

Applies To:

Asynchronous BAPI-ALE communication on Netweaver 2004s.

Summary

The following scenario will be demonstrated and explained: -

Source System creates a calendar appointment in a Target System using the ASynchronous BAPI-ALE communication mechanism.

The Source System requests the creation of an appointment. It uses a custom program to call a Business Object's method – BAPI. The method will create an outbound IDoc that will be sent to the Target System, and be received as an inbound IDoc on the Target System. The inbound IDoc will be processed accordingly in the Target System, resulting in a newly created appointment. The appointment can be viewed in Target System's SAP Office, Appointment Calendar (SSC1). The Target System does not return any messages.

To demonstrate *true* system independence between the Source and Target Systems, they must not in the same SAP System (SID).

The tilde symbol “~”, denotes a more advanced knowledge understanding. Information in this document is included for completeness.

Error handling code has been omitted to avoid complicating code and issues.

By: Glen Spalding

Company: gingle ltd

Date: 16th January 2006

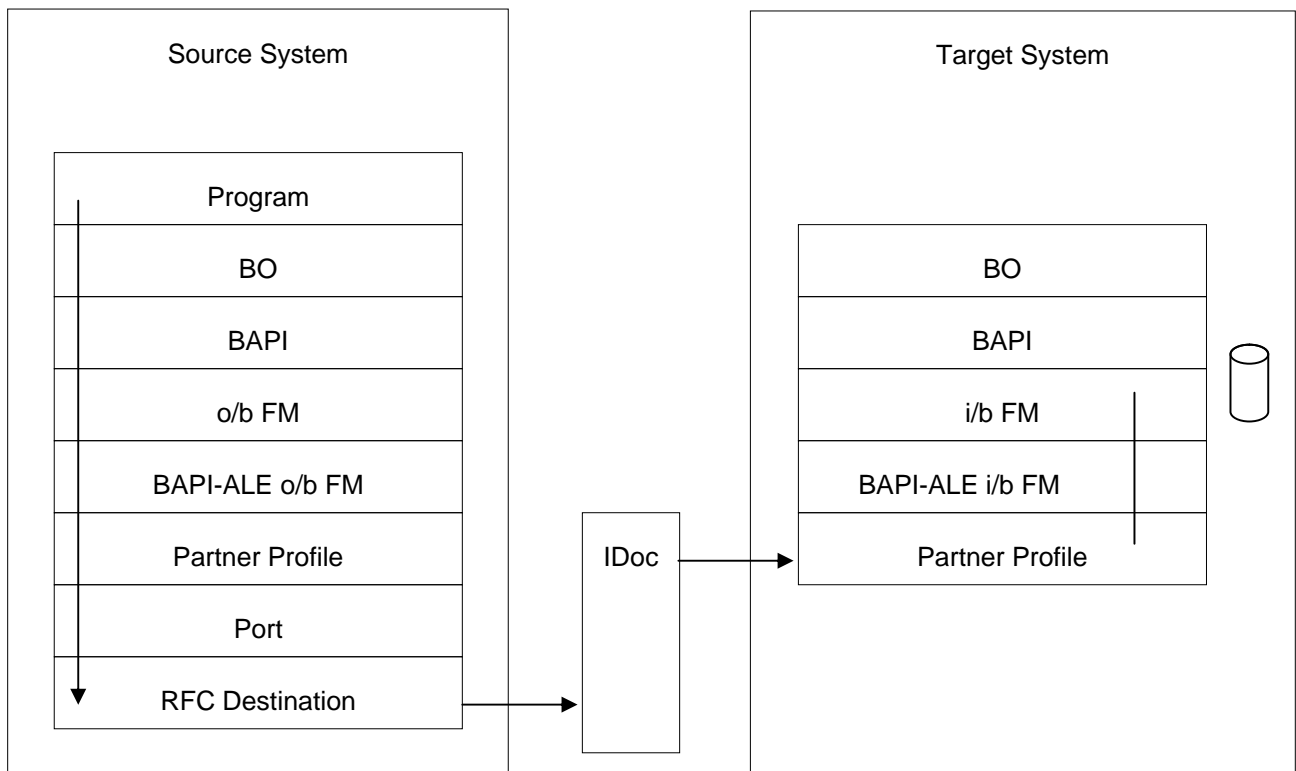
Table of Contents

Applies To:.....	1
Summary	1
Table of Contents	2
Object Stack	3
Outbound Tasks	4
Function Module.....	4
Business Object's BAPI	6
BAPI-ALE Interface	8
Processing Code.....	10
Distribution Model	13
Partner Profiles	14
Distribution of the Distribution Model	15
Calling Program	16
Inbound Tasks	19
Function Module.....	19
Processing Code.....	21
Business Object's BAPI	22
BAPI-ALE Interface	24
Partner Profiles	26
Testing	27
Appendices	28
RFC Destination.....	28
Ports	29
Releasing BAPIs	30
BAPI Explorer – BAPI	30
SAP Help.....	31
Author Bio	31
Disclaimer & Liability Notice	32

Object Stack

o/b – outbound

i/b – inbound



Notice the inbound process flow exits during the i/b FM. Above the i/b FM sits a BAPI and supporting BO. They are required for the generation of the BAPI-ALE interface objects, but are not utilized during the processing of the inbound IDoc.

Outbound Tasks

Outbound tasks are performed on the Source System.

Function Module

Create a bespoke Outbound Processing Function Module (o/b FM) that is responsible for creating the IDoc sent to the Target System – SE37.

As the o/b FM is to be used in supporting a BAPI, strict requirements will need to be adhered to.

Min requirements:

