The BAPI Gateway Component (BC-FES-AIT)

HELP.BCFESBGW

Release 4.6C

SAP ™

Copyright

Icons

Icon	Meaning
⚠	Caution
💬	Example
💡	Note
🧭	Recommendation
SYN	Syntax

Contents

The BAPI Gateway Component (BC-FES-AIT)

Purpose

The *BAPI Gateway* component allows you to call BAPIs (which are methods of SAP business objects) and RFC function modules dynamically. This means that at design time you do not have to know which BAPIs or function modules you are going to call at run time.

Integration

Using the Repository Services to Obtain Metadata

Since you do not know which BAPIs or function modules you are going to call at run time, you also do not know the metadata of those BAPIs or function modules. For example, you do not know which BAPIs exist for a business object you wish to use, and for each of the BAPIs you may call in that business object you do not know what their parameters are, and what is the data type of each of the parameters. You can use the *Repository Services component* in conjunction with the *BAPI Gateway* to get this metadata at run time.

Architecture

Internally, the *BAPI Gateway* component uses the following related products:

- It uses services from the *SAP DCOM Connector* to handle connections to R/3 when such a connection is required. You need to have the *SAP DCOM Connector* installed, and use its Destination editor to define at least one destination system, before you can call BAPIs or function modules in that R/3 system.

- It uses the *SAP Automation DCOM Connector Logon Component* to get necessary logon information.

- It also uses the *Repository Services component* to get metadata information.

The BAPI Gateway Component (BC-FES-AIT)
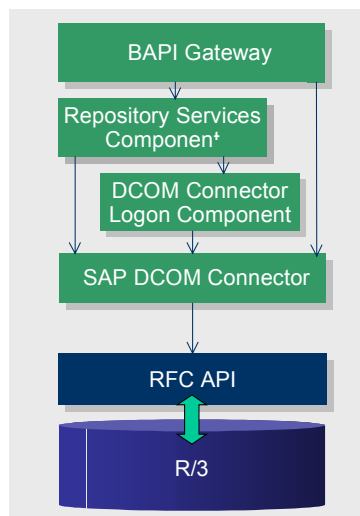
Implementation Considerations

The *BAPI Gateway* can be used by DCOM-compliant programs, such as those written in Visual Basic, Java, C++, and so on.

System Requirements

Development Requirements

To create applications using the *BAPI gateway component* you need the following:

- Windows NT (Service Pack 3) or Windows 95 operating system

- Windows NT Option Pack

- SAP R/3 release 3.1F or higher

- The Repository Services (which in turn requires the SAP DCOM Connector) must be available over the network or locally. You can get this from a SAP Automation installation.

- The *SAP Automation DCOM Connector Logon Component* needs to be installed locally, because it has to display the Logon dialog. You can get this from a SAP Automation installation.

Required Visual Basic Project References

To work with the *BAPI Gateway* in Visual Basic activate the following project references:

- Bapi Gateway 4.5B Type Library

- SAP Repository Services Type Library

- Microsoft ActiveX Data Objects recordset 2.0 Library

- SAP DCOM Connector Logon Component

Run-time Requirements

The end user of an application using the *BAPI gateway component* needs the following:

- Windows NT (Service Pack 3) or Windows 95 operating system

- Windows NT Option Pack

- SAP R/3 release 3.1F or higher

In addition, they need the following DLLs:

DLL	Description
Sapbapgw.dll	BAPI Gateway
Sapdlogn.dll	*SAP Automation DCOM Connector Logon Component*
Saprepsv.dll	*Repository Services component*
Saprsoff.dll	*Repository Services component (the offline conponent)*
Librfc32.dll	*SAP DCOM Connector Component*
Ccadmin.dll	*SAP DCOM Connector Component* (Destination editor)

System Requirements

The Sapdlogn.dll must be installed locally, if your program invokes the Logon dialog (using the Logon method of the *SAP Automation DCOM Connector Logon Component)*.

