Book 2

Remote Program Calls

U:\Book\$01.doc

The mySAP Revolution

1 R/3 DCOM Connectivity

Access via RFC to R/3 from Windows is done via number of DLLs and Active/X controls provided by SAP with every SAPGUI installation. The provided DLLs are DCOM compliant interface classes and are accessed via their proper DCOM registration.

R/3 DCOM Connectivity - Calling RFC Functions And BAPIs From Windows™

Access via RFC to R/3 from Windows is done via number of DLLs and Active/X controls provided by SAP with every SAPGUI installation. The provided DLLs are DCOM compliant interface classes and are accessed via their proper DCOM registration. This chapter summarises in brief how to call the DCOM ActiveX components that are provided by SAP to access R/3 function modules from a Windows platform. We will demonstrate how to call the standard RFC function module RFC_READ_TABLE.

U:\Book\C03@01 DCOM.doc

R/3 DCOM Connectivity

Access via RFC to R/3 from Windows is done via number of DLLs and Active/X controls provided by SAP with every SAPGUI installation. The provided DLLs are DCOM compliant interface classes and are accessed via their proper DCOM registration. This chapter su

1.1 Calling RFC Functions And BAPIs From Windows™

This chapter summarises in brief how to call the DCOM ActiveX components that are provided by SAP to access R/3 function modules from a Windows platform. We will demonstrate how to call the standard RFC function module RFC_READ_TABLE.

Reading data from an arbitrary R/3 table with RFC_READ_TABLE

Here you can see the basic example how to call R/3 RFC functions from Visual Basic. This example will read the data from table T000 as also shown in the previous chapter, where we discussed the RFC function RFC_READ_TABLE. If you analyzed this example and understood how it works, I see no reason why you should not be able to create any other application, that links to R/3.

Apart from the logon data and the name of the table we need nothing else from R/3

SAP provides a set of interface OCX control and DLLs. They were written and compiled in Visual Basic, so the OCX are technically compatible with your Visual Basic. If there are still problems with compatibility, you better take it up with Microsoft, not with SAP. SAP is seen by Visual Basic as an object like any other object, e.g. the ADO-object, DAO-object or the FileSystem-Object. You need to know the methods and the meaning of the properties, but you do not need to know anything about R/3 to be able to use R/3 as an intelligent database server.

R/3 is a stored procedure database server

If you are completely ignorant about R/3 then you should regard R/3 as a transparent database server and the function modules as stored procedures of this database system.

Example that reads data from an R/3 table

This is a little VB Script example that demonstrates how you can call an RFC function module in R/3. The function used is function RFC_READ_TABLE which takes the name of a table as a parameter and returns its contents. As an option you can pass a Visual Basic recordset to contain a simple SQL WHERE-clause, which is added to the SQL statement issued by RFC_READ_TABLE.

The example is the basic template for every R/3 access

If you fully understand how this example works then you should be able to write any program you want to connect to R/3.

1.2 Visual Basic Code to Read Data from Table T000 Via RFC

He following documented code sequence is intended to give you the necessary insight in how to call an RFC function from Visual basic.

Declarations

Declare an R/3 Logon OCX component

```
DIM LogonControl
```

Declare an R/3 Connection object to become a member of the Logon OCX

```
DIM conn
```

Declare a pointer to the R/3 RFC function repository

```
DIM funcControl
```

Declare a record set pointer to pass an SQL WHERE-clause

```
DIM TableFactoryCtrl
```

Declare a pointer to the actual R/3 RFC

```
DIM RFC_READ_TABLE
```

Declare a pointer to every parameter to the function module

```
DIM eQUERY_TAB
DIM TOPTIONS
DIM TDATA
DIM TFIELDS

'**************************************************************
' Main Program
'**************************************************************
'--------------------------------------------------------------
   call Main
'--------------------------------------------------------------
```

Sub Routines

The program is neatly split in handy, easily digestible sub routines.

Login to R/3 via RFC with the logon Active/X control component

Create a new connection with method NewConnection

```
Sub R3Logon()
  Set conn = LogonControl.NewConnection
```

The login properties are the same as are found in the R/3 Logon Panel. Only the application server name and the system number are mandatory. If the other parameters are missing you will be prompted for them. Of course, if you run the login from a web server a dialogue is desirable.

Specifying the Login properties

```
  conn.ApplicationServer = "r3dev" ' IP or DNS-Name of the R/3
application server
```

Calling Logon method
Checking if logon has been successful

```
  conn.System = "00"               ' System ID of the instance,
usually 00
  conn.Client = "100"              ' opt. Client number to logon to
  conn.Language = "EN"             ' opt. Your login language
  conn.User = ""                   ' opt. Your user id
  conn.Password = ""               ' opt. Your password
  retcd = conn.Logon(0, False)
  If retcd <> True Then
    MsgBox " Cannot log on! "
    MsgBox retcd
    Stop
   else
    MsgBox " Logon OK."
  End If
End Sub
```

Calling an RFC Function Module

Calling an RFC function module can be done via the funcControl Active/X control and whoosh, an open RFC connection has been created in the logon step. The code will set pointers to local variables that hold the import and export parameters to the function and then the function call is executed. R/3 table parameters will be represented as Visual Basic recordsets. These recordsets will be automatically typed by the RFC call.

Create a new collection object for the FM

```
Sub R3RFC_READ_TABLE(pQueryTab)
'------------------------------------------------------------
' Call the R/3 RFC function RFC_READ_TABLE
'------------------------------------------------------------
  Set RFC_READ_TABLE = funcControl.Add("RFC_READ_TABLE")
```

Set pointers to local variables for the import and export parameters

```
  Set eQUERY_TAB = RFC_READ_TABLE.Exports("QUERY_TABLE")
  Set TOPTIONS   = RFC_READ_TABLE.Tables("OPTIONS")   '
  Set TDATA      = RFC_READ_TABLE.Tables("DATA")   '
  Set TFIELDS    = RFC_READ_TABLE.Tables("FIELDS")   '
```

Import, export and tables parameters of an R/3 function module are referenced with a Visual Basic object pointer. The function collection which we created above with the funcControl.Add method provides appropriate methods, Exports, Imports and Tables which create a correct parameter object and return a reference to this object.

Once the parameter object have been created, you can assign a value to the objects value property or read the value property respectively.

Reading and writing parameter values

```
  eQUERY_TAB.Value = pQueryTab
                                             ' pQueryTab is the R/3
name of the table
  TOPTIONS.AppendRow         ' new item line
  TOPTIONS(1,"TEXT") = "MANDT EQ '000'"
```

Once the parameter values have been set, you can call the function with the CALL-property. The property returns TRUE or FALSE according to whether the call has been successful or not.

Calling the RFC function

```
  If RFC_READ_TABLE.Call = True Then
```

When the RFC call has been successful you can output the result data or process them appropriately. For our demo we will display the first row of the returned recordset (= RFC table parameter) by means of the VBS message box.

Output the result

```
    If TDATA.RowCount > 0 Then
      MsgBox "Call to RFC_READ_TABLE successful! Data found"
      MsgBox TDATA(1, "WA")
     Else
      MsgBox "Call to RFC_READ_TABLE successful! No data found"
    End If
   Else
    MsgBox "Call to RFC_READ_TABLE failed!"
  End If
End Sub
```

The rest

Main Program

The above code have been the most thrilling part but then we need to put them together in a Main procedure.

Main() procedure

```
Sub Main()
```

The number at the end of the object class name lets you specify the version number, so that you could have several versions of the same class registered in Windows registry simultaneously. If you do not specify the version there should be a default version registered in the registry, if the developer made a proper effort to do so.

Create an instance of the SAP.LogonControl class (version 1)

```
  Set LogonControl = CreateObject("SAP.LogonControl.1")
```

Create an instance of the SAP.Functions collection

```
  Set funcControl = CreateObject("SAP.Functions")
```

The SAP table factory is an object that returns a special variant of a Visual Basic recordset along with appropriate methods to manipulate the table factory recordset.