- Do not use “type” for parameters. Use “like”
 - Import parameters must be passed in as “value”
 - Must have “Return” parameter
 - Must be “RFC enabled”
- ~ For full BAPI compliance, check the “BAPI Project” in the “BAPI Explorer” - BAPI
 - ~ If Filter Objects are to be used, parameters must be passed as tables (i.e. multiline in BAPI parameters).

To create the appointment we are going to use the following data, which represent the o/b FM's interface.

date from
time from
date to
time to
description
participant_list (this is the list of users for whom the appointments will be created for).

~ Assign the o/b FM to an appropriate Function Group, or create a new one – SE80, or SE37.

~ The Function Group can be a “Local Object/\$tmp”.

At this stage, we do not have to program the logic to create the IDoc. What we need, is the o/b FM's interface. Don't forget the minimum requirements stated above.

The o/b FM interface should look like this: -

```
FUNCTION zbapi_req_appointment.  
  
* "-----  
  
* " "Local interface:  
  
* "   IMPORTING  
  
* "       VALUE (DATE_FROM) LIKE   SCAPPT-DATE_FROM  
* "       VALUE (TIME_FROM) LIKE   SCAPPT-TIME_FROM  
* "       VALUE (DATE_TO)  LIKE   SCAPPT-DATE_TO  
* "       VALUE (TIME_TO)  LIKE   SCAPPT-TIME_TO  
* "       VALUE (DESC)    LIKE   SCAPPT-TXT_SHORT  
* "   EXPORTING
```

```
* "      VALUE(RETURN) LIKE  BAPIRET2 STRUCTURE  BAPIRET2
* "  TABLES
* "      PARTICIPANT_LIST STRUCTURE  SCSPARINC
* "  -----
```

```
ENDFUNCTION.
```

Business Object's BAPI

We now need to create a BAPI for the outbound processing. This is implemented as a Business Object's (BO) method, and will utilize the o/b FM previously created.

~If the BAPI-ALE interface generation (done later) is to be performed via "BAPI Explorer" – BAPI, then the BO needs to be "Released". You can only "Release" the BO if it is "Transportable", therefore, should not be created as a "Local Object". For "Releasing" BAPIs, see appendices.

For this exercise, we will generate the BAPI-ALE interface directly using "Generate ALE Interface for BAPI" – BDBG, for which the BO can be saved as a "Local Object".

In the Source System, begin by creating the BO – SWO1

Supply the appropriate values, and create as "Local Object"

Create the BO's Method.

Click on the "Methods" node, press F5, and select "Yes" to "Create with function module as template?"

Then supply the bespoke o/b FM earlier created

Adjust fields names accordingly, e.g.

