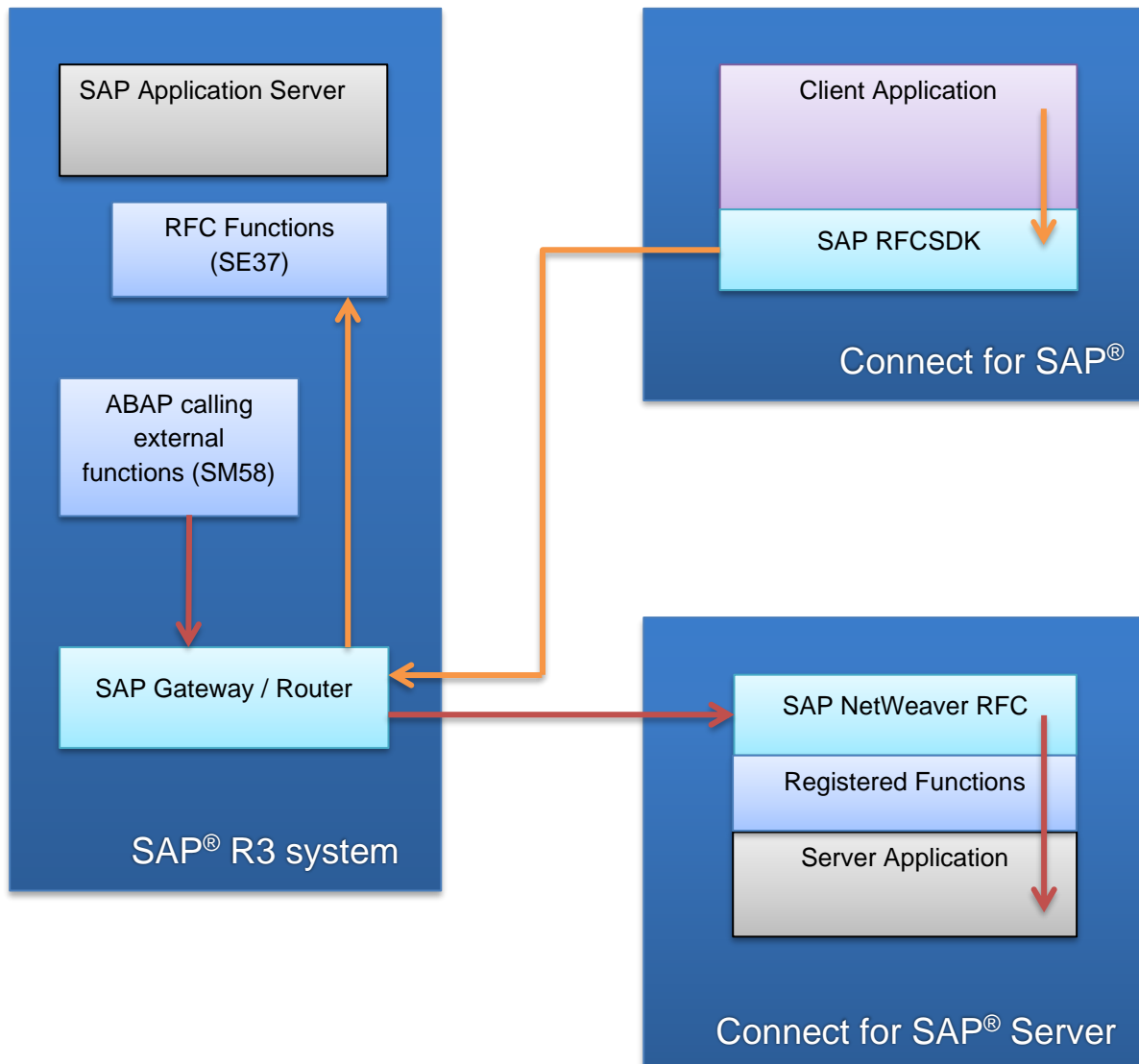
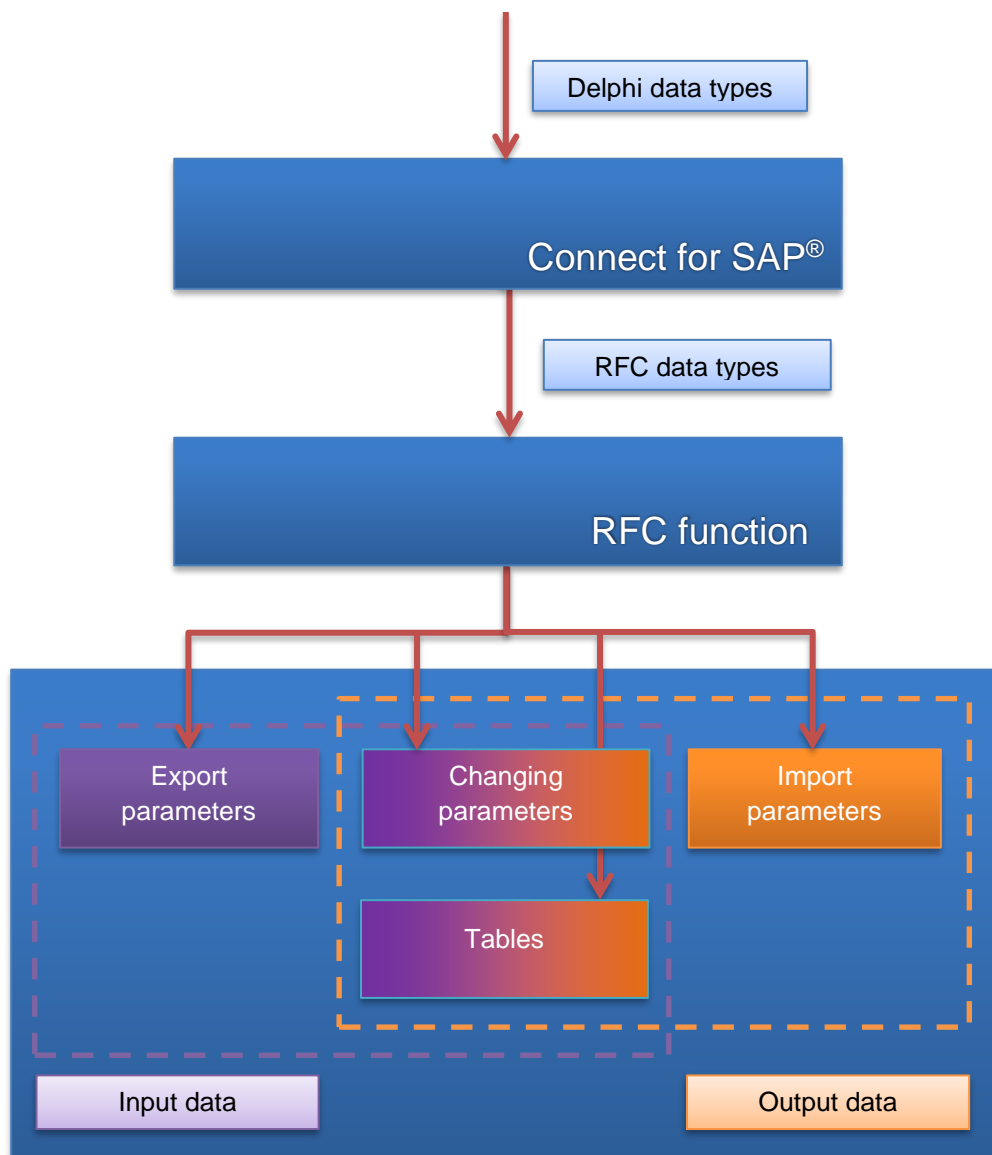


Figure 1: Interaction of a SAP system with Delphi application based on Connect for SAP®

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 an 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 and Changing parameters can contain both input and output data. Data imported from and exported to RFC function has data formats and types which 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 can 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 `TSAPxNWAliasGS.DataFormat.IntType` and `TSAPxNWAliasGS.DataFormat.FloatType`. By default, they have the values `itAutoDetect` and `ftAutoDetect`.

The alias parameter `TSAPxNWAliasGS.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 indicate the target Delphi data type. Otherwise, Connect for SAP® maps this data type to the most appropriate Delphi data type.

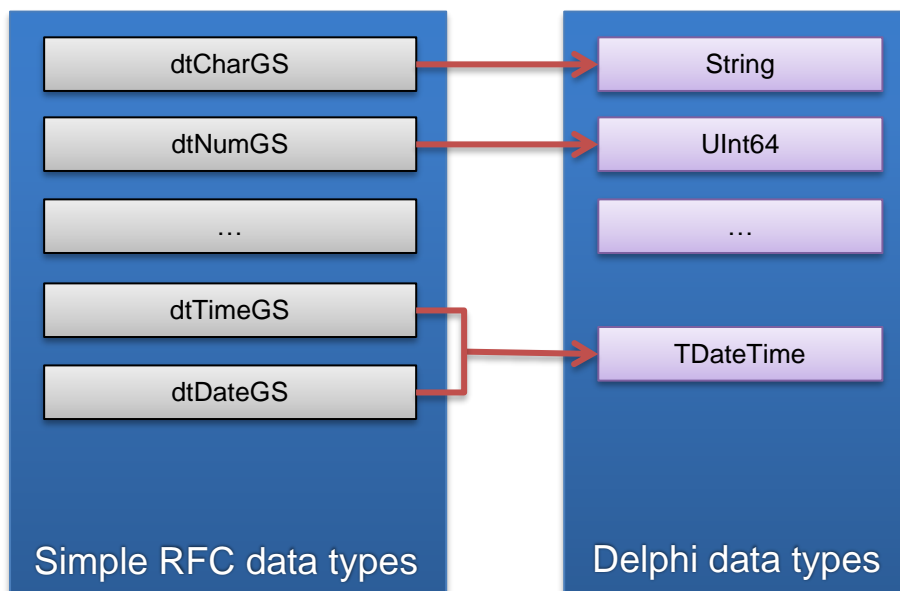


Figure 2: Simple RFC data type mapping

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

RFC code	RFCTYPE	RFC data type	Delphi data type	Delphi field type
C	RFCTYPE_CHAR	dtCharGS	UnicodeString	ftWideString
N	RFCTYPE_NUM	dtNumGS	UInt64	ftSmallInt (size <= 4)
				ftInteger (size <= 9)
				ftLargeInt (size > 9)
X	RFCTYPE_BYTE	dtByteGS	RawByteString	ftVarBytes
P	RFCTYPE_BCD	dtBCDGS	TBcd	ftFloat
I	RFCTYPE_INT	dtIntGS	Integer	ftInteger
b	RFCTYPE_INT2	dtInt1GS	ShortInt	ftSmallInt
s	RFCTYPE_INT1	dtInt2GS	SmallInt	ftSmallInt
F	RFCTYPE_FLOAT	dtFloatGS	Double	ftFloat
D	RFCTYPE_DATE	dtDateGS	TDateTime	ftDate
T	RFCTYPE_TIME	dtTimeGS	TDateTime	ftTime
g	RFCTYPE_STRING	dtStringGS	UnicodeString	ftWideString
y	RFCTYPE_XSTRING	dtXStringGS	RawByteString	ftVarBytes

### 3.1.2.2 Structured Data Types

Unlike a simple data type, structured one, does not have Delphi analogues. **Figure 3** illustrates how the Connect for SAP® wraps the dtStructureGS type by means of the TSAPxNWParameterGS class, which contains a field list of the TSAPxNWFieldCollectionGS class. So, the structure corresponds to the field list, where an individual field of the TSAPxNWFieldGS 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.