Create an instance of the SAP.TableFactory

```
  Set TableFactoryCtrl = CreateObject("SAP.TableFactory.1")
```

Call the logon part

```
  call R3Logon
```

Assign the connection to our collection

```
  funcControl.Connection = conn
```

Make the RFC call

```
  call R3RFC_READ_TABLE("T000")
```

Log off the connection

```
  conn.Logoff
  MsgBox " Logged off from R/3! "
End Sub
```

Start the Program

Call Main()

Depending on the runtime environment you use, there are different ways to call the whole procedure. If you stored the coding as Visual Basic Script in a separate file with extension .VBS you have to add an explicit call to your main routine in the file.

```
Call Main()
```

ASP

Call Main() from an ASP page

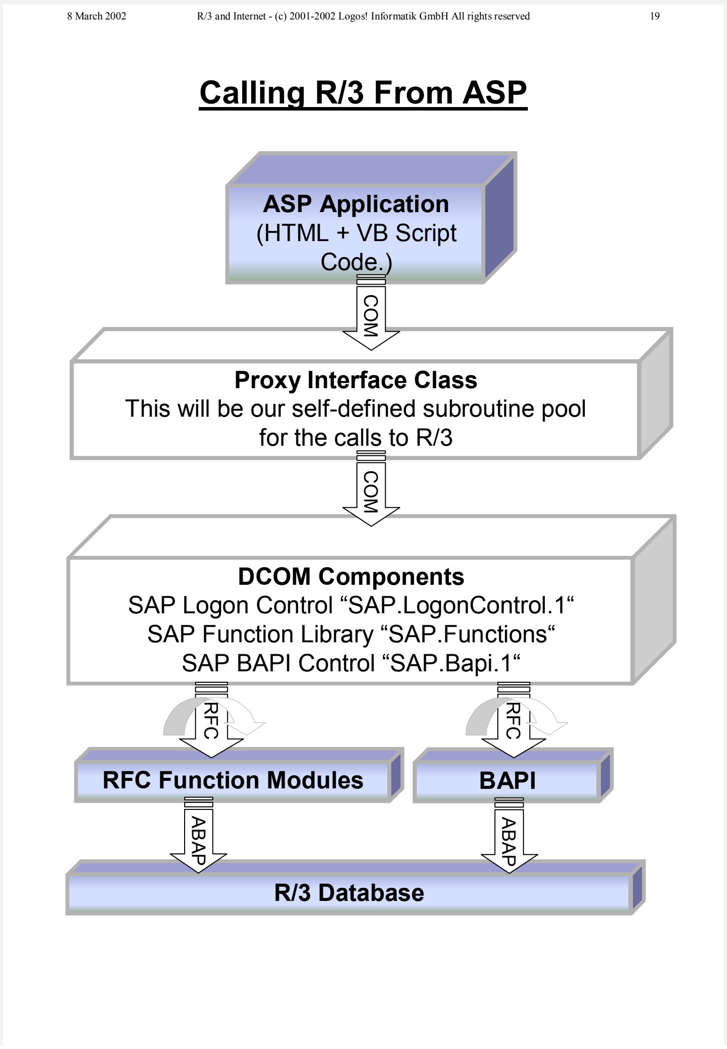
You can call that from an ASP page that would simply look similar to the following. (assume that the coding above is stored to a file called RfcReadTable.VBS.

```
<HTML>
<HEAD>
<#include RfcReadTable.vbs >
</HEAD>
<BODY><%>Call Main()<%></BODY>
```

Listing 1: The full coding to call an RFC function RFC_READ_TABLE

```
'*************************************************************
' Declarations
'*************************************************************
DIM LogonControl 'As SAPLogonCtrl.SAPLogonControl
DIM conn 'As SAPLogonCtrl.Connection
DIM funcControl 'As SAPFunctionsOCX.SAPFunctions
DIM TableFactoryCtrl 'As SAPTableFactoryCtrl.SAPTableFactory
'-------------------------------------------------------------
' Pointer to functions
'-------------------------------------------------------------
DIM RFC_READ_TABLE
'-------------------------------------------------------------
' Pointers to function parameters
'-------------------------------------------------------------
DIM eQUERY_TAB
DIM TOPTIONS
DIM TDATA
DIM TFIELDS

'*************************************************************
' Main Program
'*************************************************************
call Main

'*************************************************************
' Subroutines
'*************************************************************

Sub Main()
Set LogonControl = CreateObject("SAP.LogonControl.1")
Set funcControl = CreateObject("SAP.Functions")
Set TableFactoryCtrl = CreateObject("SAP.TableFactory.1")
call R3Logon
funcControl.Connection = conn
call R3RFC_READ_TABLE("T000")
conn.Logoff
MsgBox " Logged off from R/3! "
End Sub

Sub R3Logon()
Set conn = LogonControl.NewConnection
'-------------------------------------------------------------
' ** Set here your system data. They are also found in the R/3 Logon Panel
' Only the app server and the system number is mandatory. If the other params
' are missing you will be prompted for
'-------------------------------------------------------------
conn.ApplicationServer = "r3dev" ' IP or DNS-Name of the R/3 application server
conn.System = "00"                ' System ID of the instance, usually 00
conn.Client = "100"               ' opt. Client number to logon to
conn.Language = "EN"              ' opt. Your login language
conn.User = ""                    ' opt. Your user id
conn.Password = ""                ' opt. Your password

retcd = conn.Logon(0, False)
If retcd <> True Then
    MsgBox " Cannot log on! "
    MsgBox retcd
    Stop
else
    MsgBox " Logon OK."
End If
End Sub

Sub R3RFC_READ_TABLE(pQueryTab)
'-------------------------------------------------------------
' Add the R/3 RFC function RFC_READ_TABLE to the collection
'-------------------------------------------------------------
Set RFC_READ_TABLE = funcControl.Add("RFC_READ_TABLE")

'-------------------------------------------------------------
' Create objects for each parameter
'-------------------------------------------------------------
Set eQUERY_TAB = RFC_READ_TABLE.Exports("QUERY_TABLE")
Set TOPTIONS   = RFC_READ_TABLE.Tables("OPTIONS") '
Set TDATA       = RFC_READ_TABLE.Tables("DATA")  '
Set TFIELDS     = RFC_READ_TABLE.Tables("FIELDS")  '

eQUERY_TAB.Value = pQueryTab ' pQueryTab is the R/3 name of the table
TOPTIONS.AppendRow ' new item line
TOPTIONS(1,"TEXT") = "MANDT EQ '000'"

If RFC_READ_TABLE.Call = True Then
    If TDATA.RowCount > 0  Then
        MsgBox "Call to RFC_READ_TABLE successful! Data found"
        MsgBox TDATA(1, "WA")
    Else
        MsgBox "Call to RFC_READ_TABLE successful! No data found"
    End If
Else
    MsgBox "Call to RFC_READ_TABLE failed!"
End If
End Sub
```

Table 1: Principles components to call R/3 from ASP


8 March 2002 R/3 and Internet - (c) 2001-2002 Logos! Informatik GmbH All rights reserved 19

Calling R/3 From ASP

ASP Application
(HTML + VB Script Code.)
COM
Proxy Interface Class
This will be our self-defined subroutine pool for the calls to R/3
COM
DCOM Components
SAP Logon Control “SAP.LogonControl.1“
SAP Function Library “SAP.Functions“
SAP BAPI Control “SAP.Bapi.1“
RFC
RFC
RFC Function Modules
BAPI
ABAP
ABAP
R/3 Database

2 R/3 Java Connectivity

In order to connect from a Java class to R/3, SAP provides ihe R/3 Java Connector. It is a Java Class package that provides an easy and convenient interface to the RFC stubs from any Java implementation on Windows and Linux platforms.

R/3 Java Connectivity

The R/3 Java Connector is a Java Class package that provides easy and convenient access to the RFC stubs from any Java implementation on Windows and Linux platforms.

U:\Book\C04@01 JAVA.doc

R/3 Java Connectivity

The R/3 Java Connector is a Java Class package that provides easy and convenient access to the RFC stubs from any Java implementation on Windows and Linux platforms.

2.1 SAP R/3 Java Connector jCO,

The Java Connector is a class library for Java to allow flexible RFC access to R/3 from Java. It replaces the older Java RFC library jRFC.

jCO is the recommended access to R/3 from Java

The recommended access to R/3 via RFC from Java is the use of the Java Connector. The jCO is a class library, which has been developed by Thomas Schüssler from http://arasoft.de. It is actually an interface shim between Java and the standard RFC library. The jCO is available for NT platforms and for UNIX, AIX and LINUX.

The class library jCO.jar (com.sap.mw.jco.*) can be downloaded from the SAP mySAP.com marketplace or ordered directly from SAP by registered SAP R/3 licensees. The class is extremely well documented and we will therefore give only some small examples how to call an R/3 function module from Java.