On the machine where the *SAP DCOM Connector* is installed, the etc\services file needs to be modified to include the sapdpXX and the sapgwXX entries. Use the following InstallShield script to add the necessary lines:

```
/     Setup the services file with the SAP gateway entries.

function DoSetupServices()
BOOL   bSetupGateway,  bSetupHostdp;
number nLineNumber, nFileHandle, i, nLen;
STRING szID[300], szServiceNum [200];
STRING szReturnLine[300];
STRING szDrive [300];
STRING szGateway [300];
STRING szHostdp [300];
STRING szStatusText [300];

STRING svResult;
number nResult;

begin
VarSave(SRCTARGETDIR);

// szStatusText =   "Checking SERVICES file " ;
SetStatusWindow(-1, "Checking SERVICES file");

// Either Windows-NT or Windows for WorkGroup
if bWinNT then
SRCDIR = WINDIR ^ "\\SYSTEM32\\DRIVERS\\ETC" ;
else
SRCDIR = WINDIR ;
endif;

FileGrep("SERVICES.", "sapgw00", szReturnLine, nLineNumber,
RESTART);
if (LAST_RESULT = FILE_NOT_FOUND) then
MessageBox("    Could not locate the SERVICES file. \n\nPlease
refer to README for more information.", INFORMATION) ;
VarRestore(SRCTARGETDIR);
return ;
endif ;

bSetupGateway = TRUE ;
if (LAST_RESULT = 0) then
bSetupGateway = FALSE ;
endif ;

bSetupHostdp = TRUE ;
FileGrep("SERVICES.", "sapdp00", szReturnLine, nLineNumber,
RESTART);
if (LAST_RESULT = 0) then
bSetupHostdp = FALSE ;
```

```
      endif ;

      if (!bSetupGateway) then
      if (!bSetupHostdp ) then
      VarRestore(SRCTARGETDIR);
      return ;
      endif ;
      endif ;

      OpenFileMode(FILE_MODE_APPEND);
      OpenFile(nFileHandle, SRCDIR, "SERVICES.");
      if (LAST_RESULT < 0) then
      MessageBox("    Could not locate SERVICES file. \n\nPlease refer to
      README for more information.", INFORMATION) ;
      VarRestore(SRCTARGETDIR);
      return;
      endif;

      szGateway = "sapgw00                    3300/tcp" ;
      szHostdp  = "sapdp00                    3200/tcp" ;
      for i = 0 to 99
      NumToStr(szID, i) ;
      if (StrLength( szID ) = 1) then
      szServiceNum = "0" + szID ;
      else
      szServiceNum = szID ;
      endif ;

      if (bSetupGateway) then
      CopyBytes(szGateway,  5, szServiceNum,  0, 2) ;
      CopyBytes(szGateway, 28, szServiceNum,  0, 2) ;
      WriteLine(nFileHandle, szGateway);
      endif ;

      if (bSetupHostdp) then
      CopyBytes(szHostdp,   5, szServiceNum,  0, 2) ;
      CopyBytes(szHostdp,   28, szServiceNum,  0, 2) ;
      WriteLine(nFileHandle, szHostdp );
      endif ;
      endfor ;

      VarRestore(SRCTARGETDIR);

      end ;
```

BAPI Gateway Objects

The following table describes the BAPI Gateway objects:

BAPI Gateway Object	Represents
Table	An SAP table
Structure	A single row of a table
IGWParameter	A parameter of either a BAPI or a function module
IGWMethod	A BAPI of an SAP business object
IGWKey	The key attribute of an SAP business object
BusinessObject	An SAP business object
Function	An RFC function module