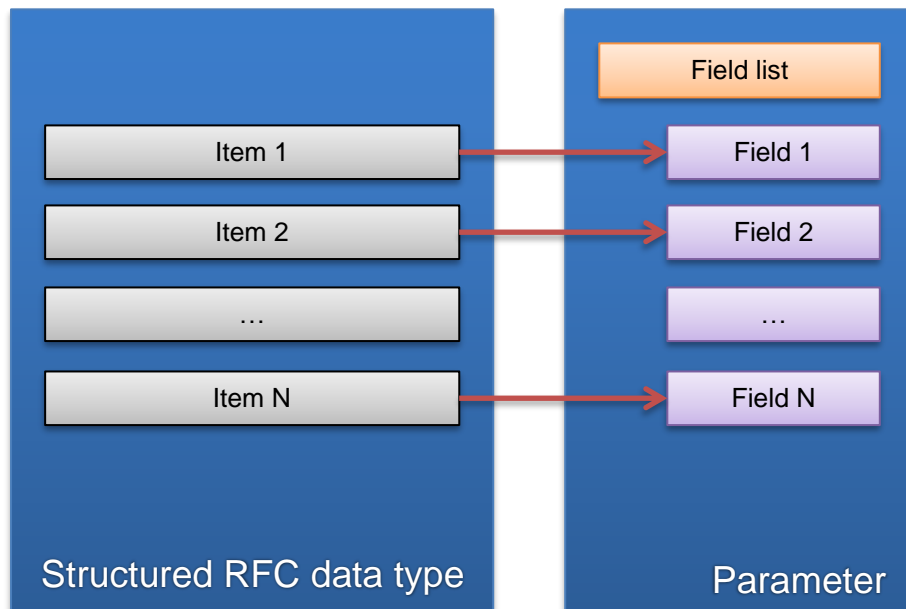


Figure 3: Wrapping structured RFC data type by Connect for SAP®

Connect for SAP® includes the TSAPxNWClientCompParametersGS 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.

TSAPxNWClientCompParametersGS 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, changing) TSAPxNWClientCompParametersGS includes by specifying ParameterTypes.

The next table shows all structured types and how to work with them in Delphi code:

RFC code	RFCTYPE	RFC data type	Delphi data type
u	RFCTYPE_STRUCTURE	dtStructureGS	
h	RFCTYPE_TABLE	dtLineTypeGS	Not supported



**Note:** Connect for SAP® supports only the flat structured type **RFCTYPE\_STRUCTURE**. The data types that have a non-flat structure (deep structured tables, Line type) are **not supported**. An error occurs when such a parameter is encountered.



### 3.1.2.3 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 `TSAPxNWClientCompTableGS` 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. `TSAPxNWClientCompTableGS` 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 component
RFC table parameter	<code>TSAPxNWTableGS</code>	<code>TSAPxNWClientCompTableGS</code>

### 3.1.2.4 Unicode Character Data

Connect for SAP® supports Unicode. This means, that for the `CHAR` data type and for the object names, the library always uses the Unicode UCS-2 encoded character data.