Figure 1: Listing of classes of the Java Connector

```
jCO
jCO.AbapException
jCO.Attributes
jCO.BasicRepository
jCO.Client
jCO.Connection
jCO.ConversionException
jCO.Exception
jCO.Field
jCO.FieldIterator
jCO.Function
jCO.FunctionTemplate
jCO.MetaData
jCO.ParameterList
jCO.Pool
jCO.PoolChangedListener
jCO.PoolManager
jCO.Record
jCO.Repository
jCO.Server
jCO.ServerErrorListener
jCO.ServerExceptionListener
jCO.ServerStateChangedListener
jCO.ServerThread
jCO.Structure
jCO.Table
jCO.Throughput
jCO.TraceListener
```

3 Calling Remote Programs From R/3 Via RFC

The SAP RFC technology is an interface technology that allows external programs to call dedicated library function in R/3 and to call non R/3 programs from ABAP. To assist you in using the (CPIC-based) RFC protocol, SAP provides a range of libraries for the platforms Windows, UNIX and AS/400 which should be used as the exclusive gateway to R/3.

Calling Remote Programs From R/3 Via RFC

The SAP RFC technology is an interface technology that allows external programs to call dedicated library function in R/3 and to call non R/3 programs from ABAP. To assist you in using the (CPIC-based) RFC protocol, SAP provides a range of libraries for the platforms Windows, UNIX and AS/400 which should be used as the exclusive gateway to R/3.

U:\Book\C05@01 RFC - Calling External Programs.doc

Calling Remote Programs From R/3 Via RFC

The SAP RFC technology is an interface technology that allows external programs to call dedicated library function in R/3 and to call non R/3 programs from ABAP. To assist you in using the (CPIC-based) RFC protocol, SAP provides a range of libraries for t

3.1 What Is RFC?

The RFC protocol is a convention between the calling client and the called RFC server program. The protocol is based on the IBM CPIC protocol. SAP provides RFC server and RFC client interface libraries for most platforms, to allow proper communication.

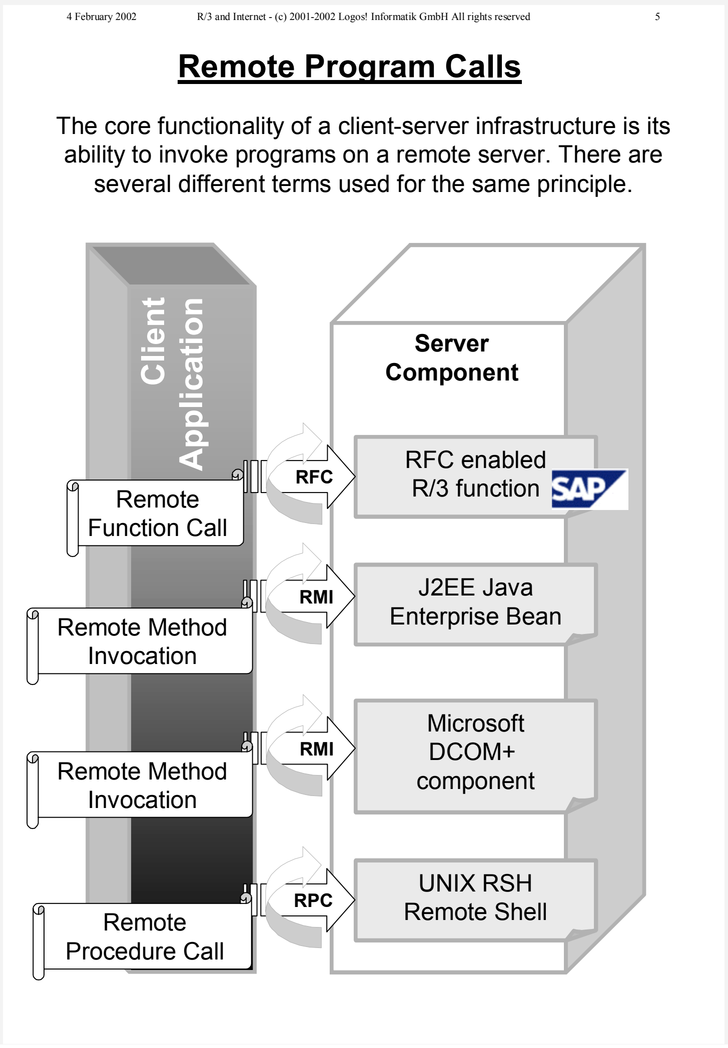
The RFC Remote Function Call protocol is SAP R/3's implementation of a distributed protocol, that allows to

- call R/3 functions in another R/3 system from one R/3 instance
- call R/3 function from a non-R/3 program, e.g. Java, C++ or Visual Basic
- call non-R/3 program from within R/3

3.2 Remote Program Call As Core of Client Server

The core functionality of a client-server infrastructure is its ability to invoke programs on a remote server. There are several different terms used for the same principle.

Figure 2: Remote Program are the core functionality of client server landscapes



4 February 2002 R/3 and Internet - (c) 2001-2002 Logos! Informatik GmbH All rights reserved 5

Remote Program Calls

The core functionality of a client-server infrastructure is its ability to invoke programs on a remote server. There are several different terms used for the same principle.

3.3 Setting Up The RFC Destinations

You have to set up the RFC destinations in your R/3 system before you can use them.

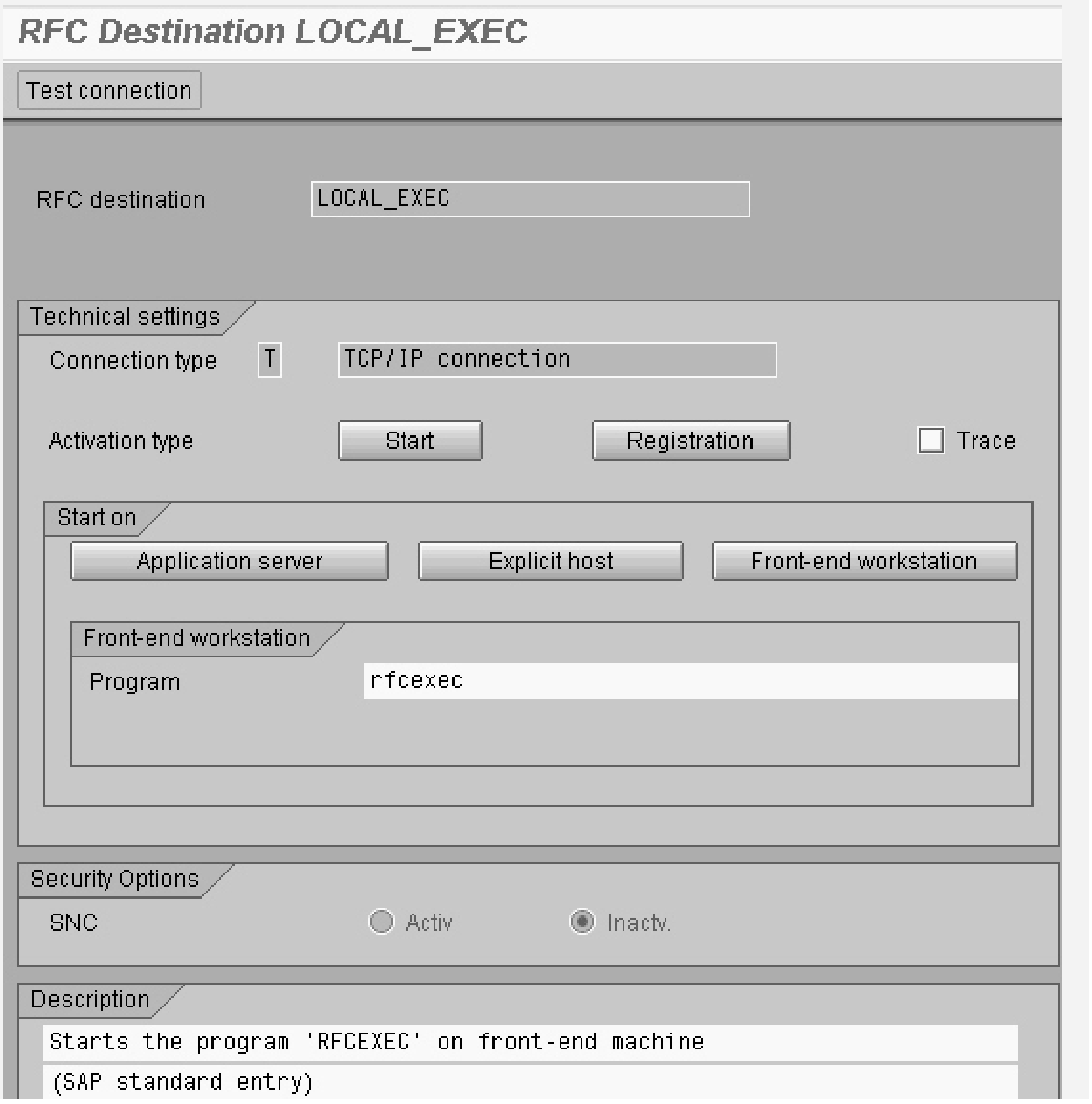
Every R/3 system is automatically an RFC server

Every R/3 system that can be reached from the calling R/3 instance is automatically a potential R/3 server. No set-up is therefore required on the side of the called R/3 server. .

