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 analysed 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.

Visual Basic code to read data from table T000 via RFC

Declarations

Declare an R/3 Logon OCX component

```
DIM LogonControl
```

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

```
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 function module

```
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

```
Sub R3Logon()
```

Create a new connection with method NewConnection

```
  Set conn = LogonControl.NewConnection
```

Specifying the Login properties

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.

```
  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
```

Calling Logon method

```
  retcd = conn.Logon(0, False)
```

Checking if logon has been successful

```
  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.

```
Sub R3RFC_READ_TABLE(pQueryTab)
'------------------------------------------------------------
' Call the R/3 RFC function RFC_READ_TABLE
'------------------------------------------------------------
```

Create a new collection object for the function module

```
  Set RFC_READ_TABLE = funcControl.Add("RFC_READ_TABLE")
```

Set pointers to local variables for the import and export parameters

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.

```
  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")  '
```

Reading and writing parameter values

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

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

Calling the RFC function

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.

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

Output the result

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.

```
    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
```

The rest

```
End Sub
```

Main Program

Main() procedure

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

```
Sub Main()
```

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

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.

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

Create an instance of the SAP.Functions collection

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

Create an instance of the SAP.TableFactory

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.

```
  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`.

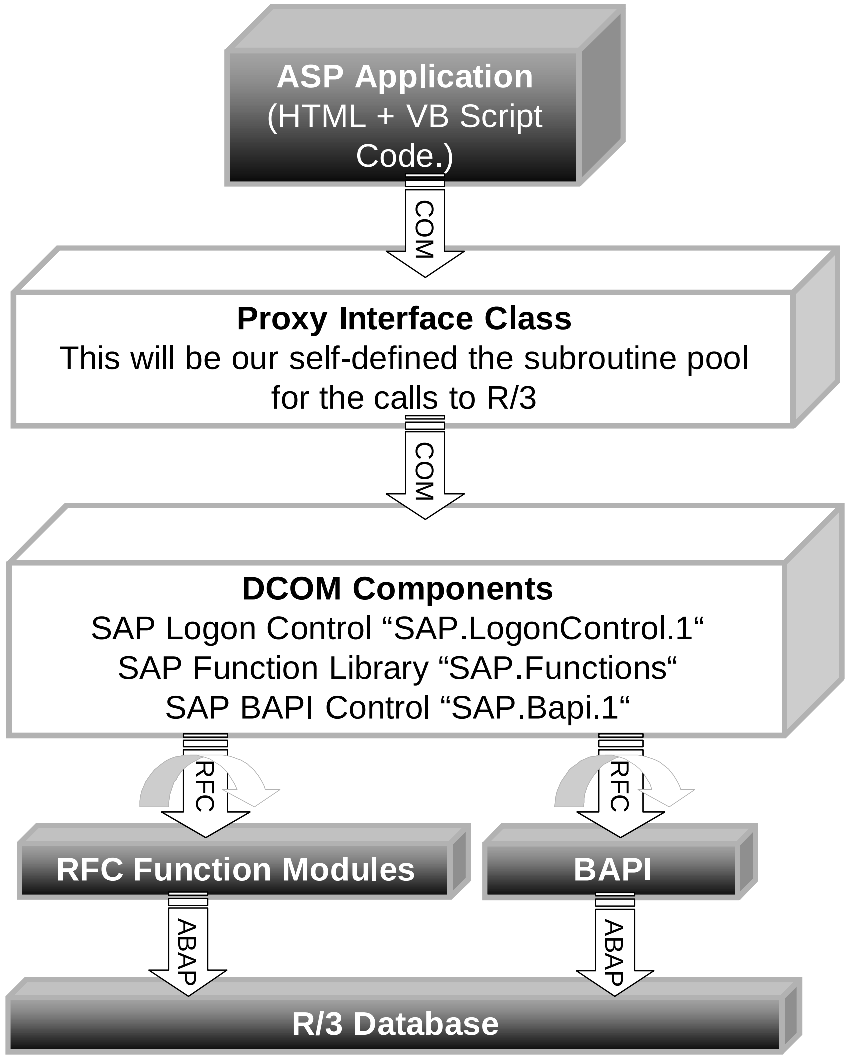
```
<HTML>
<HEAD>
<#include RfcReadTable.vbs >
</HEAD>
<BODY><%>Call Main()<%></BODY>
```

Example 1: The coding to call an RFC function RFC_READ_TABLE as a whole

```
'**************************************************************
' 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
```

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

Calling R/3 From ASP



ASP Application
(HTML + VB Script Code.)
COM
Proxy Interface Class
This will be our self-defined the 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