### 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.**

```
TSAPxNWRFC_READ_TABLEFuncGS = class(TSAPxNWClientFunctionGS)
private
  procedure SetDELIMITER(const AValue: String);
  function GetDELIMITER: String;
  function GetDATA: TSAPxNWTAB512TableGS;
public
  constructor Create; override;
  property DELIMITER: String read GetDELIMITER write SetDELIMITER;
  property DATA: TSAPxNWTAB512TableGS 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 TSAPxNWClientFunctionGS object interface
  with FCFunction do begin
    ObjName := 'RFC_READ_TABLE';
    Prepared := True;
    InParameters.ItemByName('DELIMITER').ValueAdapter.AsString := '%';
    ExecFunction;
    with Tables.ItemByName('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 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.

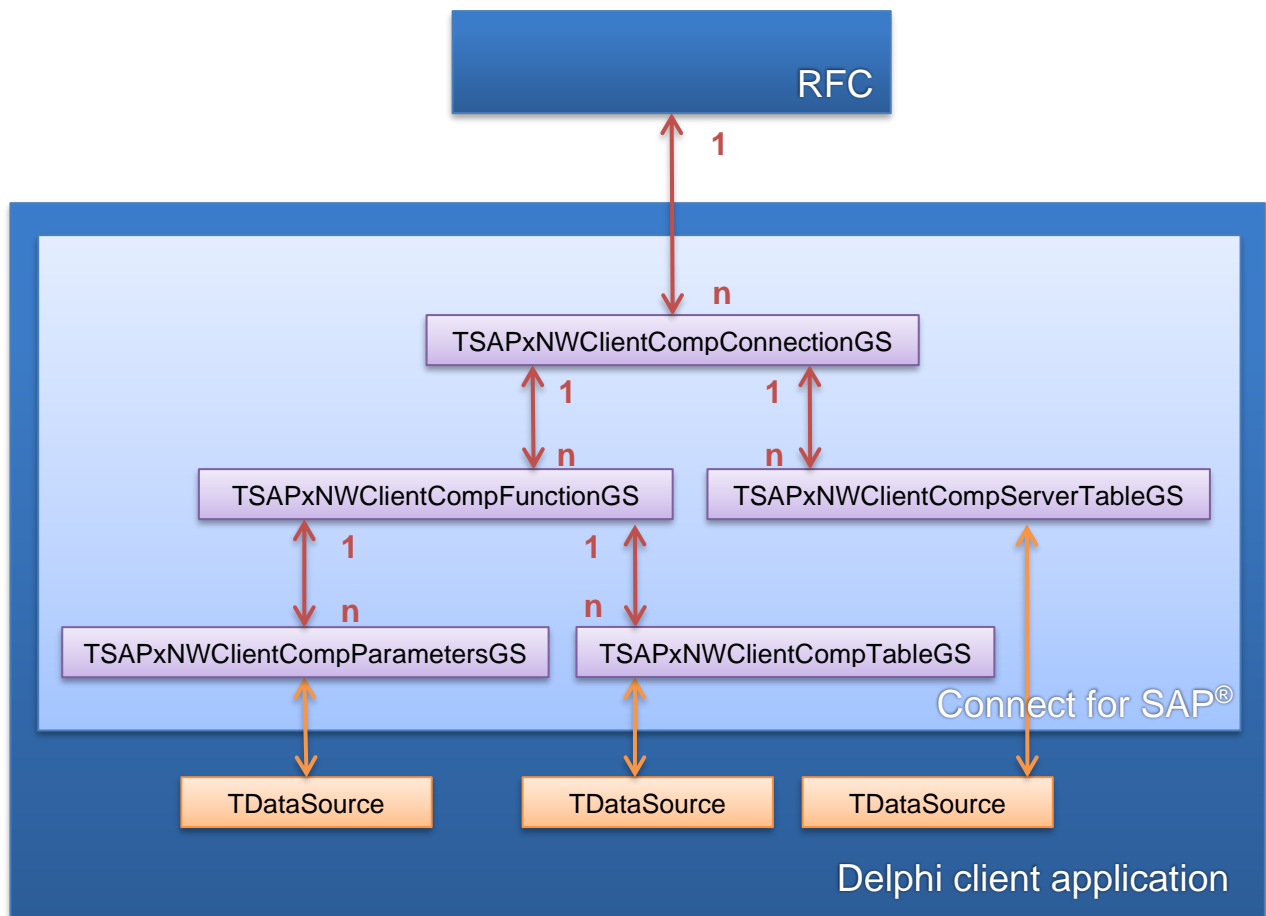


**Note:** The first preparation of a client function's call requires a lookup in back-end system's data dictionary (DDIC). In case the function module has many parameters or tables it may take up to several seconds. After that, further calls will use data cached by the RFC library locally. For additional details see description of NetWeaver API function [RfcGetFunctionDesc](#).

So, it is recommended an explicit preparation (TSAPxNWClientFunctionGS.Prepare) of the calling function before the first execution (for example, in a background thread).

### 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 `TSAPxNWClientCompConnectionGS` component is responsible for connection to 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 `TSAPxNWClientCompFunctionGS` and `TSAPxNWClientCompServerTableGS` use the `TSAPxNWClientCompConnectionGS` object to communicate with the RFC library.

The main Connect for SAP® client component is `TSAPxNWClientCompFunctionGS`. 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 `OutParameters` property represents an output parameter collection, the `InParameters` – collection of input parameters, the `VarParameters` – collection of parameters used for input and output, the `Tables` – collection of tables. These properties are not accessible at design time.

The developer can use the `TSAPxNWClientCompParametersGS` and `TSAPxNWClientCompTableGS` components to operate with parameters and tables of `TSAPxNWClientCompFunctionGS`. Set the `Func` property to a function component. Use `TSAPxNWClientCompParametersGS.ParameterTypes` to specify parameter types (input, output, changing) to represent. Use `TSAPxNWClientCompTableGS.TableName` to specify table parameter to represent.

The components `TSAPxNWClientCompParametersGS`, `TSAPxNWClientCompTableGS`, and `TSAPxNWClientCompServerTableGS` 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 `TSAPxNWClientCompFunctionGS`.

### 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%\SAPxNWAliases.ini` file.

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

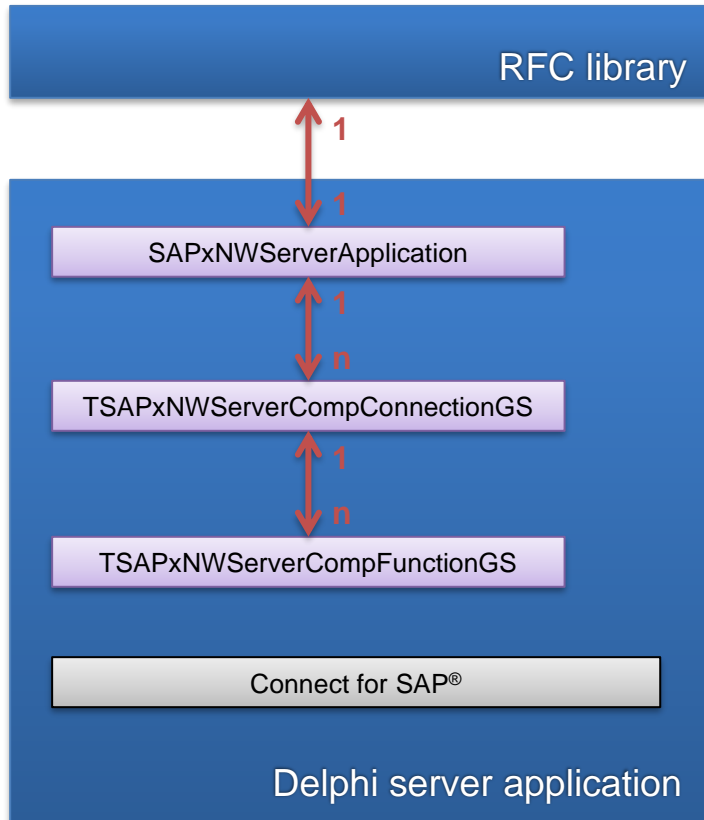
### 3.2.2 Features for Transactional Calls

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 TSAPxNWClientCompFunctionGS should be set to ftTransactionalGS.
- TSAPxNWClientCompFunctionGS 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 `TSAPxNWServerCompConnectionGS` component is responsible for a registration of a server on a gateway and a communication of the `TSAPxNWServerCompFunctionGS` 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 serialized and handled in a single thread.

The main Connect for SAP® server component is `TSAPxNWServerCompFunctionGS`. 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` – collection of input parameters, the `VarParameters` – collection of parameters used for input and output, the `Tables` – collection of tables. These properties are not accessible at design time.



**Note:** When you construct a table or a parameter of structure type for Server Function, it is recommended to explicitly define the property `StructName`. This allows obtaining of the required metadata from the SAP system the function will be connected to. In case the property has not been defined the metadata will be created locally based on defined fields. That is less safe due to possible mismatches between fields defined on SAP system and ones added to the function locally.

And the `SAPxNWServerApplication` is a singleton, controlling the Connect for SAP® server application life cycle. All the `TSAPxNWServerCompConnectionGS` and `TSAPxNWServerCompFunctionGS` objects must be created and setup before calling `SAPxNWServerApplication.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 `SAPxNWServerApplication.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 `TSAPxNWServerCompConnectionGS` 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 need 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 explicitly specify the `CommandLine` property of the `TSAPxNWServerCompConnectionGS` 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

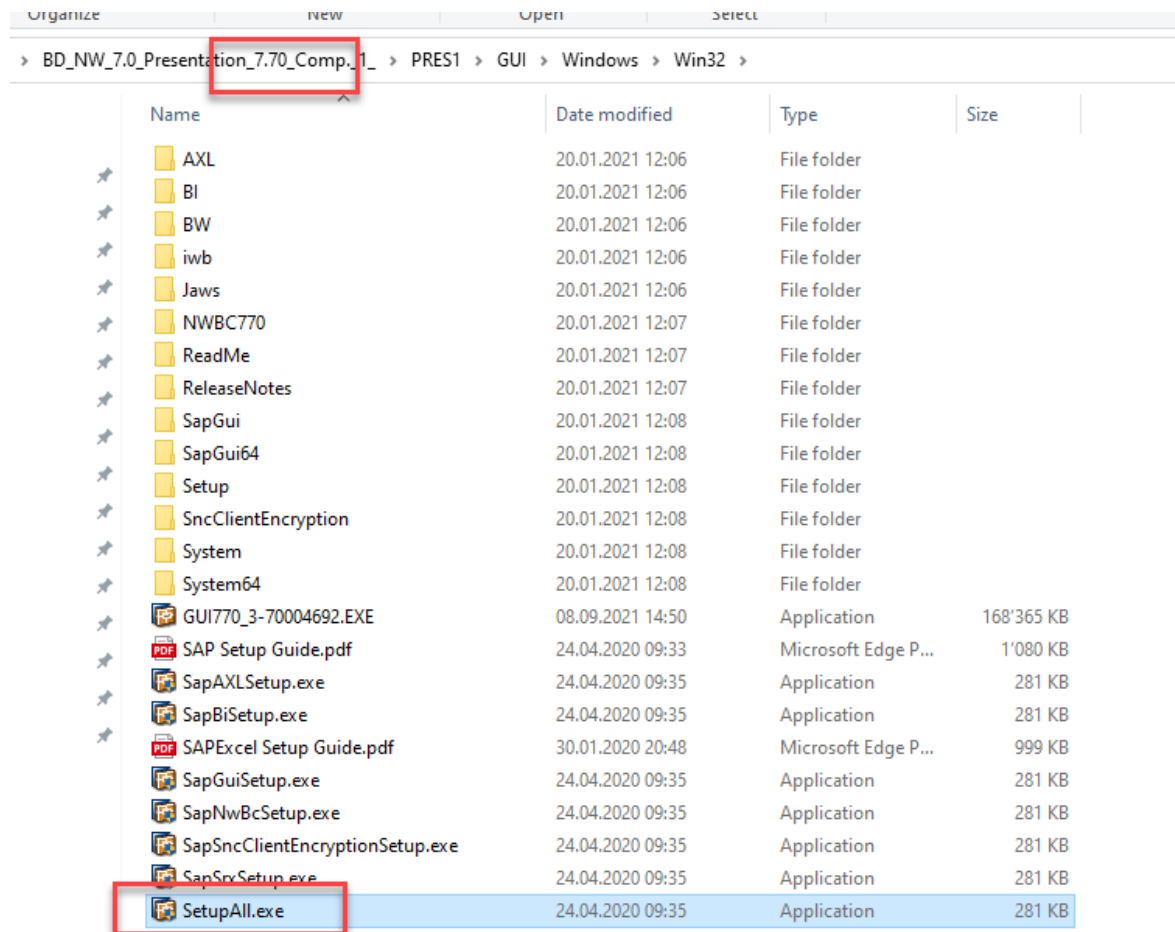
Before installing Connect for SAP® ensure that Embarcadero Delphi™ XE6 or higher is installed on your PC.

### 4.2 Installing RFC Library Using SAPGUI

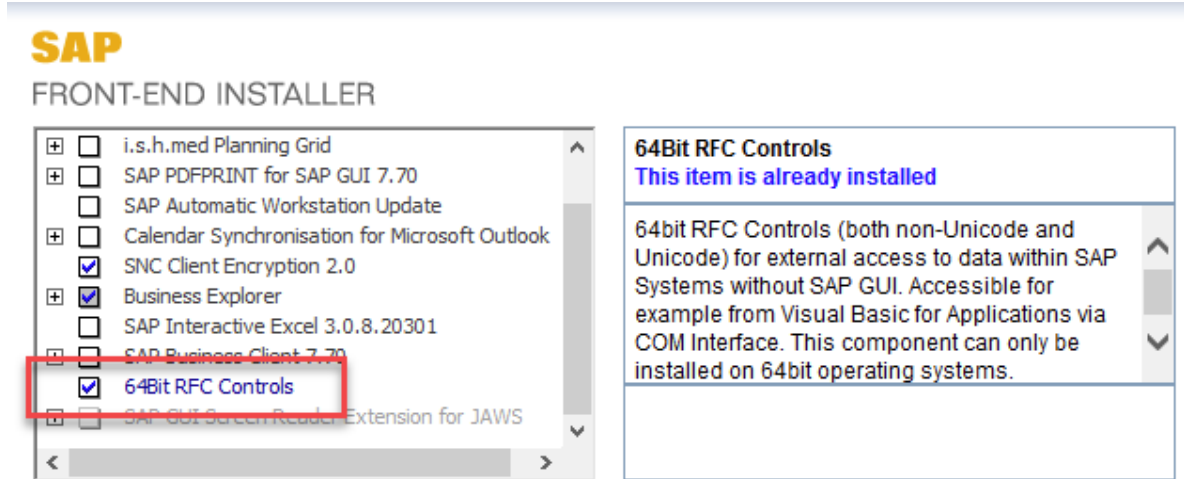
Connect for SAP® uses SAP NetWeaver RFC library internally. The recommended way to install the required RFC DLLs is to install the add-on “Business Explorer” distributed by SAP along with SAPGUI.

The main steps of the installation are shown below:

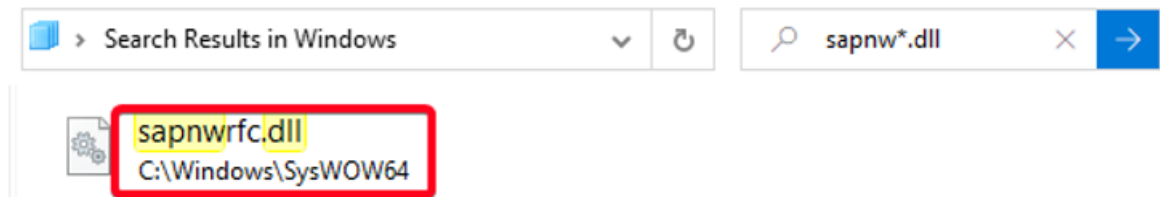
1. Start the SAPGUI installer.