Object Hierarchy

The following diagram shows the containment hierarchy between the various BAPI Gateway objects:
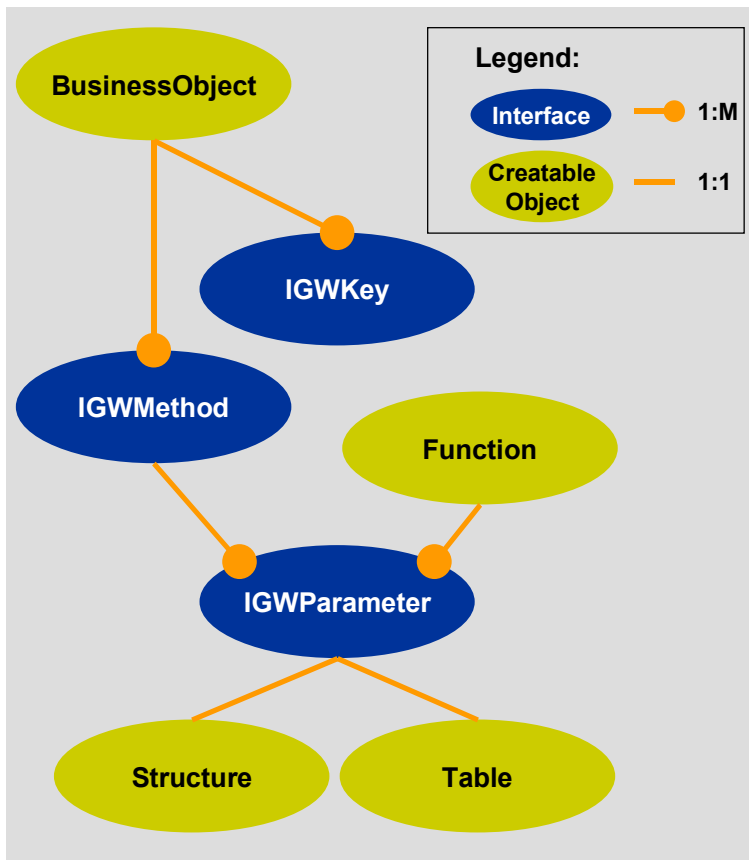
Relationship to Repository Services Objects

The following table lists the BAPI Gateway objects and their equivalent Repository Services objects.

BAPI Gateway Object	Repository Services Object
Table	ITable
Structure	IStructure
IGWParameter	IParameter
IGWMethod	IMethod
IGWKey	IField
BusinessObject	IBusinessObject
Function	IFunction

Using the BAPI Gateway

Process Flow

1. Let an end user specify which BAPI or RFC they wish to use, or define these in your program.

2. Set up the necessary connection properties in both the BAPI Gateway object and in the Repository Services object [Page 12]

3. Use the BAPI Gateway and the *Repository Services component* to prepare and call a BAPI [Page 16] or to prepare and call a function module [Page 19].

Setting Up the Necessary Connection properties

Use

SAP DCOM Connector Objects Need Connection Properties

The *BAPI Gateway* uses services from the *SAP DCOM Connector* to handle connections to R/3 when such a connection is required.

Objects using services from the *SAP DCOM Connector* require that a set of connection properties contain the information necessary for logging onto an R/3 system. Once you do so, you do not need to actively establish a connection to R/3. When you call a BAPI or a function module, which requires a connection to R/3, the *SAP DCOM Connector* establishes the connection to R/3 as needed, based on the connection properties you had set.

The *BAPI Gateway* has the following connection properties, which you may set directly to fulfill this requirement.

- Destination

- Client

- Language

- Password

- UserID

Every BAPI Gateway object also contains a PutSessionInfo method, with which you can specify the various logon parameters. Calling this method assigns these values into the various properties of your BAPI Gateway object.

Setting Up Connection Properties with the SAP Automation DCOM Connector Logon Component

However, an easier way to set the necessary Connection properties is by using the *SAP Automation DCOM Connector Logon Component.*

The *SAP Automation DCOM Connector Logon Component* provides a Logon dialog with which you can obtain the necessary logon parameters from an end user.

It also allows you to hold all the necessary logon parameters in the various properties of a Connection object.

You can then assign this Connection object to the Connection property of the BAPI Gateway object, which automatically sets all the necessary information into the various connection properties of that BAPI Gateway object.

Setting up the Repository Services Object

Since you use the *Repository Services component* with the *BAPI Gateway* to get metadata at run time, the Repository Services object will also require a connection to the R/3 system. Therefore, you also need to set the Repository Services object set with the appropriate connection properties before calling any BAPI or function module.

You can use the *SAP Automation DCOM Connector Logon Component* to set up the connection properties of the Repository Services object in the same way that you can use it to set up the

connection properties of the BAPI Gateway object. The Repository Services object has a Connection property. You can assign the Connection object of the *SAP Automation DCOM Connector Logon Component* to the Connection property of the Repository Services object.

Procedure

1. Set up destination systems for your program to use, by using the Destination editor of the *SAP DCOM Connector*.

    This sets up an entry for each destination in the Windows Registry.

2. Create a Connection object from the *SAP Automation DCOM Connector Logon Component*.

3. Use the Logon method of the *SAP Automation DCOM Connector Logon Component* to display a Logon dialog to your end user. When the user dismisses the dialog by choosing OK, the various logon values are filled into the appropriate properties of the Connection object.