Method	ReqAppointment
Name	Request Appointment
Description	Request Appointment

Change the Property of the Method to "Instance-independent" (not mandatory) and continue by pressing the "Forward" icon

Asynchronous BAPI-ALE Communication

Name in function module	Method parameter	Name
DATE_FROM	DateFrom	Start Date
TIME_FROM	TimeFrom	From Time
DATE_TO	DateTo	End Date
TIME_TO	TimeTo	To Time
DESC	Desc	Short text
RETURN	Return	Return Parameter
PARTICIPANT_LIST	ParticipantList	Interface / Include

Leave parameter defaults, and continue forward

Press "Yes" to confirm the "Extend Program" popup, "Method ... not yet implemented", generation for the Method's missing section

The Method is now created

The screenshot shows the 'ABAP' tab in the SAP Object Explorer. Under the 'General' section, the 'API function' radio button is selected. Below it, the 'Name' field is populated with 'ZBAPI_REQ_APPOINTMENT'.

Double click the Method, and navigate to the "ABAP" tab, then select the "API function" option

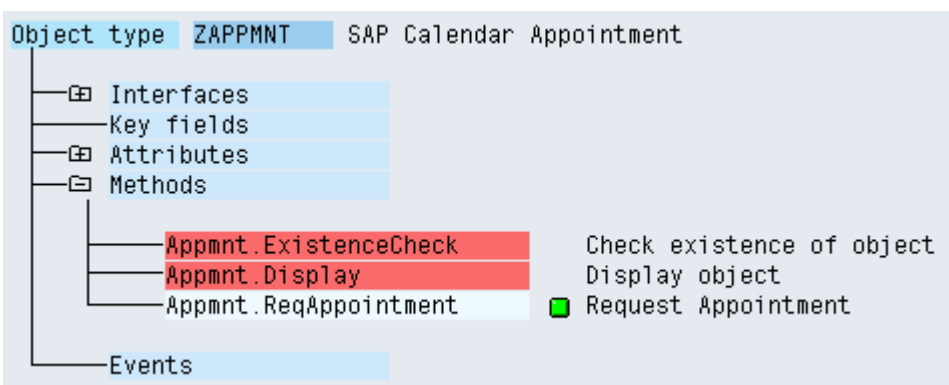
"Save" the BO

"Implement" the BO so that it can be *generated*, Menu path

Edit->Change release status ->Object type ->To implemented

"Generate" the BO

In the end, the BO should look something like the following: -

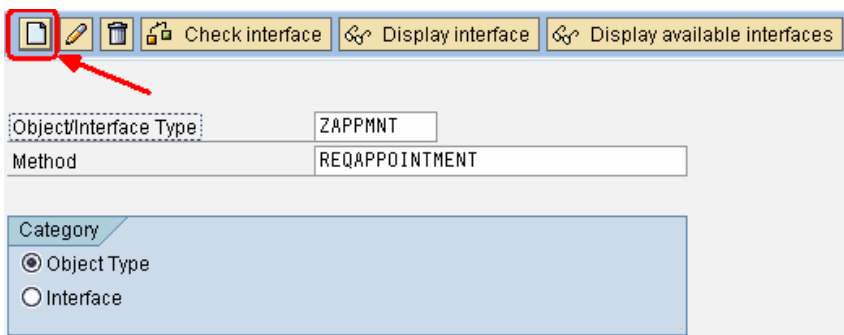


BAPI-ALE Interface

To assist in the process of Async BAPI communication, SAP has implemented functionality that can generate the standard objects and code for bespoke utilization. This greatly reduces time and effort throughout the creation process.

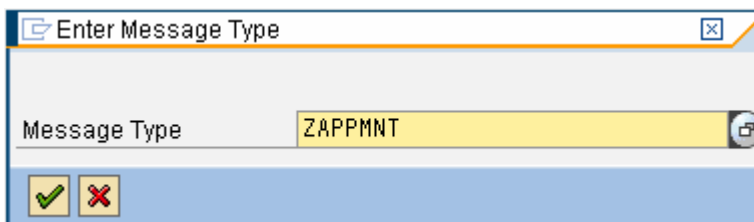
~If the generation is to be performed via the BAPI Explorer, then the BAPI will need to be “Released”. For this exercise will generate the objects manually using transaction BDBG, and so, there is no requirement to “Release” the BAPI. See appendices for “Releasing” BAPIs, and creating the BAPI-ALE interface objects via the BAPI Explorer.