2. Select (at least) the following components.



3. After installation, all files required for NetWeaver RFC are added to the system.

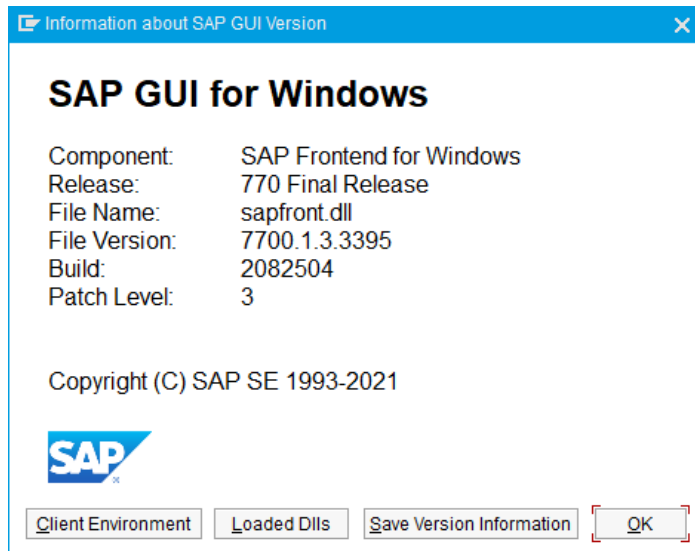


**Note:** Use SAPGUI of version 7.70 and higher for the automatic installation of NetWeaver RFC DLLs.

In case, you have already installed Classic RFC DLLs, NetWeaver RFC DLLs will be installed by SAPGUI 7.70 in parallel.




The exact version of your SAPGUI installation can be found by using the “About” window of SAP Logon.



### 4.3 Manual Installation of RFC Library Binaries

The process of the RFC library installation is described in details in <http://wiki.scn.sap.com/wiki/display/ABAPConn/Download+and+Installation+of+NW+RFC+SDK>.



**Hint:** To install NetWeaver RFC for x64 environment use the manual installation described in this topic.

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. The approach assumes that they are rather already installed a workstation.

During the installation the following set of files has to be copied to the system folder according to **Table 1**

File	Description
<i>sapnwrfc.dll</i>	Specifies the RFC library functions.
<i>icudt.dll</i>	Internal code page converter.
<i>icuuc50.dll</i>	Internal code page converter.
<i>icuin50.dll</i>	Internal code page converter.

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:

1. Use the paths defined by the Environment Variable *PATH* starting from the first path in the list. If an appropriate DLL is not found, then continue the search.
2. Use standard locations depending on OS and the application platform. The next table specifies standard locations.

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

It means that if you plan to run your 32-bit application on 64-bit Windows then the RFC library files should be 32-bit and be 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 is an alias for the %Windows%\SysWOW64 in the host 64-bit system.

There are cases when 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.

#### 4.4 Requirements for SAP Server

As mentioned in 4.2 Connect for SAP® uses SAP NetWeaver RFC library. That means NetWeaver must be supported on SAP server as well. SAP guarantees that NetWeaver RFC runs for all modern versions of SAP server (see <http://help.sap.com/netweaver>). More details can be found under <http://scn.sap.com/community/icc/blog/2012/08/15/support-for-classic-rfc-library-ends-march-2016> (see the FAQ section).

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.	TSAPxNWClientFunctionGS, TSAPxNWClientCompFunctionGS
RFC_GET_FUNCTION_INTERFACE_US	To dynamically obtain the function interface from Unicode servers.	TSAPxNWClientFunctionGS, TSAPxNWClientCompFunctionGS
RFC_GET_STRUCTURE_DEFINITION	To dynamically obtain the record data type layout from non-Unicode servers.	TSAPxNWClientFunctionGS, TSAPxNWClientCompFunctionGS, TSAPxNWTableGS, TSAPxNWClientCompTableGS
RFC_GET_UNICODE_STRUCTURE	To dynamically obtain the record data type layout from Unicode servers.	TSAPxNWClientFunctionGS, TSAPxNWClientCompFunctionGS, TSAPxNWTableGS, TSAPxNWClientCompTableGS
RFC_SYSTEM_INFO	To get the server representation of integer and float data types.	TSAPxNWClientConnectionGS, TSAPxNWClientCompConnectionGS
RFC_READ_TABLE	To read SAP server tables data.	TSAPxNWClientDataMoveGS, TSAPxNWClientCompServerTableGS
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

## 4.5 Installing into Embarcadero Delphi or C++ Builder

### 4.5.1 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 BAT files have naming like `_compile<Tool>.bat` (e.g. `_compileDelphiXE6.bat`). Each of them compiles Connect for SAP® packages (both runtime and design time) for required IDE version and copies result packages to the location specified by the IDE. If a target IDE is not installed an error is raised.

The compiled binary files will be put into the `<SAPx>\Lib\<Tool>` folder. For example, the Delphi XE6 files will be put into `<SAPx>\Lib\DelphiXE6`.

### 4.5.2 Installing Components

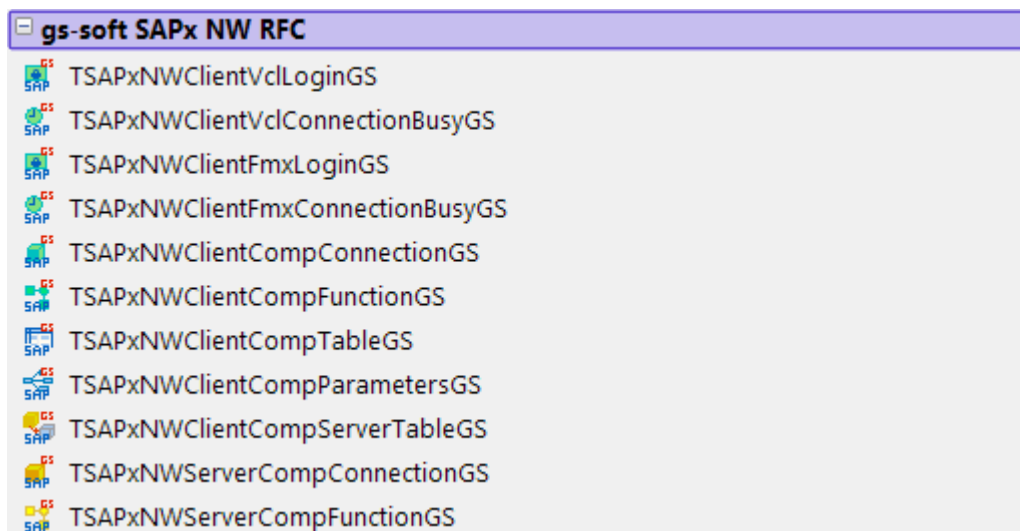
Installation of new components has become easy due to the Delphi package system. To install Connect for SAP®:

1. Run Delphi IDE.
2. Choose File -> Open. Set Files of type to Delphi package (\*.dpc) 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 `gsSAPxNW<Suffix>.dpc` and `gsSAPxNW<Suffix>D.dpc`. As well there are packages for VCL and FireMonkey frameworks, which named `gsSAPxNW<Suffix>VCL.dpc` and `gsSAPxNW<Suffix>FMX.dpc` correspondingly. In the names `<Suffix>` value can be `d20`, `d21`... (Delphi / C++ Builder XE6, XE7...).



**Note:** Since Delphi XE8 both VCL and FMX packages are supported. Earlier versions have VCL packages only.

3. Right-click on the project in the Project Manager window and choose Compile in the context menu. To install the design time package after compilation click Install in the menu.
4. The new components should appear in the Tool Palette as shown on the next figure.



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

## 5 Connectivity to SAP

The Connect for SAP® internally calls the RFC library and passes all RFC-specific parameters needed to connect to a SAP system, or to register an RFC server program at a gateway and wait for RFC calls from any SAP system.

These parameters can be specified either directly in code or in external files.

The first approach is simpler to implement. However, if you want to change or add parameters for your external RFC connections without the need to change the program code, the second approach is used. This approach can use both following files:

- The native SAP RFC connection file to store all connection definitions, which can be shared among all programs accessing SAP from the PC (see **Using sapnwrfc.ini** for more details)
- The specific Connect for SAP® connection alias file, which stores connection definitions for client applications (see **5.1.1**)

### 5.1 Client Connection

#### 5.1.1 Connection Alias

A connection alias is a named persistent set of client connection parameters. Connect for SAP® applications may use the aliases to connect to SAP systems. The connection aliases are stored in ini-file. The application can use either default alias file or file specified at the application level. Before using the default file its full path must be defined under the registry key `HKCU\Software\gs-soft\SAPx\CfgFile`. Use the Explorer tool to perform operations with the alias file (see **6.1**)

#### 5.1.2 Connection Parameters Priority

Connection parameters can be defined at different levels. The resulting connection string transmitted to SAP system is formed from different sources in accordance with the following priorities (higher first):

1. Parameters from `TSAPxNWClientConnectionGS.Params` properties defined at runtime.
2. Parameters from `TSAPxNWClientConnectionGS.Params` properties defined at design time.
3. Alias file explicitly specified in the `AliasFileName` property.
4. Default alias file.
5. Parameters from the section `sapnwrfc.ini` which is referenced with the `DEST` property.

As a result, the parameter value is determined by the level with the highest priority. Immediately before opening, the connection checks that all required parameters are specified and raises an error otherwise. The set of required parameters depends on whether the SNC mode is used:

- Normal mode: `CLIENT`, `USER`, `PASSWD` and `LANG`
- SNC mode: `SNC_PARTNERNAME`

However, in case when the `DEST` parameter is defined, remaining parameters may be omitted. In this case, the connection assumes that all required parameters are specified in the `sapnwrfc.ini` file.

### 5.2 Server Connection

RFC server connection parameters are defined in the `TSAPxNWServerConnectionGS.CommandLine` property. You can either specify all RFC specific parameters or refer to a destination in the `sapnwrfc.ini` file. You can find information about possible parameters in the appendix **Server Parameters**.

## 6 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 SAP system. The Explorer carries out the following functions:

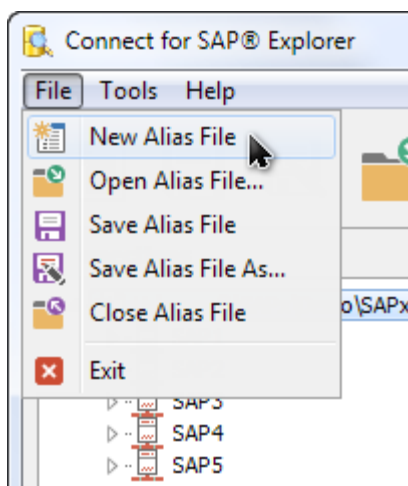
- Creating and maintaining the Connect for SAP® connection aliases
- Getting a metadata for the SAP RFC objects (functions, structures, tables)
- Executing RFC functions
- Generating Delphi wrapping source code for using RFC functions

### 6.1 Creating and Maintaining Connection Aliases

The Explorer allows defining the default alias file used by Connect for SAP® applications (see 5.1.1 for more information about aliases).

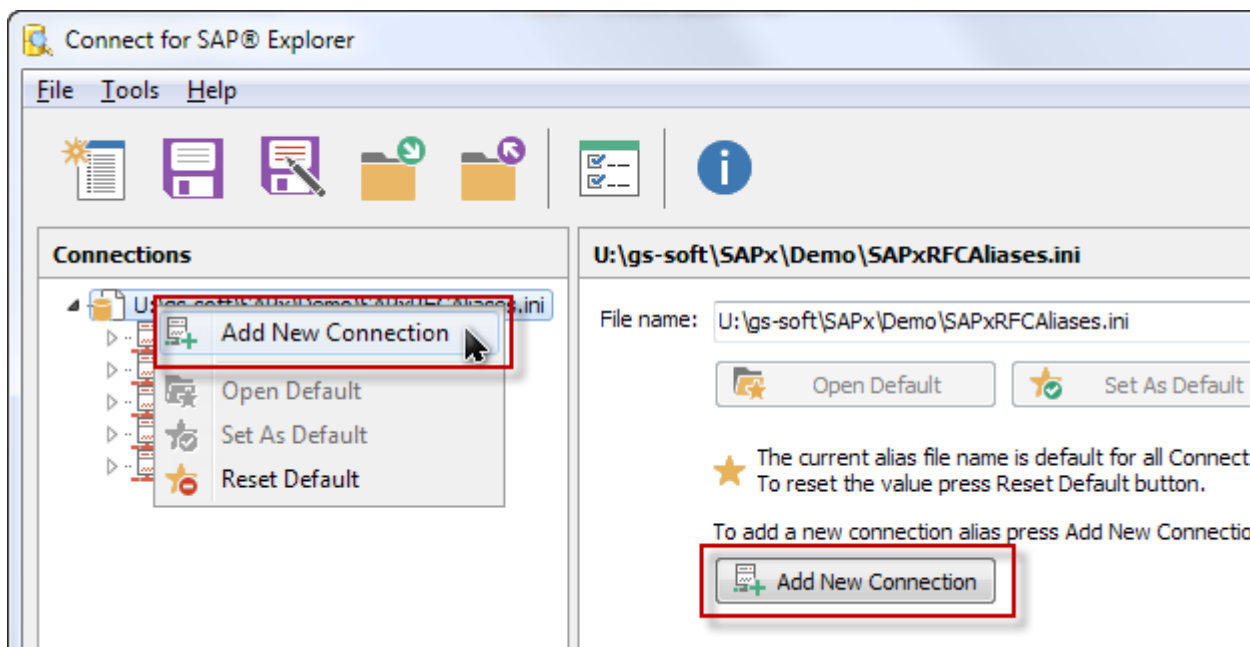
#### 6.1.1 Creating or Opening a Connection Alias File

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.



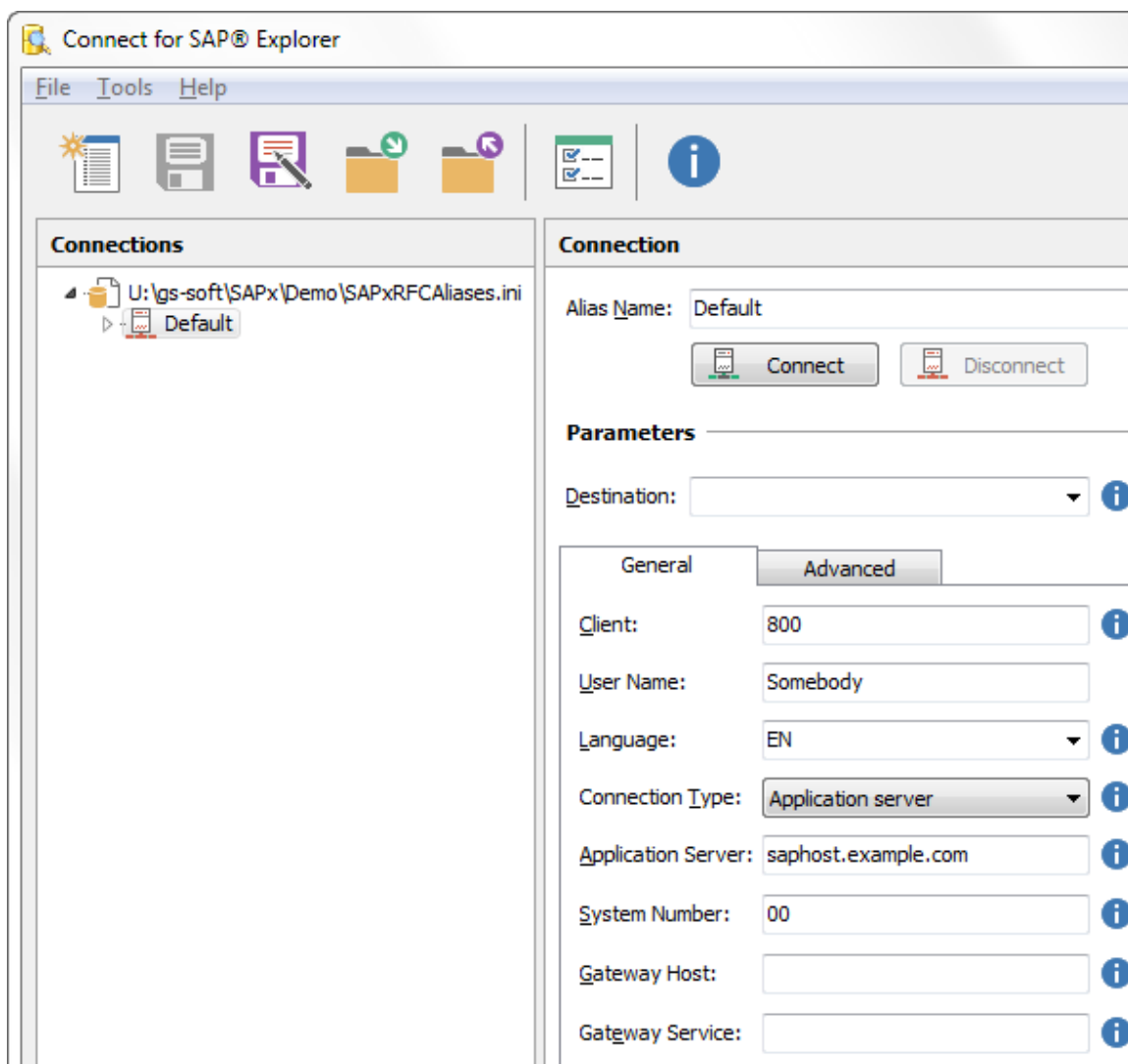
#### 6.1.2 Creating a New Connection Alias

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.



### 6.1.3 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 SAP R/3 server the following parameters are required:

- Client
- User Name
- Language
- Connection Type (normally it should be set to “Application server”)
- Application Server
- System Number

The Destination parameter allows to refer to a connection definition in the SAP RFC ini-file. You can either specify the destination by hand or choose from the Destination dropdown list.

The list publishes all destination entries from the SAP RFC ini-file. The path to the file can be specified either in the system environment variable RFC\_INI. If not, the explorer tool looks for the sapnwrfc.ini file in its working directory.

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

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.

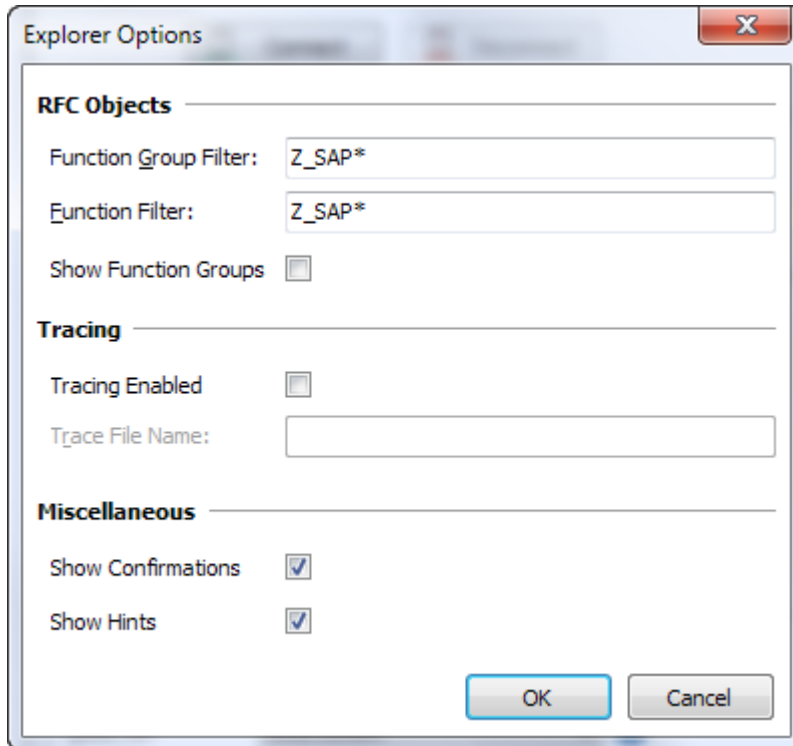


## 6.2 Browsing SAP Dictionary Information of RFC Functions

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

### 6.2.1 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, there are two text input fields: 'Function Group Filter' and 'Function Filter', both containing the text 'Z\_SAP\*'. Below them is a checkbox labeled 'Show Function Groups' which is currently unchecked.  
- In the 'Tracing' section, there is a checkbox labeled 'Tracing Enabled' which is unchecked, and a text input field labeled 'Trace File Name' which is empty.  
- In the 'Miscellaneous' section, there are two checkboxes: 'Show Confirmations' and 'Show Hints', both of which are checked.  
At the bottom right of the dialog, there are two buttons: 'OK' and 'Cancel'.

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.

## 6.2.2 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: [Dropdown] ⓘ

General **Advanced**

Client: 800 ⓘ

User Name: Somebody

Language: EN ⓘ

Connection Type: Application server ⓘ

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

**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: [Empty]

Application: [Empty]

**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.

### 6.3 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 on the right panel to show the Execute RFC Function form.
3. Specify the necessary Import Parameters, Var Parameters and Tables fields.