4. Create a Repository Services object.

5. Assign the Connection object you have created with the *SAP Automation DCOM Connector Logon Component* to the Connection property of the Repository Services object.

6. Create a BAPI Gateway object.

7. Assign the Connection object you have created with the *SAP Automation DCOM Connector Logon Component* to the Connection property of the BAPI Gateway object.

The following diagram summarizes this process:
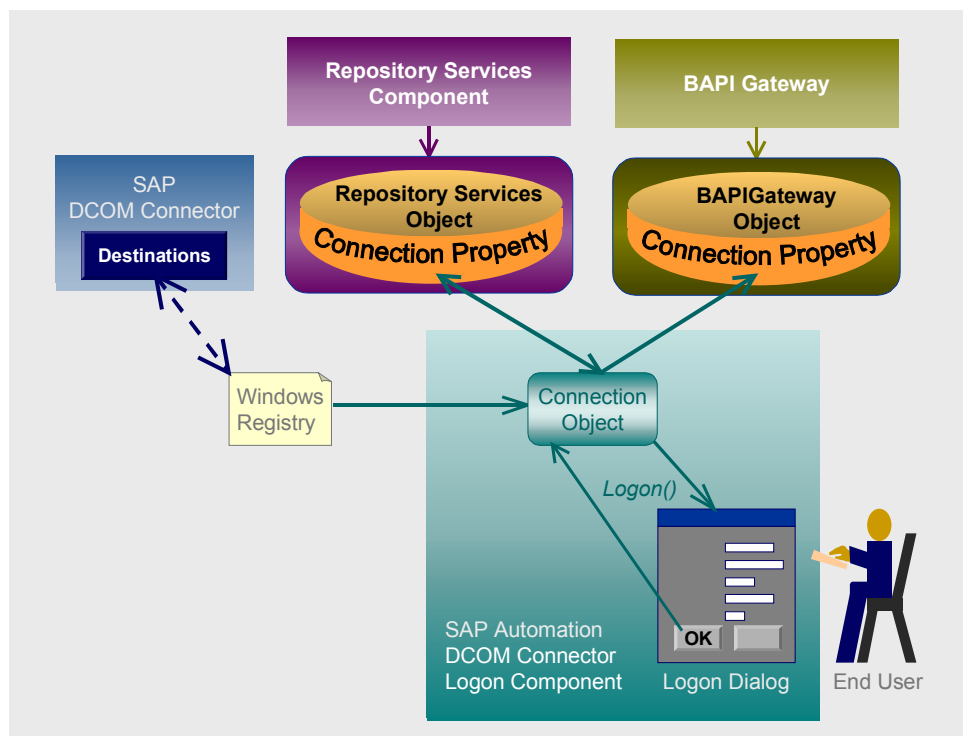
Setting Up the Necessary Connection properties

Minimum Requirement: Specifying Destinations

When you set up destinations with the *SAP DCOM Connector*, you can specify the client, user ID, and even password for the connection.

It is not likely that you would want to store password information in the Windows Registry in a production environment. However, if you do specify all these parameters, the *BAPI Gateway* can get this information given a destination you specify.

Therefore, if the Destination entry in the *SAP DCOM Connector* (and in the Windows registry) includes all the necessary parameters to log onto the destination R/3 system, all your program has to specify is the destination. In this case, only the destination property of your BAPI Gateway object has to be set.

However, if any of these parameters are not included in the *SAP DCOM Connector* entry, your program or your end user must supply them.

Example

The following code uses the *SAP Automation DCOM Connector Logon Component* to set up the Connection object, and to get its connection properties set with data entered by an end user:

```
' Declare and Instantiate the Connection object
' (from the DCOM Connector Logon Component)
Dim conn As Connection
Set conn = New Connection
' Use the Logon method to display a Logon dialog
'     and fill the Connection object's properties
'     with the values entered by the user
conn.logon
```

The following code creates the Repository Services object and sets its Connection property to point to the Connection object.

```
' Declare and Instantiate the Repository Services object
Dim rs As RepositoryServices
Set rs = New RepositoryServices
' Assign the Connection object
'    to the Connection property of the Repository Services object
rs.Connection = conn
```