Run transaction BDBG



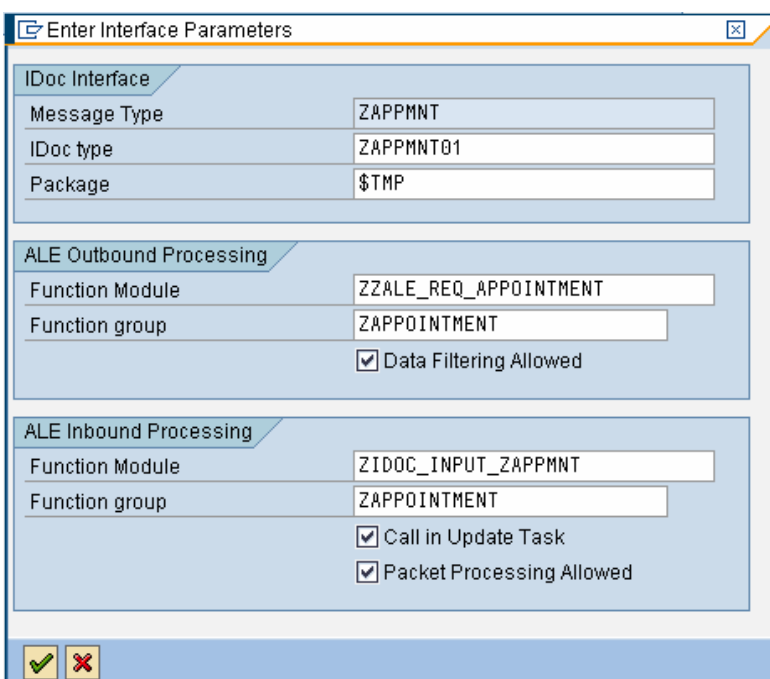
Enter the appropriate values for your BO and BAPI (Method), for “Category” – “Object type”

Press the “Create” icon, or F5.



Type a name for your “Message type”, that is going to be automatically created.

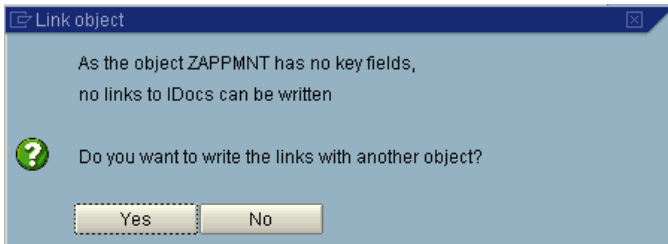
~Be mindful of backward compatibility issues. i.e. over 8 characters



~ For future proofing, it may be prudent to select the “Data Filtering Allowed”, “Call in Update Task”, and “Packet Processing Allowed” check boxes.

Accept the default name values or change to suit. Make sure the “Function group”, and “Package” defaults are appropriate.

Confirm/create any “Workbench Request”



If the BO does not have any key fields, the following popup appears, click "No" to continue.

Message type
ZAPPMNT
ZAPPMNT has been generated
IDoc type
ZAPPMNT01
Check basic type ZAPPMNT01
Basic type ZAPPMNT01 is linked to logical message ZAPPMNT
No predecessors exist
Basic type ZAPPMNT01 is not released
Segment
Z1ZAPPMNT
Z1ZAPPMNT has been generated
Z1SCSPARINC
Z1SCSPARINC has been generated
Function Module for Outbound ALE With Data Filtering
ZZALE_REQ_APPOINTMENT
ZZALE_REQ_APPOINTMENT has been generated
Function Module for Inbound ALE With Packet Processing
ZIDOC_INPUT_ZAPPMNT
ZIDOC_INPUT_ZAPPMNT has been generated

All the necessary objects have now been created.

Click on each of the generated **Segment's Yellow nodes**, to jump to WE31, where each segment will require "Releasing". It is not evident from the Object creation summary that this needs to be performed. Do the Segments first.

WE31 – Menu path

Edit->Set release

Do the same for the IDoc type, and confirm prompt.

WE30 – Menu path

Edit->Set release

Checking message type
Message type ZAPPMNT exists already
Check basic type ZAPPMNT01
Basic type ZAPPMNT01 exists
Basic type ZAPPMNT01 is released
Basic type ZAPPMNT01 is linked to logical message ZAPPMNT
No predecessors exist
Check segment Z1ZAPPMNT
Segment Z1ZAPPMNT consistent
Check segment Z1SCSPARINC
Segment Z1SCSPARINC consistent
Checking outbound function module
ZZALE_REQ_APPOINTMENT has been generated
Checking inbound function module
ZIDOC_INPUT_ZAPPMNT has been generated

Return to the main screen of BDBG, and perform a "Check interface".

~Entries for BAPI-ALE are written in table TBDDBE.