4. Press Execute to run the function.
5. Check the Export parameters and the Table fields as soon as the function is executed.

### 6.4 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.

### 6.4.1 Generating Wrapping Code

To generate an Object Pascal code, you should:

1. Select an RFC function in the tree.
2. Press Generate 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: TSAPxNW%sFunctionGS
  - Table: TSAPxNW%sTableGS
  - Structure: TSAPxNW%sStructureGS
- Base Classes:**
  - Function: TSAPxNWClientFunctionGS
  - Table: TSAPxNWTableGS
  - Structure: TSAPxNWParameterGS
- Structured Data Types:** A table listing various data types and their corresponding class names and units.

Object	Type	Class Name	Base Class Name	Unit
T000	Str	TSAPxNWT000Struc	TSAPxNWParamete	gsSAPxNWZ_S
T000	Tab	TSAPxNWT000Table	TSAPxNWTableGS	gsSAPxNWZ_S
T042	Tab	TSAPxNWT042Table	TSAPxNWTableGS	gsSAPxNWZ_S
Z_SAPX_DU_CLIENT_TRANSITION	Fnc	TSAPxNWZ_SAPX_I	TSAPxNWClientFun	gsSAPxNWZ_S

At the bottom of the dialog, there are three buttons: "Refresh Structured Data Types", "Generate" (which is highlighted with a blue border), 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.

## 6.4.2 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 TSAPxNWClientCompConnectionGS.RfcConnection

## 7 Migrate Code to New Connect for SAP®

The previous versions of the Connect for SAP® library (up to version 3.x) have been using [Classic RFC library](#) internally. However, support of the RFC library will be [stopped by SAP after 31 March 2016](#).

The new 4.x version of the Connect for SAP® library uses [NetWeaver RFC library](#) internally.

Due to changes in API between the Classic and NetWeaver SAP RFC libraries, API of Connect for SAP® got some changes also. It means that applications using Connect for SAP® have to be updated to the changes.

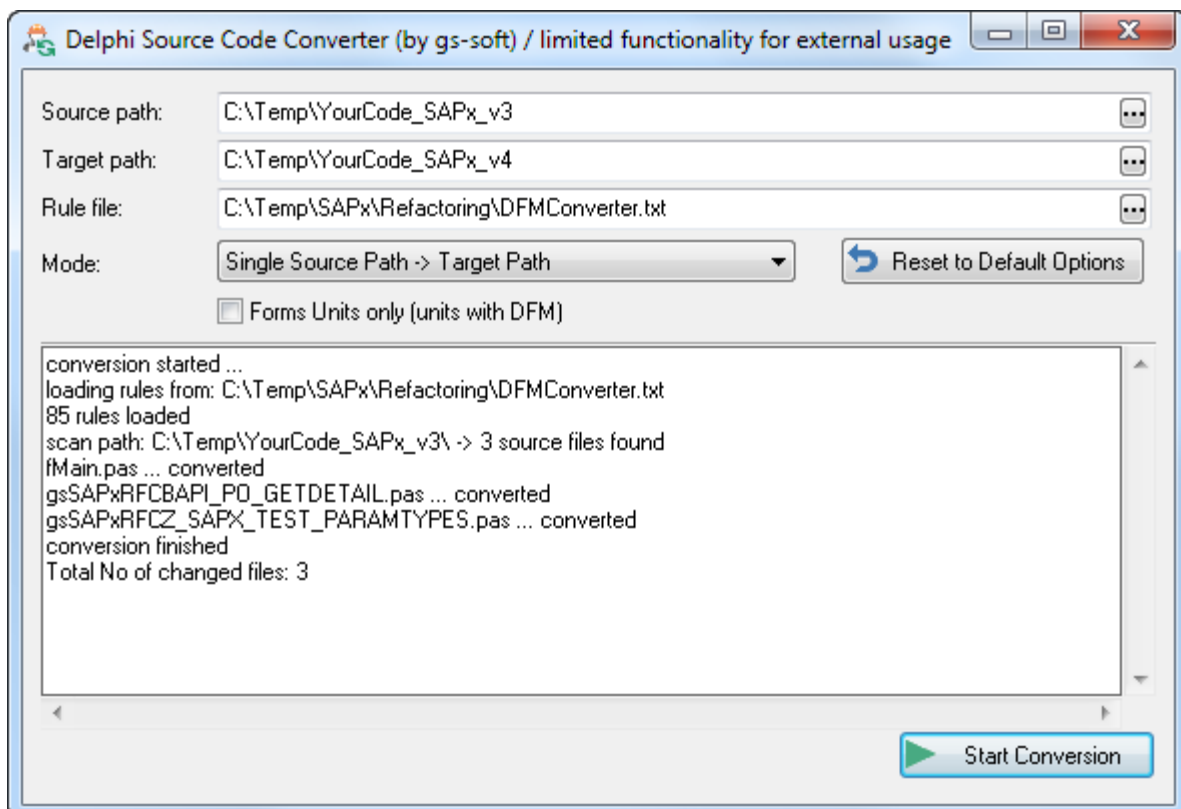
**Appendix D** shows changes of Connect for SAP® API between 3.x and 4.x versions.

The migration of users' applications can significantly be simplified by usage of *DFM Converter*. It is a special tool for automatic replacing of "old" code with "new" according to rules described in **Appendix D**.

*DFM Converter* and the corresponding rule-file are placed in <SAPx>\Refactoring.

The following steps show migration of code with *DFM Converter*:

1. Run the application `DFMConverter.exe` from <SAPx>\Refactoring.
2. Choose a directory where your source files are located (your code using Connect for SAP® of 3.x version). Please note that files in child directories (if any) will not be processed during the conversion.
3. Choose a directory where target files will be stored to. Please note that only changed files will be placed to the target directory.
4. Choose a Rule file containing rules of the conversion. By default, you should use the Rule file `DFMConverter.txt` placed in the directory <SAPx>\Refactoring.
5. Press Start Conversion.
6. Results of the conversion are shown in the log.





**Note:** *DFMConverter* will not apply automatically changes listed in **Appendix E. Changes of Applications to be Applied Manually**.



**Note:** Some of rules inside the Rule file are disabled by default (see the last section of **DFMConverter.txt**). The reason is that the rules may introduce changes not related to the migration. You can activate them on your own risk but please carefully check results of the renaming.



**Note:** Code wrappers for your RFC functions must be regenerated using Connect for SAP® Explorer during the migration.

## 8 Troubleshooting

### 8.1 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 is in the file `<SAPx>\Docu\Connect for SAP - Issue Report.docx`.

### 8.2 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.

#### 8.2.1 SAP RFC Library Tracing

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

#### 8.2.2 “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 `gsSAPxNWCore.SAPxNWSetTracing` call is used. If you do not specify exact file name for the trace file name it will have a name like `SAPxNW<xxxxxxx>` and be in the working directory of your application.



**Important Note:** To have a possibility to activate the tracing system in the application it must be built with the **SAPX\_TRACE compiler directive**. This directive is by default enabled (see `gsSAPxNW.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. Using sapnwrfc.ini

To use the sapnwrfc.ini file it must be in the same directory as the RFC client/server program, or you can define it with full path and file name by the environment variable RFC\_INI.

All connection parameters that can be used in the sapnwrfc.ini are divided into 3 different categories:

- General Connection parameters. These can be used with client and server programs
- Parameters used in client programs
- Parameters depending on the connection type you use

The typical connectivity types with required parameters are listed in the client and server sections.

See more details at <https://help.sap.com/viewer/index> (use “The SAPNWRFC.INI File” in the search).

### A.1 Client Parameters

The following part of the sapnwrfc.ini file shows how to define two typical use cases:

- Client using a specific application server (see MY\_SAP\_SYS107)
- Client using load balancing (see MY\_SAP\_VIA\_LOADBALANCER)

#### Listing 3: SAP System using Load Balancing and Application Server