The destination must be declared with SM59

Transaction SM59 is used to define new R/3 remote destinations. Here you specify the access parameters for the remote destination.

Table 2: Example setting of an R/3 remote destination with SM59



RFC Destination LOCAL_EXEC

Test connection

RFC destination: LOCAL_EXEC

Technical settings

Connection type: T — TCP/IP connection

Activation type: Start | Registration | [ ] Trace

Start on: Application server | Explicit host | Front-end workstation

Front-end workstation

Program: rfcexec

Security Options

SNC: ( ) Activ [✓ Inactv.]

Description

Starts the program 'RFCEXEC' on front-end machine
(SAP standard entry)

3.4 Calling a Program on a Workstation With RFC via SAPGUI

rfcexec.exe is accessible via a predefined RFC destination LOCAL_EXEC

In a plain installation there already exists the destination LOCAL_EXEC that points to rfcexec.exe and is set up as a frontend application. If you make a call to one of the rfcexec.exe methods to the destination LOCAL_EXEC, the program will be called on the workstation of the user, who calls the program. It should be evident, that this works only if the calling program is executed in online mode.

Listing 2: Execute a program on the workstation from R/3 with RFC_REMOTE_EXEC

```
DATA: command(256) DEFAULT 'echo Hello World. >> rfctest.dat'
DATA: rfc_mess(128).
CALL FUNCTION 'RFC_REMOTE_EXEC'
     DESTINATION rfcdest
     EXPORTING
       command = command
     EXCEPTIONS
       system_failure        = 1 MESSAGE rfc_mess
       communication_failure = 2 MESSAGE rfc_mess.
```

Listing 3: Execute a program on the workstation from R/3 with RFC_REMOTE_PIPE

```
DATA: command(256) DEFAULT 'echo Hello World. >> rfctest.dat'
DATA: rfc_mess(128).
DATA: pipedata(80) occurs 0 with header line.
CALL FUNCTION 'RFC_REMOTE_PIPE'
     DESTINATION rfcdest
     EXPORTING
       command = command
       read    = 'X'
     TABLES
       pipedata = pipedata
     EXCEPTIONS
       system_failure        = 1 MESSAGE rfc_mess
       communication_failure = 2 MESSAGE rfc_mess.
```

Listing 4: Read a file on a workstation from R/3 with RFC_REMOTE_FILE

```
DATA: command(256) DEFAULT 'echo Hello World. >> rfctest.dat'
DATA: rfc_mess(128).
DATA: pipedata(80) occurs 0 with header line.
CALL FUNCTION 'RFC_REMOTE_FILE'
     DESTINATION rfcdest
     EXPORTING
       file  = filename
       write = write   “space: read the file; 'X': save filedata to filename
     TABLES
       filedata = filedata
     EXCEPTIONS
       system_failure        = 1 MESSAGE rfc_mess
       communication_failure = 2 MESSAGE rfc_mess.
```

3.5 RFC via Remote Shell

If you want to execute a program on a remote computer, i.e. a computer that is neither the workstation nor the R/3 application server, the remote computer must be enabled to acccept remote program calls.

UNIX natively supports remote program calls RPC

On a UNIX installation, the RPC facility is usually installed by default. The program used for it is called RSH or RSHELL. This program runs on the SAP R/3 application server and calls the desired program on the remote destination. Usually this will be again rfcexec.exe, but now run on the remote computer. So to say it in brief:

- RSH runs on the application server
- 185 rfcexec.exe runs on the remote computer

For NT you need to install a remote shell utility like the ATAMAN manager

On NT you need to install a remote shell service. A standard Windows add-on as a remote shell host is the ATAMAN manger by ATAMAN Inc. The ATAMAN manager emulates the most common remote services of a UNIX installation on a Windows NT machine.

For NT you need to install a remote shell utility like the ATAMAN manager

Once you have correctly installed you should test everything from the UNIX command line, without SAP R/3. If, and only if this works fine, you continue to call the RSH from R/3 using the destination SERVER_EXEC. This destination is setup to call rfcexec.exe on the application server.

3.6 RFC via Web Server

Calling a remote program via a web server is an economic way to add remote calling security.

Security through Gateway Proxies

Using RFC_EXEC is very flexible because it principally allows to run any arbitrary program on any computer within the reach of calling computer. However, it is often not desired that the SAP R/3 application server should have generous access rights on the remote computer. IN that case the calls would usually be re-routed through a gateway interface or proxy. A proxy is a piece of software that operates as a broker between a client and a server. It receives a request message from the calling client, decides whether the call is permitted and then executes the program call. If a result is returned by the application it is first received by the proxy and then forwarded to the client.

Security through Gateway Proxies

SAP provides its own gateway services, and so do all the operating systems. As it is time for heading towards a standard, it appears to be wise to define a web server as the central gateway for rerouting the communication in your networked.

HTTP_GET

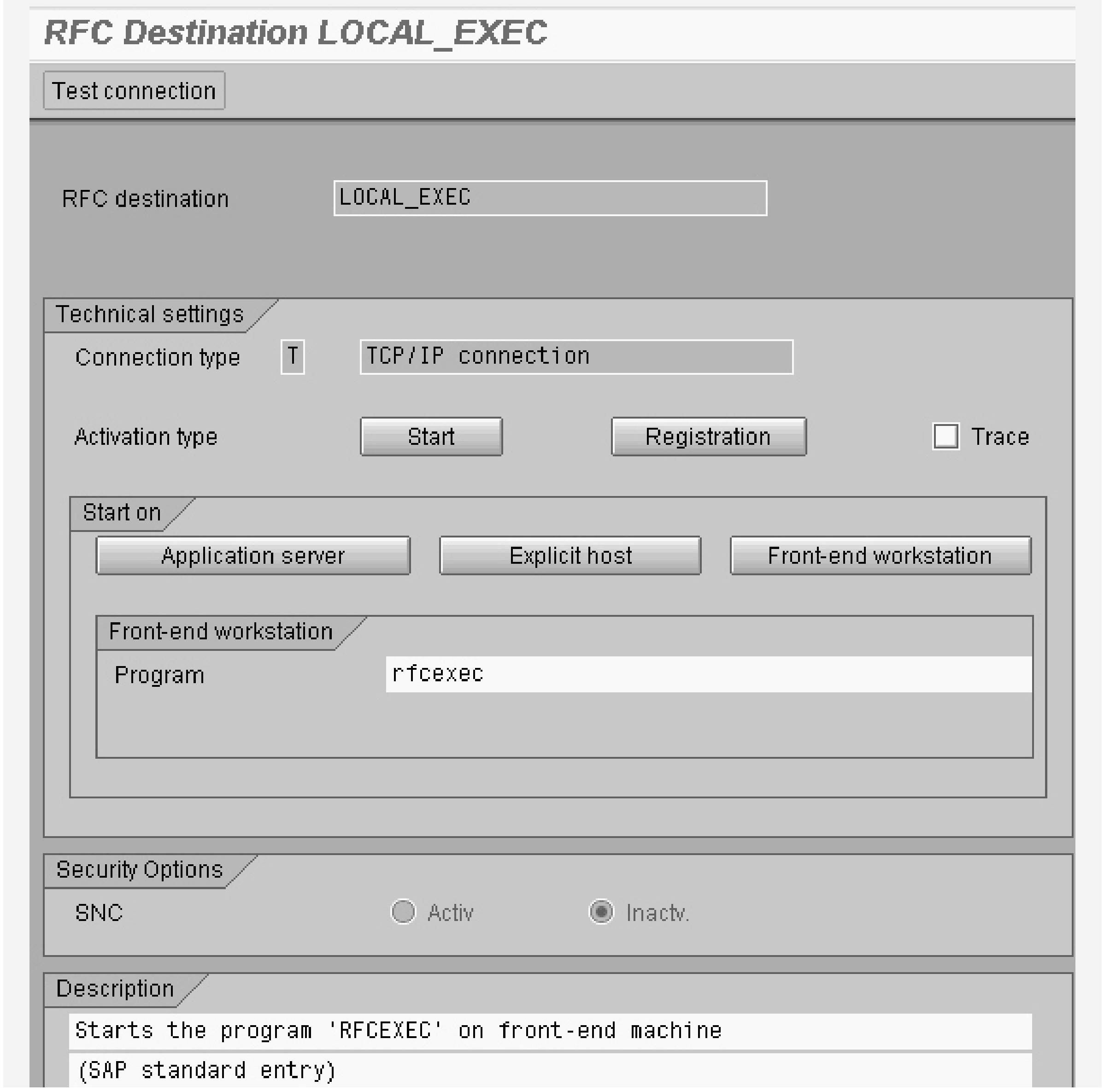
In order to allow the call of the program, one would install a web server on the remote computer. On the web server a scripting page (ASP, JSP, CGI etc) is installed. When the script page is requested via an HTTP URL, it executes the associated program and returns the result string. There exist utility programs called HTTP_GET in many varieties. These are command line executables, that request or post an HTTP request the same way as a browser does, however without any interactivity. HTTP_GET requests and URL puts the received HTTP data stream in its output pipe. A version of HTTP_GET is installed with every SAP application server and can be called through rfcexec and destination SERVER_EXEC.