Processing Code

Having generated the BAPI-ALE objects, we are now in a position to utilize the BAPI-ALE o/b FM.

Return to the bespoke o/b FM created in the first step, and program a call function to the BAPI-ALE o/b FM.

Pass the parameter from the bespoke o/b FM directly to the BAPI-ALE o/b FM. See below.

```
FUNCTION ZBAPI_REQ_APPOINTMENT.

* " -----
* " "Local interface:
* "   IMPORTING
* "       VALUE( DATE_FROM ) LIKE SCAPPT-DATE_FROM
* "       VALUE( TIME_FROM ) LIKE SCAPPT-TIME_FROM
* "       VALUE( DATE_TO ) LIKE SCAPPT-DATE_TO
* "       VALUE( TIME_TO ) LIKE SCAPPT-TIME_TO
* "       VALUE( DESC ) LIKE SCAPPT-TXT_SHORT
* "   EXPORTING
* "       VALUE( RETURN ) LIKE BAPIRET2 STRUCTURE BAPIRET2
* "   TABLES
* "       PARTICIPANT_LIST STRUCTURE SCSPARINC
* " -----

CALL FUNCTION 'ZZALE_REQ_APPOINTMENT'

EXPORTING
    datefrom           = date_from
    timefrom           = time_from
    dateto             = date_to
    timeto             = time_to
    desc               = desc
* OBJ_TYPE             = 'ZAPPMNT'
* SERIAL_ID           = '0'
TABLES
    participantlist    = participant_list
    receivers          =
* COMMUNICATION_DOCUMENTS =
* APPLICATION_OBJECTS   =

EXCEPTIONS
    error_creating_idocs = 1
    OTHERS               = 2

.
```

```
IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
ENDFUNCTION.
```

Note:

- We need to pass a new parameter to the BAPI-ALE o/b FM – “receivers”. This is the list of Target Systems retrieved from the Distribution Model where the BAPI will be configured later. To determine the Target Systems (Receivers) we use the standard SAP FM “ALE_ASYNC_BAPI_GET_RECEIVER” and pass it our BO, and BAPI names.
- The “filterobject_values” being returned into the program, is not being used in this example. It is present, because it is a mandatory parameter.
- You supply the BO “**name**”, and not “**type**” to “Object” parameter of “ALE_ASYNC_BAPI_GET_RECEIVER”.

The final bespoke o/b FM should look something like this: -

```
FUNCTION ZBAPI_REQ_APPOINTMENT.
* " -----
* " "Local Interface:
* "   IMPORTING
* "       VALUE( DATE_FROM ) LIKE   SCAPPT-DATE_FROM
* "       VALUE( TIME_FROM ) LIKE   SCAPPT-TIME_FROM
* "       VALUE( DATE_TO ) LIKE     SCAPPT-DATE_TO
* "       VALUE( TIME_TO ) LIKE     SCAPPT-TIME_TO
* "       VALUE( DESC ) LIKE        SCAPPT-TXT_SHORT
* "   EXPORTING
* "       VALUE( RETURN ) LIKE      BAPIRET2 STRUCTURE BAPIRET2
* "   TABLES
* "       PARTICIPANT_LIST STRUCTURE SCSPARINC
* " -----
DATA: lt_receivers TYPE STANDARD TABLE OF bdi_logsys,
      lt_filters   TYPE STANDARD TABLE OF bdi_fobj.

CALL FUNCTION 'ALE_ASYNC_BAPI_GET_RECEIVER'
EXPORTING
    object          = 'ZAPPMNT'
    method          = 'REQAPPOINTMENT'
TABLES
```

```
*  RECEIVER_INPUT                                =
    receivers                                    = lt_receivers
    filterobject_values                          = lt_filters
EXCEPTIONS
    error_in_filterobjects                       = 1
    error_in_ale_customizing                     = 2
    OTHERS                                       = 3
.
IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

CALL FUNCTION 'ZZALE_REQ_APPOINTMENT'
    EXPORTING
        datefrom                                = date_from
        timefrom                                = time_from
        dateto                                  = date_to
        timeto                                  = time_to
        desc                                    = desc
*  OBJ_TYPE                                      = 'ZAPPMNT'
*  SERIAL_ID                                    = '0'
    TABLES
        participantlist                         = participant_list
        receivers                              = lt_receivers
*  COMMUNICATION_DOCUMENTS                      =
*  APPLICATION_OBJECTS                          =
EXCEPTIONS
    error_creating_idocs                       = 1
    OTHERS                                       = 2
.
IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
ENDFUNCTION.
```

Distribution Model

Create/amend a Distribution Model – BD64.