```
DEST=MY_SAP_SYS107
ASHOST=<host name of a specific SAP application server>
SYSNR=<SAP system number>
GWHOST=<optional; default: gateway on application server>
GWSERV=<optional; default: gateway on application server>
RFC_TRACE= <0/1: OFF/ON, optional; default:0(OFF)>

DEST=MY_SAP_VIA_LOADBALANCER
R3NAME=<name of SAP system, optional; default: destination>
MSHOST=<host name of the message server>
GROUP=<group name of the application servers, optional; default: PUBLIC>
RFC_TRACE=<0/1: OFF/ON, optional; default:0(OFF)>
```

### A.2 Server Parameters

The following table contains parameters to specify a server connection in the sapnwrfc.ini file.

Server connection parameters	Description
DEST	Destination name
PROGID	Identifier of the server connection registered on SAP gateway
GWHOST	Host name of SAP gateway
GWSERV	Service name on SAP gateway
TRACE	Indicator of tracing

#### Listing 4: RFC server registered at an SAP gateway

```
DEST=MY_RFCSERVER_REG
PROGID=<program ID, optional; default: destination>
GWHOST=<host name of the SAP gateway>
GWSERV=<service name of the SAP gateway>
RFC_TRACE=<0/1: OFF/ON, optional; default:0(OFF)>
```

In an SAP system the program ID and this SAP gateway must be specified in the Destination entry defined with transaction SM59: use connection type T and register mode.

## 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 must 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 TSAPxNWServerCompConnectionGS 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.

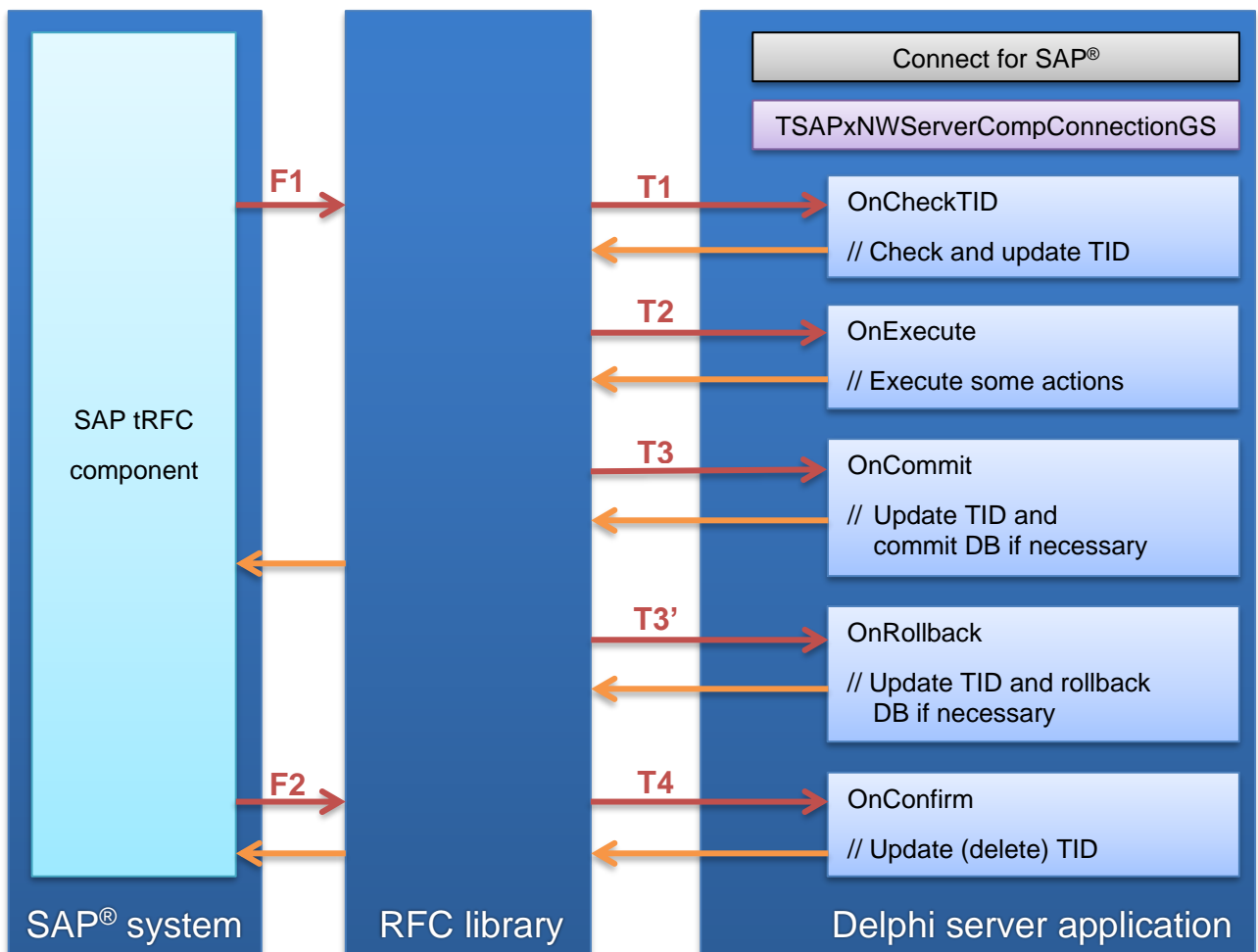


Figure 4: Scheme of calling a transactional function

## Appendix C. Connect for SAP® Component List

The Connect for SAP® components are divided into three groups: components for client programs, components for GUI-based client programs (VCL/FMX) and for non-SAP server programs. In the following sections you can find a description of each of these groups.

### C.1 Client components



#### TSAPxNWClientCompConnectionGS

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



#### TSAPxNWClientCompTableGS

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



#### TSAPxNWClientCompParametersGS

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



#### TSAPxNWClientCompFunctionGS

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.



#### TSAPxNWClientCompServerTableGS

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.

### C.2 Client components for VCL and FMX



#### TSAPxNWClientVclLoginGS, TSAPxNWClientFmxLoginGS

Components for registering of the corresponding login form used for authentication.



#### TSAPxNWClientVclConnectionBusyGS, TSAPxNWClientFmxConnectionBusyGS

Components for registering of Busy-handler used for displaying of Wait-cursor.

### C.3 Server components



#### TSAPxNWServerCompFunctionGS

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.



#### TSAPxNWServerCompConnectionGS

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

## Appendix D. Changes of Connect for SAP® API between 3.x and 4.x

Renamed units:

v3.x name	v4.x name
gsSAPxRFCBase	gsSAPxNWCoreBaseTypes
gsSAPxRFCStdObj	gsSAPxNWCore
gsSAPxRFCUtil	gsSAPxNWCoreBaseClasses
gsSAPxRFCServer	gsSAPxNWServer
gsSAPxRFCClient	gsSAPxNWClient
gsSAPxRFCStdFunc	gsSAPxNWClientFunctions
gsSAPxRFCAllTools	gsSAPxNWClientUtils
gsSAPxRFCDataMove	gsSAPxNWClientDataMove
gsSAPxRFCvStdObj	gsSAPxNWCoreComp
gsSAPxRFCvServer	gsSAPxNWServerComp
gsSAPxRFCvClient	gsSAPxNWClientComp
gsfSAPxRFCLogin	gsSAPxNWClientVclLoginFrm
gsSAPxRFCReg	gsSAPxNWReg

Renamed classes:

v3.x name	v4.x name
TSAPxMulticastErrorEventGS	TSAPxNWMulticastErrorEventGS
TSAPxRFCFunctionGS	TSAPxNWClientFunctionGS
TSAPxRFCAliasGS	TSAPxNWClientAliasGS
TSAPxRFCCustomAliasListGS	TSAPxNWClientCustomAliasListGS
TSAPxRFCRegistryAliasListGS	TSAPxNWClientRegistryAliasListGS
TSAPxRFCFileAliasListGS	TSAPxNWClientFileAliasListGS
TSAPxRFCsrvConnectionThreadGS	TSAPxNWServerConnectionThreadGS
TSAPxMulticastCheckTIDEEventGS	TSAPxNWServerMulticastCheckTIDEEventGS
TSAPxMulticastTIDEEventGS	TSAPxNWServerMulticastTIDEEventGS
TSAPxMulticastCommitEventGS	TSAPxNWServerMulticastCommitEventGS
TSAPxMulticastRollbackEventGS	TSAPxNWServerMulticastRollbackEventGS
TSAPxMulticastConfirmEventGS	TSAPxNWServerMulticastConfirmEventGS
TSAPxRFCTracerMsgGS	TSAPxNWTraceMessageGS
TSAPxRFCTracerTechMsgGS	TSAPxNWTraceTechMessageGS
TSAPxMulticastEventBaseGS	TSAPxNWMulticastEventGS
TSAPxMulticastNotifyEventGS	TSAPxNWMulticastNotifyEventGS
TSAPxRFCIniFileGS	TSAPxNWClientIniFileGS
TSAPxRFCTextTableGS	TSAPxNWClientTextTableGS

TSAPxRFCLogFileGS	TSAPxNWClientLogFileGS
TSAPxRFCEasyDataMoveGS	TSAPxNWClientDataMoveGS
TSAPxRFCFunctionFactoryGS	TSAPxNWClientFunctionFactoryGS
TSAPxRFCvCustomRemoteObjectGS	TSAPxNWClientCompCustomRemoteObjectGS
TSAPxRFCvClientConnectionGS	TSAPxNWClientCompConnectionGS
TSAPxRFCvRemoteObjectGS	TSAPxNWClientCompRemoteObjectGS
TSAPxRFCvFunctionGS	TSAPxNWClientCompFunctionGS
TSAPxRFCvCustomTableGS	TSAPxNWClientCompCustomTableGS
TSAPxRFCvBlobStreamGS	TSAPxNWClientCompBlobStreamGS
TSAPxRFCvTableGS	TSAPxNWClientCompTableGS
TSAPxRFCvParamsMapItemGS	TSAPxNWClientCompParametersMapItemGS
TSAPxRFCvParamsMapGS	TSAPxNWClientCompParametersMapGS
TSAPxRFCvParamsGS	TSAPxNWClientCompParametersGS
TSAPxRFCvServerTableGS	TSAPxNWClientCompServerTableGS
TSAPxRFCvServerConnectionGS	TSAPxNWServerCompConnectionGS
TSAPxRFCvServerFunctionGS	TSAPxNWServerCompFunctionGS
TSAPxMulticastActiveChangeEventGS	TSAPxNWCoreCompActiveChangeEventGS
TSAPxRFCvCustomConnectionGS	TSAPxNWCoreCompCustomConnectionGS
TSAPxRFCvCustomServerFunctionGS	TSAPxNWServerCompCustomFunctionGS
TSAPxRFCvCustomObjectGS	TSAPxNWCoreCompCustomObjectGS
TSAPxRFCListObjectGS	TSAPxNWItemGS
TSAPxRFCObjectsListGS	TSAPxNWItemCollectionGS
TSAPxRFCParametersListGS	TSAPxNWParameterCollectionGS
TSAPxRFCFieldsListGS	TSAPxNWFieldCollectionGS
TSAPxRFCTablesListGS	TSAPxNWTableCollectionGS
TSAPxRFCEnvironmentGS	TSAPxNWEnvironmentGS
TSAPxRFCTracingObjectGS	TSAPxNWTracingObjectGS
TSAPxRFCFuncWrapperGS	TSAPxNWFuncWrapperGS
TSAPxRFCStringBuilderGS	TSAPxNWStringBuilderGS
TSAPxRFCTracerGS	TSAPxNWTracerGS
TSAPxRFCTraceFileWriterGS	TSAPxNWTraceFileWriterGS
TSAPxRFCTraceWriterBaseGS	TSAPxNWTraceWriterBaseGS
TSAPxRFCServerApplicationGS	TSAPxNWServerApplicationGS
TSAPxRFCServerConnectionListGS	TSAPxNWServerConnectionListGS
TSAPxRFCServerConnectionGS	TSAPxNWServerConnectionGS
TSAPxRFCServerCommandLineGS	TSAPxNWServerCommandLineGS
TSAPxRFCServerFunctionGS	TSAPxNWServerFunctionGS

TSAPxRFCClientConnectionGS	TSAPxNWClientConnectionGS
TSAPxRFCRemoteObjectGS	TSAPxNWRemoteObjectGS
TSAPxRFCCustomConnectionGS	TSAPxNWCustomConnectionGS
TSAPxRFCConnectedObjectGS	TSAPxNWConnectedObjectGS
TSAPxRFCTableGS	TSAPxNWTableGS
TSAPxRFCFieldGS	TSAPxNWFieldGS
TSAPxRFCParameterGS	TSAPxNWParameterGS
TSAPxRFCValueAdapterGS	TSAPxNWImportExportValueAdapterGS
TSAPxRFCDataFormatGS	TSAPxNWDataFormatGS
TSAPxRFCConnectionAttributesGS	TSAPxNWConnectionAttributesGS
TSAPxRFCObjectGS	TSAPxNWObjectGS