3.7 RFC Via Operating System Services

The most simple and flexible way to start a remote program execution from R/3 is to invoke it from the underlying operating system. Basically this means to call the operation system shell or the operating system API from R/3 and pass the requested program command string or message to it.

In UNIX the operating system shell is call via the UNIX command ENV or also via the remote shell RSH (or RMSHELL). In Windows you can call the DOS command interpreter COMMAND.COM (or CMD in NT) or make a call to the Windows API. Practically you call a program through rfcexec on the application server, and this program does the job for you.

Figure 3: Settings of the predefined RFC destination LOCAL_EXEC in SM59



RFC Destination LOCAL_EXEC

Test connection

RFC destination: LOCAL_EXEC

Technical settings

Connection type: T TCP/IP connection

Activation type: Start Registration [ ] Trace

Start on: Application server / Explicit host / Front-end workstation

Front-end workstation

Program: rfcexec

Security Options

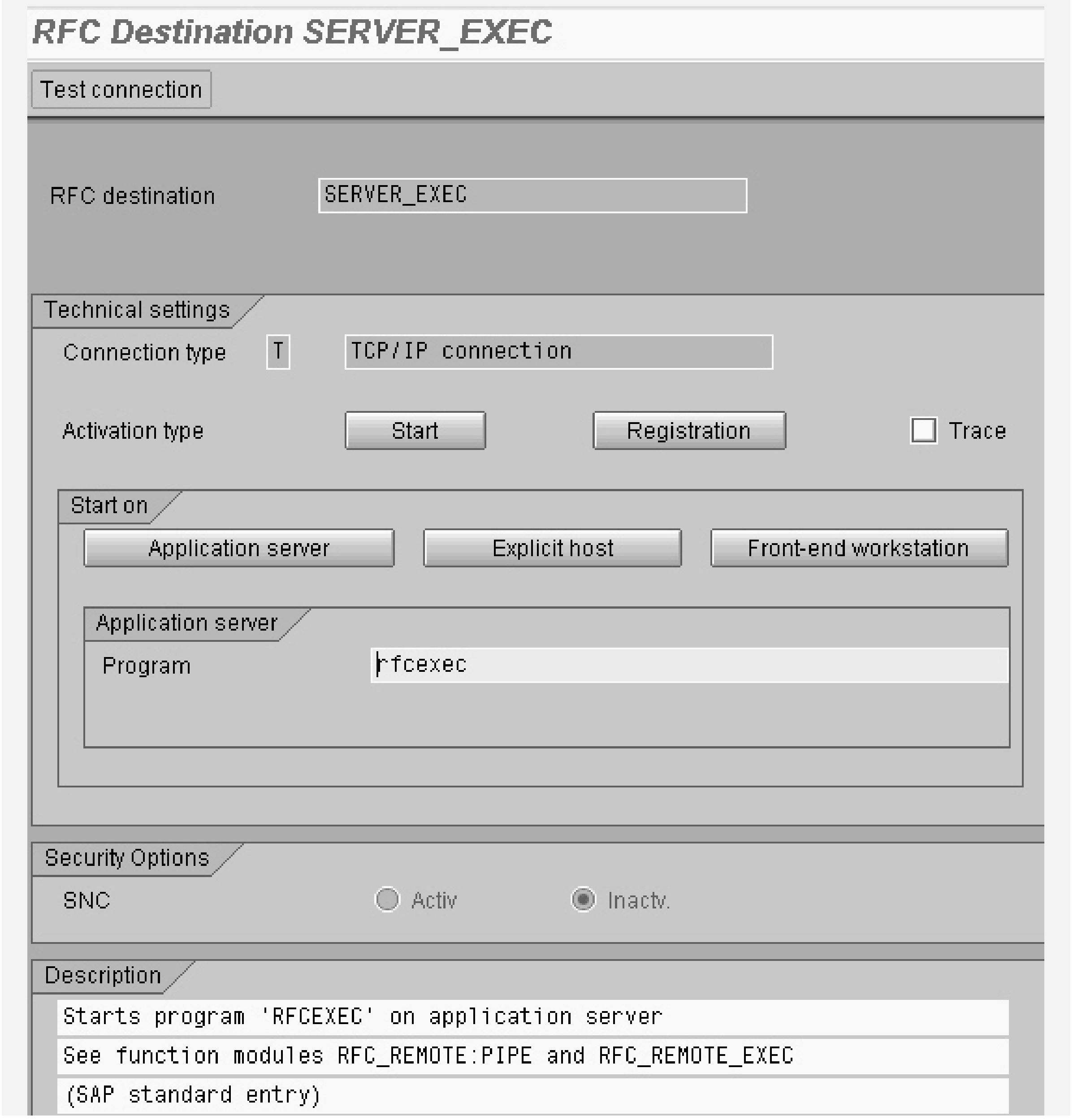
SNC: ○ Activ [● Inactv.]

Description

Starts the program 'RFCEXEC' on front-end machine

(SAP standard entry)

Figure 4: Settings of the predefined RFC destination LOCAL_EXEC in SM59



RFC Destination SERVER_EXEC

Test connection

RFC destination: SERVER_EXEC

Technical settings

Connection type: T — TCP/IP connection

Activation type: Start / Registration — ☐ Trace

Start on: Application server / Explicit host / Front-end workstation

Application server

Program: rfcexec

Security Options

SNC: Activ [✓ Inactv.]

Description

Starts program 'RFCEXEC' on application server
See function modules RFC_REMOTE:PIPE and RFC_REMOTE_EXEC
(SAP standard entry)

3.8 Calling A Program Remotely From R/3

R/3 allows to use RFC technique to call a program on a remote computer or on the workstation.

R/3 calls external program through an agent server like rfcexec.exe

R/3 uses a simple client-server technique to call a program on a remote destination where R/3 acts as the client. This requires that there exists a server object on the computer which is called by R/3. Depending on the server operating system and the location of the remote computer there are different techniques and server programs necessary.

Rfcexec.exe serves as RFC server for R/3

230

The called program must comply with the R/3 RFC protocol. As this is a proprietary protocol, there exists a program called rfcexec.exe, which acts as a broker or socket application and can in turn execute another program. This program is called by R/3 via an RFC destination which you have to define in transaction SM59. The program provides a number of predefined methods, which can be called with CALL FUNCTION DESTINATION from within R/3.

Figure 5: Predefined methods of the rfcexec.exe program

Method	Description

`RFC_REMOTE_PIPE`
`FUNCTION 'RFC_REMOTE_EXEC'`
`IMPORTING command`
`EXCEPTIONS`
`system_failure`
`communication_failure`
	Execute a program through rfcexec

`RFC_REMOTE_PIPE`
`FUNCTION 'RFC_REMOTE_PIPE'`
`IMPORTING command`
`TABLES pipedata(80)`
`EXCEPTIONS`
`system_failure`
`communication_failure`
	Execute a program and pass an input data through the input pipe and return the result pipe. The result pipe is the text stream which is usually output to a DOS box or UNIX command line by the program. E.g. if you execute dir > mydata.txt the current directory listing is written or “piped” to file mydata.txt. RFC_REMOTE_PIPE retrieves this pipe data instead of writing it to a file.
	file := name of the file at the remote site
	write := if write=’X’, filedata is written to file, otherwise the file is read an the result stored in filedata
	pipedata := content of the data passed to or retrieved from the called program

`RFC_REMOTE_FILE`
`FUNCTION 'RFC_REMOTE_FILE'`
`IMPORTING file(256)`
`Write(1)`
`TABLES filedata(80)`
`EXCEPTIONS`
`system_failure`
`communication_failure`
	Reads or writes a file at the RFC destination.
	file := name of the file at the remote site
	write := if write=’X’, filedata is written to file, otherwise the file is read an the result stored in filedata
	filedata := file content