The following code creates a BAPI Gateway Function object and assigns the Connection object to its Connection property:

```
' Declare and Instantiate a BAPI Gateway Function object
Dim gwFun As New BAPIGWLib.Function
' Assign the Connection object
'    to the Connection property of the BAPI Gateway Function object
gwFun.Connection = conn
```

The following code creates a BAPI Gateway BusinessObject and assigns the Connection object to its Connection property:

```
' Create the BAPI Gateway BusinessObject object
'    and assign the Connection object to its Connection property
Dim oCompanyCode As New BAPIGWLib.BusinessObject
oCompanyCode.Connection = conn
```

Result

You can now use the Repository Services object to get metadata into the BAPI Gateway objects you wish to use. You can then use this information when calling a BAPI or an RFC.

Preparing and Calling BAPIs

Use

Calling a BAPI at run time requires that you obtain the metadata for the specified BAPI at run time. You can use the *Repository Services component* to obtain this metadata.

Prerequisites

Set up the necessary connection properties in both the BAPI Gateway object and in the Repository Services object [Page 12]

Procedure

1. Create a BAPI Gateway *BusinessObject* object.

2. Create an object based on the *IBusinessObjects* collection of the Repository Services.

3. Find the Repository Services object for the business object whose BAPI you wish to call. You do so by finding the specific *IBusinessObject* in the *IBusinessObjects* collection of the Repository Services.

4. Use the *CreateFromRepository* method of the BAPI Gateway object to copy the metadata of the Repository Services *IBusinessObject* object into the BAPI Gateway *BusinessObject* object.

5. Use the *Call* method of the BAPI Gateway BusinessObject object to call the BAPI.

6. If no error occurred when calling the BAPI, get the data returned through the export parameters of the BAPI call.

You can use the *AdviseRfcGuiSink* method of the BAPI Gateway *BusinessObject* object before calling the BAPI. This invokes the SAPGUI, if it is installed, running the ABAP Debugger. It then hits a breakpoint at the next call to the underlying RFC.

Comprehensive Example: Preparing and Calling a BAPI

The following example calls two BAPIs of the *CompanyCode* business object: *GETLIST* and *GETDETAIL*.

The example sets the necessary Connection properties of the Repository Services object and of the BAPI Gateway object representing the SAP business object (The Prerequisites above).

```
' Record sets to hold returned data from BAPI calls

Dim oReturn As ador.Recordset

Dim oDetail As ador.Recordset

Dim otabCompanyCodes As ador.Recordset

...

' Create the Connection object and set its destination property

Dim oConnection As SAPLogonLib.Connection

Set oConnection = New SAPLogonLib.Connection

oConnection.Destination = "MySystem"
```

```
...
' Create the Repository Services object
'   and assign the Connection object to its Connection property
Dim oRepo As RepositoryServices
Set oRepo = New RepositoryServices
oRepo.Connection = oConnection
...
' Create the BAPI Gateway BusinessObject object
'   and assign the Connection object to its Connection property
Dim oCompanyCode As New BAPIGWLib.BusinessObject
oCompanyCode.Connection = oConnection
' Get metadata from the Repository Services
'       into the BAPI Gatway's BusinessObject object
Dim ibos As SAPRepositorySvc.IBusinessObjects
Set ibos = oRepo.BusinessObjects("CompanyCode")
Dim ibo As IBusinessObject
' Get to a specific business object
'           within the IbusinessObjects collection
'                   (of Repository Services)
Set ibo = ibos.ItemByIndex(1)
oCompanyCode.CreateFromRepository ibo
...
' Call the GETLIST method of the CompanyCode business object
'   and place the resulting data in oReturn and otabCompanyCodes
oCompanyCode.Method("GETLIST").Call
Dim oReturn As ador.Recordset
Set oReturn = oCompanyCode.Method("GETLIST").Parameter("RETURN").Value
Set otabCompanyCodes =
oCompanyCode.Method("GETLIST").Parameter("COMPANYCODELIST").Value
...
' Call the GETDETAIL BAPI of the CompanyCode business object
'   and place the resulting data in oReturn and oDetail
oCompanyCode.Key("COMPANYCODEID").Value = "CoXYZ"
oCompanyCode.Method("GETDETAIL").Call
Set oReturn = oCompanyCode.Method("GETDETAIL").Parameter("RETURN").Value
```

Preparing and Calling BAPIs

```
Set oDetail =
oCompanyCode.Method("GETDETAIL").Parameter("COMPANYCODEDETAIL").Value

...
```

Preparing and Calling Function Modules

Use

Calling an RFC function module at run time requires that you obtain the metadata for the specified function module at run time. You can use the *Repository Services component* to obtain this metadata.