TSAPxRFCObjectBaseGS	TSAPxNWObjectBaseGS

Renamed types:

v3.x name	v4.x name
TSAPxRFCFunctionCallTypeGS	TSAPxNWFunctionCallTypeGS
TSAPxRFC SapTypeGS	TSAPxNWSapTypeGS
TSAPxRFC AliasPasswordModeGS	TSAPxNW AliasPasswordModeGS
TSAPxRFC UseSapGUIGS	TSAPxNW UseSapGUIGS
TSAPxRFC SncQopGS	TSAPxNWSncQopGS
TSAPxRFC CharConversionErrorModeGS	TSAPxNW CharConversionErrorModeGS
TSAPxRFC ErrorActionGS	TSAPxNW ErrorActionGS
TSAPxRFC DataTypeGS	TSAPxNW DataTypeGS
TSAPxRFC DataFormatValueGS	TSAPxNW DataFormatValueGS
TSAPxRFC DataFormatValuesGS	TSAPxNW DataFormatValuesGS
TSAPxRFC MapBCDToTypeGS	TSAPxNW MapBCDToTypeGS
TSAPxRFC IntTypeGS	TSAPxNW IntTypeGS
TSAPxRFC FloatTypeGS	TSAPxNW FloatTypeGS
TSAPxRFC BytesPerCharGS	TSAPxNW BytesPerCharGS
TSAPxRFC ReadTableFormatGS	TSAPxNW ReadTableFormatGS
TSAPxRFC ValueFormatGS	TSAPxNW ValueFormatGS
TSAPxRFC ParameterTypeGS	TSAPxNW ParameterTypeGS
TSAPxRFC TableStateGS	TSAPxNW TableStateGS
TSAPxRFC TreeStateGS	TSAPxNW TreeStateGS
TSAPxRFC RemoteObjectDefinitionStateGS	TSAPxNW RemoteObjectDefinitionStateGS
TSAPxRFC TraceMessageTypeGS	TSAPxNW TraceMessageTypeGS
TSAPxRFC CheckTIDResultGS	TSAPxNW CheckTIDResultGS
TSAPxRFC SrvConThreadStateGS	TSAPxNW SrvConThreadStateGS

TSAPxRFCsRvAppActionGS	TSAPxNWSrvAppActionGS
TSAPxRFCsRvAppStateGS	TSAPxNWSrvAppStateGS
TSAPxRFCsRvAppStatesGS	TSAPxNWSrvAppStatesGS
TSAPxRFCActiveChangeEventGS	TSAPxNWActiveChangeEventGS
TSAPxRFCTxCheckTIDEventGS	TSAPxNWTxCheckTIDEventGS
TSAPxRFCTxTIDEventGS	TSAPxNWTxTIDEventGS
TSAPxRFCConnectionLoginEventGS	TSAPxNWConnectionLoginEventGS
TSAPxRFCWaitEventGS	TSAPxNWWaitEventGS
TSAPxRFCErrorEventGS	TSAPxNWErorEventGS
TSAPxRFCServerFunctionExecuteEventGS	TSAPxNWServerFunctionExecuteEventGS

Renamed constants:

v3.x name	v4.x name
SSAPxRFCDefCfgFileName	SSAPxNWDefCfgFileName
SSAPxRFCDefRootKey	SSAPxNWDefRootKey
SSAPxRFCDefRootKeyOld	SSAPxNWDefRootKeyOld
SSAPxRFCDefCfgKeyName	SSAPxNWDefCfgKeyName
SSAPxRFCCfgKeyName	SSAPxNWCfgKeyName
SSAPxRFCCfgValName	SSAPxNWCfgValName

Renamed global variables:

v3.x name	v4.x name
FSAPxRFCEnvironment	TSAPxNWEEnvironmentGS.Instance

Renamed functions:

v3.x name	v4.x name
SAPxRFCInitialize	SAPxNWInitialize
SAPxRFCFinalize	SAPxNWFinalize
SAPxRFCRaiseException	SAPxNWRaiseException
SAPxRFCRaiseNativeException	SAPxNWRaiseNativeException
SAPxRFCRaiseCustomException	SAPxNWRaiseCustomException
SAPxRFCGetSAPRFCINIFilename	SAPxNWGetSAPRFCINIFilename
SAPxRFCSetTracing	SAPxNWSetTracing
SAPxRFCGetTracing	SAPxNWGetTracing
SAPxRFCStrToAPIStr	SAPxNWStrToAPIStr
SAPxRFCMemFill	SAPxNWMemFill
SAPxRFCMemMove	SAPxNWMemMove
SAPxRFCReverseBytes	SAPxNWRReverseBytes

SAPxRFCPadLeft	SAPxNWPadLeft
SAPxRFCTrimAll	SAPxNWTrimAll
SAPxRFCStrToCh	SAPxNWStrToCh
SAPxRFCCharInSet	SAPxNWCharInSet
SAPxRFCUtf8Decode	SAPxNWUtf8Decode
SAPxRFCGetVersionInfo	SAPxNWGetVersionInfo
SAPxRFCGetModuleName	SAPxNWGetModuleName
SAPxRFCToolVer	SAPxNWToolVer
SAPxRFCServerApplication	SAPxNWServerApplication



## Appendix E. Changes of Applications to be Applied Manually

Changed members:

Member's class (v3.x name)	v3.x member	v4.x member
TSAPxRFCAliasGS	property ConnectionString: String	property ConnectionParameters: TSAPxNWConnectionParamsGS
	property RFC_TRACE: Boolean	property TRACE: TSAPxNWSapTraceGS
ISAPxRFCErrorHandlerGS	function OnError(const AException: Exception; ASender: TObject): Boolean	function OnListError(const AException: Exception; ASender: TSAPxNWItemCollectionGS; AContainer: ISAPxNWNameProviderGS): Boolean
TSAPxRFCValueAdapterGS	property AsBCD: Currency	property AsBCD: TBcd
TSAPxRFCTableGS	property RecordSize: LongWord	Fields.DataSize
TSAPxRFCFieldsListGS	function AddFieldEx(AClass: TSAPxRFCFieldClassGS): TSAPxRFCFieldGS	function Add(AClass: TSAPxNWItemClassGS): TSAPxNWFieldGS
	function AddField: TSAPxRFCFieldGS	function Add: TSAPxNWFieldGS
	function FieldByName(const AName: String): TSAPxRFCFieldGS	function ItemByName(const AName: String): TSAPxNWFieldGS
	function FindField(const AName: String): TSAPxRFCFieldGS;	function FindItem(const AName: String): TSAPxNWFieldGS;
	property Fields[AIndex: integer]: TSAPxRFCFieldGS	property Items[AIndex: integer]: TSAPxNWFieldGS
TSAPxRFCParametersListGS	function AddParameter: TSAPxRFCParameterGS	function Add: TSAPxNWParameterGS
	function AddParameterEx(AClass: TSAPxRFCParameterClassGS): TSAPxRFCParameterGS;	function Add(AClass: TSAPxNWItemClassGS): TSAPxNWParameterGS
	function ParameterByName(const AName: String): TSAPxRFCParameterGS	function ItemByName(const AName: String): TSAPxNWParameterGS
	property Parameters[AIndex: integer]: TSAPxRFCParameterGS	property Items[AIndex: integer]: TSAPxNWParameterGS
	function FindParameter(const AName: String): TSAPxRFCParameterGS	function FindItem(const AName: String): TSAPxNWParameterGS
TSAPxRFCTablesListGS	function AddTable: TSAPxRFCTableGS	function Add: TSAPxNWTableGS

	function AddTableEx(AClass: TSAPxRFCTableClassGS): TSAPxRFCTableGS	function Add(AClass: TSAPxNWItemClassGS): TSAPxNWTableGS
	function TableByName(const AName: String): TSAPxRFCTableGS	function ItemByName(const AName: String): TSAPxNWTableGS
	function FindTable(const AName: String): TSAPxRFCTableGS	function FindItem(const AName: String): TSAPxNWTableGS
	property Tables[AIndex: integer]: TSAPxRFCTableGS	property Items[AIndex: integer]: TSAPxNWTableGS
TSAPxRFCEasyDataMoveGS	property Table: TSAPxRFCTableGS	property Table: ISAPxNWTableGS
TSAPxRFCServerCommandLineGS	property Argv: ppRfc_char_t	property ConnectionParameters: TSAPxNWConnectionParamsGS
	property RFC_TRACE: Boolean	property TRACE: TSAPxNWSapTraceGS
TSAPxRFCvCustomTableGS	property RFCTable: TSAPxRFCTableGS	property RFCTable: ISAPxNWTableGS
TSAPxRFCvCustomConnectionGS	procedure AllowStartProgram(const APrograms: array of String);	function AllowStartProgram(AArgCount: Integer; AArgumens: PPSAP_UC; const AConnectionParams: array of RFC_CONNECTION_PARAMETER): RFC_CONNECTION_HANDLE;
TSAPxRFCCustomConnectionGS	procedure AllowStartProgram(const APrograms: array of String);	function AllowStartProgram(AArgCount: Integer; AArgumens: PPSAP_UC; const AConnectionParams: array of RFC_CONNECTION_PARAMETER): RFC_CONNECTION_HANDLE;
TSAPxRFCvParamsGS	ParamKinds	ParameterTypes

Removed items:

v3.x name	Comment
dtXMLDataGS	XML Data type is not supported in NetWeaver and Connect for SAP® 4.x version
TSAPxRFCvParamSetKindGS	Use TSAPxNWPParameterTypesGS
TSAPxRFCvParamKindGS	Use TSAPxNWPParameterTypeGS
TSAPxRFCvParamKindsGS	Use TSAPxNWPParameterTypesGS
TSAPxRFCLoginCallBackGS, FSAPxRFCLoginCallBack	Use TSAPxNWClientConnectionGS.RegisterLoginProvider

TSAPxRFCDataFormatGS .MapBCD(ADataSize, ADecimals: Integer): TSAPxRFCMapBCDToTypeGS;	Use BcdToXXX functions from Data.FmtBcd
TSAPxRFCAliasGS.WAN_CONN	Use "NO_COMPRESSION" instead
SAPxRFCIsTextNum	Not used anymore
SAPxRFCIsNumEmpty	Not used anymore
SAPxRFCCurrToStr	Not used anymore
SAPxRFCUtf8Encode	Not used anymore
SAPxRFCAnsiToUtf8	Not used anymore
SAPxRFCUtf8ToAnsi	Not used anymore
SAPxRFCWStrCopy	Not used anymore
SAPxRFCWideStrLen	Not used anymore