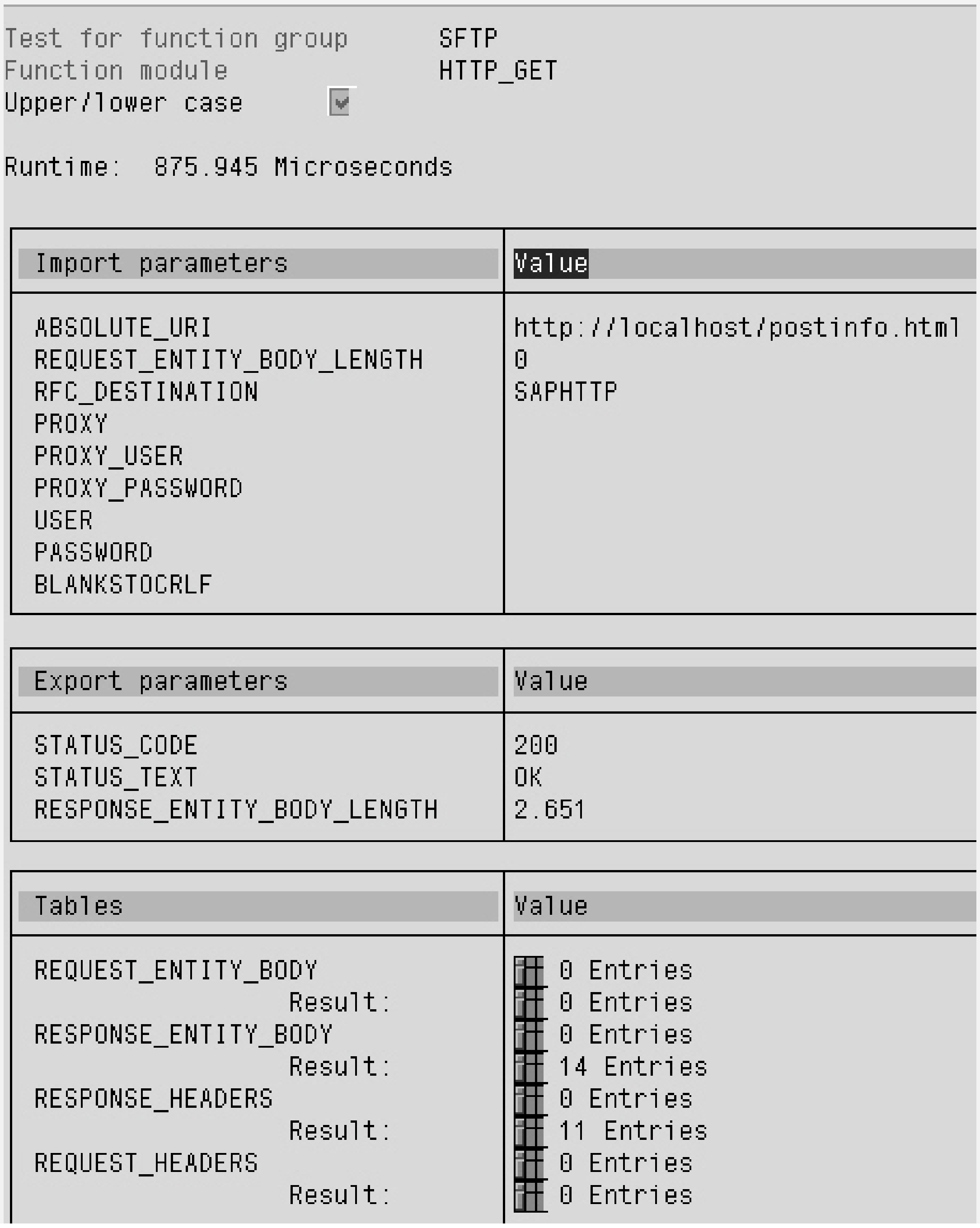
3.9 Calling An HTTP Web Server from R/3

R/3 comes with RFC-enabled HTTP_Get and HTTP_Post utilities for both the frontend and the application server that allows to call a web service via RFC.

Hiatus

R/3 allows to call an arbitrary server that runs on a TCP/IP port. In the case that the server is HTTP compliant (“a web server”)

Figure 6: A web service can be requested with HTTP_GET or HTTP_POST via RFC connection SAPHTTP



```
Test for function group      SFTP
Function module              HTTP_GET
Upper/lower case      ✓

Runtime:  875.945 Microseconds
```

Import parameters	Value
ABSOLUTE_URI	http://localhost/postinfo.html
REQUEST_ENTITY_BODY_LENGTH	0
RFC_DESTINATION	SAPHTTP
PROXY	
PROXY_USER	
PROXY_PASSWORD	
USER	
PASSWORD	
BLANKSTOCRLF	

Export parameters	Value
STATUS_CODE	200
STATUS_TEXT	OK
RESPONSE_ENTITY_BODY_LENGTH	2.651

Tables	Value
REQUEST_ENTITY_BODY	0 Entries
Result:	0 Entries
RESPONSE_ENTITY_BODY	0 Entries
Result:	14 Entries
RESPONSE_HEADERS	0 Entries
Result:	11 Entries
REQUEST_HEADERS	0 Entries
Result:	0 Entries

Listing 5: Result of retrieving an URI with HTTP_GET

```
Test for function group      SFTP
Function module              HTTP_GET
Upper/lower case

  Import parameters               Value
  ABSOLUTE_URI                    http://localhost/postinfo.html
  REQUEST_ENTITY_BODY_LENGTH      0
  RFC_DESTINATION                 SAPHTTP
  PROXY
  PROXY_USER
  PROXY_PASSWORD
  USER
  PASSWORD
  BLANKSTOCRLF

  Export parameters               Value
  STATUS_CODE                     200
  STATUS_TEXT                     OK
  RESPONSE_ENTITY_BODY_LENGTH     2.651

  Tables                          Value
  REQUEST_ENTITY_BODY                0 Entries
                   Result:           0 Entries
  RESPONSE_ENTITY_BODY               0 Entries
                   Result:           14 Entries
  RESPONSE_HEADERS                   0 Entries
                   Result:           11 Entries
  REQUEST_HEADERS                    0 Entries
                   Result:           0 Entries
```

Listing 6: The header of the retrieved URI response is reported in parameter RESPONSE_HEADERS

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Cache-Control: no-cache
Expires: Sat, 08 Sep 2001 12:36:33 GMT
Date: Sat, 08 Sep 2001 12:36:33 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Sun, 21 Jan 2001 14:24:08 GMT
ETag: "0b4b1d0b583c01:96a"
Content-Length: 2651
 <blank line>
```

3.10 Registering an RFC Listener With The SAP Gateway

SAP R/3 RFC allows a passive mode operation, where the external (non-R/3) RFC server program registers itself to the SAP Gateway. This allows the RFC server to act as a demon, which remains idle in memory until its services are requested by an RFC function call.

Example registering an RFC listener to the gateway

```
Srfcserv -aMYHOST.srfcserv -gSAPR3.logosworld.com -xSAPGW00 -t
```

The given example registers itself to the SAPGW00 at the SAP R/3 instance SAPR3 at the domain logosworld.com.

SAPGWxx where xx is the instance number

The SAPGW00 is the gateway for the database instance 00, if you had an instance 01 the gateway would automatically be numbered SAPGW01. The SAPR3.logosworld.com is the name of the application server of your R/3 instance.

Parameters can also be stored in saprfc.ini

Instead of specifying all the parameters in the command line, you can store them in the saprfc.ini file. The saprfc.ini must be in the same directory than the RFC server program.