Prerequisites

Set up the necessary connection properties in both the BAPI Gateway object and in the Repository Services object [Page 12]

Procedure

7. Create a BAPI Gateway *Function* object.

8. Find the Repository Services *IFunction* object for the function module you wish to call. You do so by finding the specific *IFunction* object in the IFunctions collection of the Repository Services.

9. Use the *CreateFromRepository* method of the BAPI Gateway *Function* object to copy the metadata of the Repository Services *IFunction* object into the BAPI Gateway *Function* object.

10. Use the *Call* method of the BAPI Gateway *Function* object to call function module.

11. If no error occurred when calling the function module, get the data returned through the export parameters of the function call.

You can use the *AdviseRfcGuiSink* method of the BAPI Gateway *Function* object before calling the function. This invokes the SAPGUI, if it is installed, running the ABAP Debugger. It then hits a breakpoint at the next RFC call.

Examples: Preparing and Calling Function Modules

The following example calls the RFC function module *RFC_CUSTOMER_GET* using the BAPI Gateway. The example includes the setup of the Connection object and of the Repository Services object

```
' Declare and Instantiate the Connection object
' (from the DCOM Connector Logon Component)
Dim conn As Connection
Set conn = New Connection
' Use the Logon method to display a Logon dialog
'     and fill the Connection object's properties
'     with the values entered by the user
conn.logon

' Declare and Instantiate the Repository Services object
Dim rs As RepositoryServices
Set rs = New RepositoryServices
' Assign the Connection object
'     to the Connection property of the Repository Services object
rs.Connection = conn

' Declare the Repository Services IFunction object
```

Preparing and Calling Function Modules

```
Dim rsFun As IFunction
' Set the Repository Services IFunction object
'         to the function module you wish to call
Set rsFun = rs.Functions("RFC_CUSTOMER_GET").ItemByIndex(1)

' Declare and Instantiate the BAPI Gateway Function object
Dim gwFun As New BAPIGWLib.Function
Dim cust As ADOR.Recordset
' Assign the Connection object
'     to the Connection property of the BAPI Gateway Function object
gwFun.Connection = conn
gwFun.CreateFromRepository rsFun
gwFun.Parameter("NAME1").Value = "S*"

'Use AdviseRfcGuiSink method of SAP DCOM Connection
'     to get debugging information
Set GuiSink = CreateObject("RfcGuiSink.RfcGuiSink.1")
gwFun.AdviseRfcGuiSink GuiSink, 1, 0

' Call the function module RFC_CUSTOMER_GET
gwFun.Call

' Display the resulting record set in spreadsheet-like format
'         using OLE control available with VB 6.0
Set MSHFlexGrid1.Recordset = gwFun.Parameter("CUSTOMER_T").Value
```

The following example calls the RFC function module *CATS_READ_TIMESHEET_DATA.* Setting up the Connection object is similar to the previous example:

```
' Get the metadata for the function you specify
Dim rsFun As IFunction
Set rsFun = rs.Functions("CATS_READ_TIMESHEET_DATA").ItemByIndex(1)
'Create the BAPI Gateway Function object,
'         get its metadata from the Repository services
Dim gwFun As New BAPIGWLib.Function
gwFun.Connection = conn
gwFun.CreateFromRepository rsFun

' Specify the parameters of the function module
gwFun.Parameter("I_KEY_DATE") = "1/15/1999"
gwFun.Parameter("i_pernr") = "70000000"
gwFun.Parameter("i_profile") = "ESS"

' Dislay debugging information
Set GuiSink = CreateObject("RfcGuiSink.RfcGuiSink.1")
gwFun.AdviseRfcGuiSink GuiSink, 1, 0

' Call the Function module
gwFun.Call

' Display the resulting data
Set MSHFlexGrid1.Recordset = gwFun.Parameter("E_CATSD").Value
```