Press F9 for edit mode.

Menu path

Edit->Model view->Create

Or "Ctrl + F4"

Provide a short description and "technical name".

Distribution Model	Description/ technical name
Model views	
CRM Scenarios	CRMSZ
Customizing Data Synchronization	CONTRLDATA
HR <-> FI Scenario	HRFICOUPLI
Internet Scenarios	INTERNET
Logistics Scenarios	LOGISTICS
Master Data Distribution (MDM)	MASTERDATA
BAPI ALE Scenario	BAPI-ALE

Select the created Distribution Model, then press the "Add BAPI" button.

Enter the appropriate Logical Systems for the Sender (Source) and Receiver (Target) Systems, then the BO and Method for the outbound processing.

Entries for the Obj Name, and Method, is case sensitive, suggest using F4.

Model views		
BAPI ALE Scenario		BAPI-ALE
Netweaver Sneek Preview		NSPCLNT000
Gingle		GNGCLNT500
Appmnt.ReqAppointment		Request Appointment

Expand the Model to check the entries, then "Save".

Partner Profiles

~Recommend configuring the [RFC Destination](#) and [Port](#) to the Target Systems manually prior to automatic generation of the Partner Profiles. See [Appendices](#) for instructions.

Generate the Partner Profiles, from the Distribution Model – BD64

Select the Distribution Model created earlier, then generate the partner profile.

Menu path

Environment -> Generate partner profiles

Make sure the “Model view” is correct

~ For now, we will retain the “Postprocessing: Authorized processors” settings.

~ Set the appropriate default parameters.

In this case,

“Version” = 3

“Packet Size” = 1

“Transfer IDoc immediately”

“Trigger immediately”

Then “Execute” - F8

The following summary should be displayed confirming the valid creation of the Partner Profiles.

Log for Partner Profile Generation	
Partner	
System GNGCLNT500	Partner GNGCLNT500 as partner has been created
System NSPCLNT000	Partner NSPCLNT000 as partner has been created
Port	
System GNGCLNT500	Port GNGCLNT500 with RFC destination GNGCLNT500 already exists
Outb. Parameters	
System GNGCLNT500	Outbound parameters for message type SYNCH SYNCHRON successfully created Outbound parameters for message type ZAPPMNT ZAPPMNT01 successfully created

If some entries indicate that they already exist, this is ok.

Use WE20 to view/create the Partner Profiles manually.

Distribution of the Distribution Model

~Distribution of the Distribution Model is only necessary should you be configuring returning Messages from the Target System (Outbound messages from the Target System back to the Source System), or wish to automatically generate the Partner Profiles from within the Target System. For this exercise, we will be performing an automatic generation for the Partner Profiles, therefore, we will distribute the Model.

In the Source System, select the appropriate Distribution Model, and distribute the model – BD64.

Menu path

Edit->Model view->Distribute

Model view BAPI-ALE	
Receiver of model view	
Logical system	Technical name
Client 100	NSPCLNT100
Client 200	NSPCLNT200
Gingle	GNGCLNT500

Systems that will receive the Model are, by default, selected.

Accept by pressing "Enter" or the "Tick" icon.

Confirmation of the distribution is given

Distribution of model view BAPI-ALE	
Target system GNGCLNT500	Model view BAPI-ALE has been created

Log on to the Target System, and check the Distribution Model is present – BD64.

The Distribution Models received from another systems will always appear "Grayed Out", even if in change mode. The model cannot be changed, however, it can be deleted, in the Target System, as normal.