```
Srfcserv -DRFCEXT_LISTENER
```

Listing 7: Contents of the SAPRFC.INI

```
/*===================================================================*/
/*  Type R:  Register a RFC server program at a SAP gateway          */
/*           or connect to an already registered RFC server program  */
/*===================================================================*/
DEST=RFCEXT_LISTENER
TYPE=R
PROGID=SAPR3.srfcserv
GWHOST=SAPr3.logosworld.com
GWSERV=sapgw00
RFC_TRACE=0
```

SMGW displays registered services

250 After an RFC listener is registered successfully with the gateway, the registered service is visible with transaction SMGW under the menu entry “Logged on systems”.

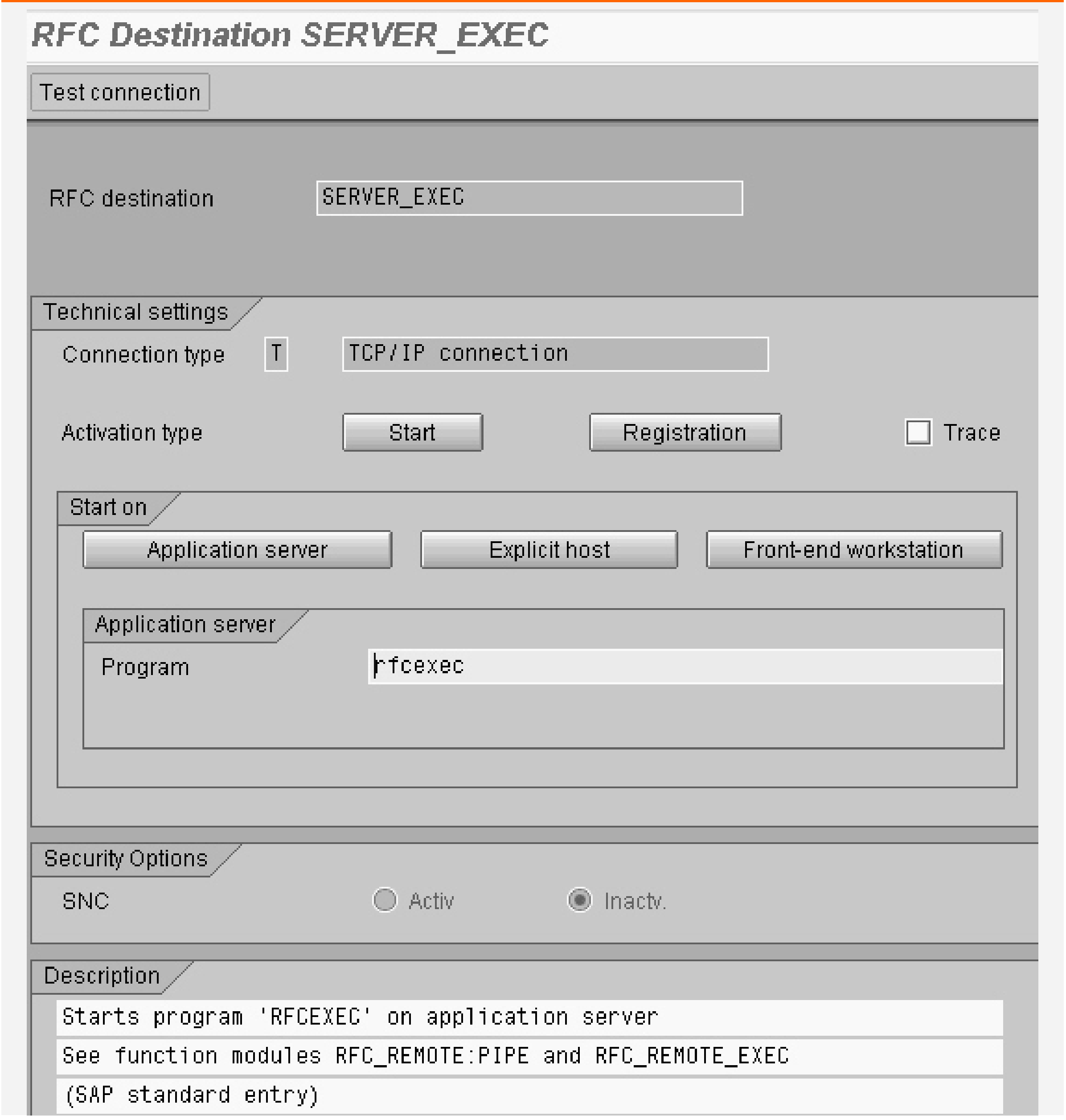
Listing 8: Options to register an RFC Server DLL to the SAP Gateway

```
C:\Programme\sapgui46b\SAPGUI\RFCSDK\bin>srfcserv
Syntax for start and run in register mode:
  srfcserv [options]
  with
  options = -D<destination with type 'R' in saprfc.ini>
          = -t             RFC-Trace on
  or
  options = -a<program ID> e.g.  <own host name>.srfcserv
          = -g<SAP gateway host name>        e.g.  hs0311
          = -x<SAP gateway service>          e.g. sapgw53
          = -t             RFC-Trace on
          = -L<SNC library, optional>
          = -S<SNC myname, optional>
          = -Q<SNC quality of protection, optional>

 Option L, S and Q can be set if working with SNC
 (Secure Network Communication).
```

3.11 Create an RFC Destination For A Registered Listener

In order to access a registered service on the SAP Gateway, you have to create an RFC destination for an TCP/IP connection in transaction SM59 with the start option: "Registration". You would also have to specify the name of the gateway, where the service is registered.

Table 3: Settings of an RFC destination for a registered service in SM59



RFC destination: SERVER_EXEC

Connection type: T TCP/IP connection

Activation type: Start / Registration ☐ Trace

Program: rfcexec

SNC: ○ Activ ● Inactv.

```
Starts program 'RFCEXEC' on application server
See function modules RFC_REMOTE:PIPE and RFC_REMOTE_EXEC
(SAP standard entry)
```

3.12 Writing Your Own RFC Listener

A detailed example how to write an RFC listener in C is shown inn detail in the SAP RFCSDK, which is part of the (full) SAPGUI installation. There is the full source code for the standard RFC server `rfcexec` and `srfcserv`. There is also detailed information in the `saprfc.hlp` file. 255

Listing 9: Extracts from the implementation of the rfcexec server

```
static RFC_RC DLL_CALL_BACK_FUNCTION _loadds remote_pipe( RFC_HANDLE handle );
static RFC_RC DLL_CALL_BACK_FUNCTION _loadds remote_file( RFC_HANDLE handle );
static RFC_RC DLL_CALL_BACK_FUNCTION _loadds remote_exec( RFC_HANDLE handle );
static RFC_RC DLL_CALL_BACK_FUNCTION _loadds mail       ( RFC_HANDLE handle );

/*
 *  main function for an RFC server program
 */

/*ARGSUSED*/
main( int argc, char ** argv )
{
   /* initialized data */
   static RFC_ENV    env;

   RFC_HANDLE handle;
   RFC_RC     rc;

   if (argc == 1)
   {
     help();
     return 0;
   }

   /*
    * install error handler
    */
   env.errorhandler = myErrorhandler;
   RfcEnvironment( &env );

   /*
    * accept connection
    * (command line argv must be passed to RfcAccept)
    */
   handle = RfcAccept( argv );

   /*
    * static function to install offered function modules
    */
   rc = install(handle);
   if( rc != RFC_OK )
   {
     /*
      * if error occured,
      * close connection with error message and exit
      */
     RfcAbort( handle, "Initialization error" );
     exit(1);
   }

static RFC_RC DLL_CALL_BACK_FUNCTION _loadds remote_pipe(  RFC_HANDLE handle )
{
    char          command[256];
    RFC_PARAMETER parameter[4];
    RFC_TABLE     table[2];
    RFC_RC        rc;
    RFC_CHAR      read_flag = 0;
    int           mode;

    memset( command, 0, sizeof( command ) );

    parameter[0].name = "COMMAND";
    parameter[0].nlen = 7;
    parameter[0].addr = (void *) command;
    parameter[0].leng = sizeof(command);
    parameter[0].type = RFCTYPE_CHAR;
    parameter[1].name = "READ";
    parameter[1].nlen = 4;
    parameter[1].addr = (void *) &read_flag;
    parameter[1].leng = sizeof(read_flag);
    parameter[1].type = RFCTYPE_CHAR;
    parameter[2].name = NULL;

    table[0].name = "PIPEDATA";
    table[0].nlen = 8;
    table[0].type = RFCTYPE_CHAR;
    table[0].leng = table_size;
    table[0].itmode = RFC_ITMODE_BYREFERENCE;

    table[1].name = NULL;

    rc = RfcGetData( handle, parameter, table );
    if( rc != RFC_OK ) return rc;

#ifdef SAPonWINDOWS
    RfcAbort(handle, "Function RFC_REMOTE_PIPE is not supported on Windows");
    exit(1);
#endif

    if( read_flag != 'X' )
      mode = RUN_WAIT;
    else
      mode = RUN_READ;

    rc = run( handle, command, sizeof(command),
                 table[0].ithandle, mode, (RFC_INT *)0 );

    if( rc != RFC_OK ) return rc;

    parameter[0].name = NULL;
    rc = RfcSendData( handle, parameter, table );

    return rc;
} /* remote_pipe */
```

3.13 Troubleshooting RFC Access

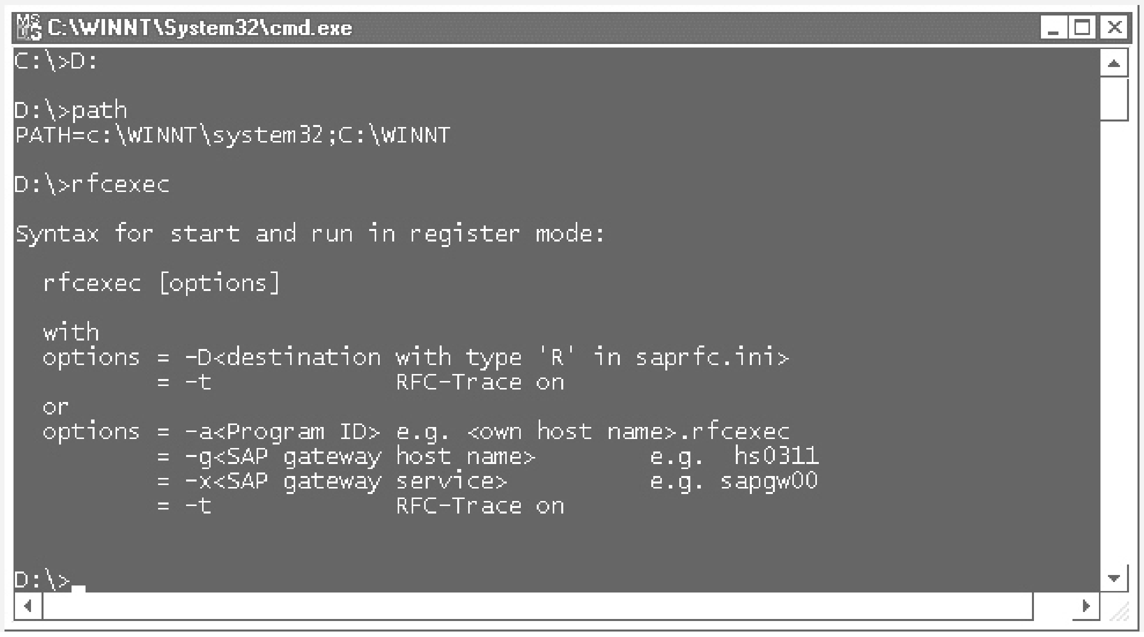
ABAP SRFCTEST allows to test the RFC features

There exists a standard ABAP in R/3 SRFCTEST that demonstrates the use of most important outbound RFC features and allows to test them.

If your connections fail, try first to test without R/3

If your connections fail, try first to test them without R/3. E.g. if the RFC call does not execute, you should check if the program rfcexec.exe is installed and can be called from everywhere. If rfcexec is to executed on the frontened, sit at the machine and open a DOS box. Go to an arbitrary directory (not the one where rfcexec.exe is stored) and call rfcexec from the command line. If it fails, the program is not in the search path of your operating system. Copy the program 265 rfcexec.exe to a directory within the search path of Windows. The current search paths are displayed with the DOS PATH command.

Figure 7: Calling rfcexec.exe from DOS should display a result similar to this



C:\WINNT\System32\cmd.exe

```
C:\>D:

D:\>path
PATH=c:\WINNT\system32;C:\WINNT

D:\>rfcexec

Syntax for start and run in register mode:

  rfcexec [options]

  with
  options = -D<destination with type 'R' in saprfc.ini>
          = -t             RFC-Trace on
  or
  options = -a<Program ID> e.g. <own host name>.rfcexec
          = -g<SAP gateway host name>        e.g.  hs0311
          = -x<SAP gateway service>          e.g. sapgw00
          = -t             RFC-Trace on


D:\>_
```

If this works fine, then probably have a problem with the definition of the RFC destination in transaction SM59. Provided that the settings in SM59 are correct, there are several possibilities why the connection may fail. However, the principle remains always the same:

First make the program run locally and then try to call it remotely.

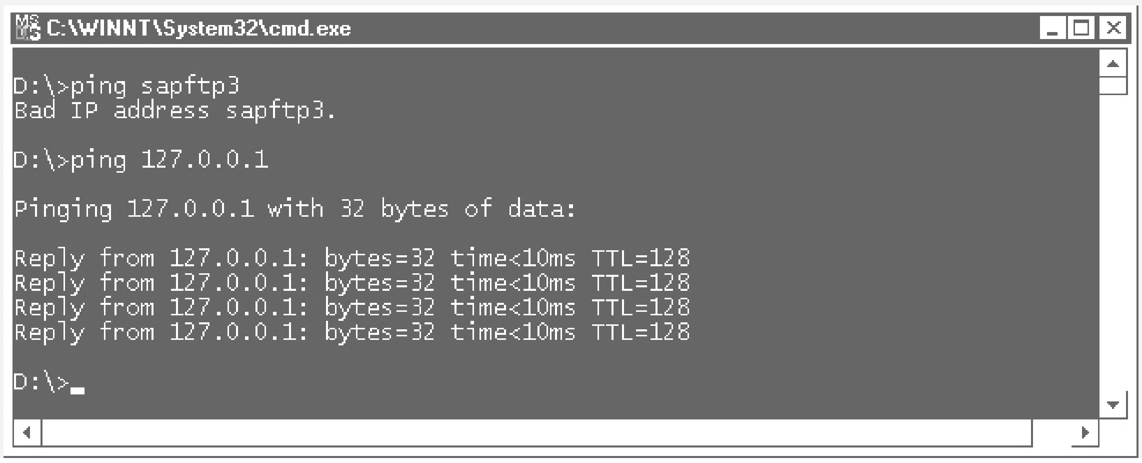
Make sure you have a well installed SAPGUI

If you connect through the SAPGUI front-end, a frequent cause is a corrupted or incomplete SAPGUI. Try to reinstall the SAPGUI and choose a full install to be sure that all elements are put on the PC and registered in Windows

Check IP connection with PING and TRACEROUTE

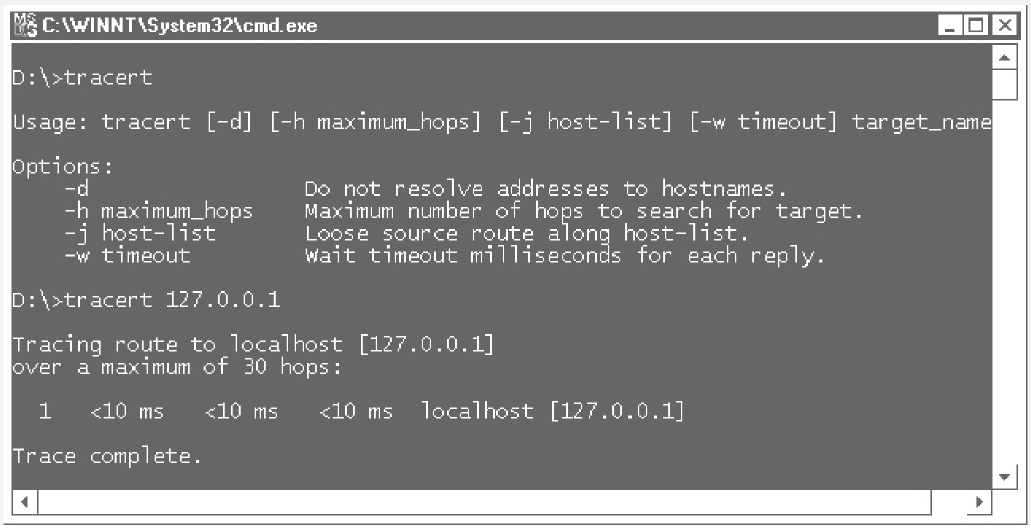
If you connect through TCP/IP, be sure that the SAP application server can contact the remote computer at all. To check this, log in with TELNET on your application server with TELNET as SAP administrator (usually SAPsid, where sid is the system ID of the instance) and use the standard IP utilities PING and TRACEROUTE (name may vary, e.g. in Windows it is called TRACERT). The traceroute utility list all the computers (“hops”) that an IP package visits during its journey to the destination address.

Figure 8: Successful and unsuccessful PING to a remote computer



```
D:\>ping sapftp3
Bad IP address sapftp3.

D:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:

Reply from 127.0.0.1: bytes=32 time<10ms TTL=128
Reply from 127.0.0.1: bytes=32 time<10ms TTL=128
Reply from 127.0.0.1: bytes=32 time<10ms TTL=128
Reply from 127.0.0.1: bytes=32 time<10ms TTL=128

D:\>_
```

Figure 9: Using TRACERT to find a remote IP host



```
D:\>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] target_name

Options:
    -d                  Do not resolve addresses to hostnames.
    -h maximum_hops     Maximum number of hops to search for target.
    -j host-list        Loose source route along host-list.
    -w timeout          Wait timeout milliseconds for each reply.

D:\>tracert 127.0.0.1

Tracing route to localhost [127.0.0.1]
over a maximum of 30 hops:

  1    <10 ms    <10 ms    <10 ms   localhost [127.0.0.1]

Trace complete.
```

285