Calling Program

We need to create a standard program is responsible for starting the communication process. Use the ABAP Editor – SE38

Utilize the code example below.

```
*&-----*
*& Report  ZREQE_APPOINTMENT                      *
*&                                                *
*&-----*
*&                                                *
*&                                                *
*&-----*
```

```
REPORT  ZREQ_APPOINTMENT                      .
    INCLUDE: <cntain>.
```

```
DATA: l_return          TYPE swotreturn,
      lt_cont           TYPE swconttab,
      l_objhnd          TYPE swo_objhnd,
      lt_participantlist TYPE STANDARD TABLE OF scsparinc,
      ls_participantlist TYPE scsparinc.
```

```
*****
* parameter screen
PARAMETERS:
    p_datefm TYPE sc_datefro DEFAULT sy-datum,
    p_timefm TYPE sc_timefro DEFAULT sy-uzeit,
    p_dateto TYPE sc_dateto  DEFAULT sy-datum,
    p_timeto TYPE sc_timeto  DEFAULT sy-uzeit,

    p_desc   TYPE sc_txtshor DEFAULT 'BAPI Test',

    p_user   TYPE syuname    DEFAULT sy-uname.
```

```
*****
* instantiate object
CALL FUNCTION 'SWO_CREATE'
```



```
EXPORTING
    objtype          = 'ZAPPMNT'
    objname          = 'APPMNT'
IMPORTING
    object           = l_objhnd
EXCEPTIONS
    no_remote_objects = 1
    OTHERS           = 2.

IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

*****
* pass data to container for bapi

swc_set_element lt_cont 'DateFrom' p_datefm.
swc_set_element lt_cont 'TimeFrom' p_timefm.
swc_set_element lt_cont 'DateTo' p_dateto.
swc_set_element lt_cont 'TimeTo' p_timeto.
swc_set_element lt_cont 'Desc' p_desc.

ls_participantlist-participan = p_user.
APPEND ls_participantlist TO lt_participantlist.

swc_set_table lt_cont 'ParticipantList' lt_participantlist.

*****
* fire BAPI
CALL FUNCTION 'SWO_INVOKE'
    EXPORTING
*       ACCESS          = 'C'
        object          = l_objhnd
        verb            = 'REQAPPOINTMENT'
*       PERSISTENT      = ' '
```

```
*      REQUESTER                = ' '
*      SYNCHRON                 = '*'
*      UNSORTED_CONTAINER       = ' '
*      NO_ARFC                  = ' '

IMPORTING
    return                      = l_return
*      VERB                     =
*      MODE_ID                  =

TABLES
    container                   = lt_cont.

*****
* must have this, else the receiving system never sees the inbound idoc
* COMMIT WORK.

*****
* write out return parameter
WRITE:/ 'holá amigo', l_return.
```

Pay special attention to: -

- The instantiating of the BO
- The passing of parameters from the program, to the BO via the use of a “container”
- The calling of the BAPI
- The “commit work” code at the end

At this point, an outbound message could be sent by executing the program above.

However, no inbound processing programs or configuration has been made.

Inbound Tasks

Earlier we created the BAPI-ALE interface on the Source System. Although the IDoc and both outbound and inbound BAPI-ALE FMs were created in the Source System, we do not have visibility of them in the Target System.

Normally we would transport the Message Type, IDoc and Inbound FM to the Target, however, should the Target System not be on a Transport Path, as in this exercise, we must replicate some of the actions performed in the Outbound Processing steps, over in the Target System.

Inbound tasks are performed on the Target System.

Function Module

Create the bespoke Inbound Processing Function Module (i/b FM) that is responsible for creating the appointment in the Target System – SE37.

Once again, as the i/b FM is to be used in supporting a BAPI, strict requirements will need to be adhered to.

Min requirements:

- Do not use “type” for parameters. Use “like”
- Import parameters must be passed in as “value”
- Must have “Return” parameter
- Must be “RFC enabled”

~ For full compliance check the “BAPI Project” in the “BAPI Explorer” - BAPI

~ If Filter Objects are to be used, parameters must be passed as tables (i.e. multiline in BAPI parameters).

The interface of this bespoke i/b FM needs to be exactly the same as with the bespoke o/b FM in order for the IDoc to be accurately generated from the BAPI-ALE interface objects, and compatible with what the Source System is going to send.

Therefore, replicate the parameters from the bespoke o/b FM in the Source System, for the bespoke i/b FM in the Target System.

~ Assign the o/b FM to an appropriate Function Group, or create a new one – SE80, or SE37.

~ The Function Group can be a “Local Object/\$tmp”.

Name the i/b appropriately.

The bespoke i/b FM should look something like this: -

```
FUNCTION ZBAPI_CREATE_APPOINTMENT.  
* "-----  
* " "Local interface:  
* " IMPORTING  
* "     VALUE (DATE_FROM) LIKE SCAPPT-DATE_FROM  
* "     VALUE (TIME_FROM) LIKE SCAPPT-TIME_FROM  
* "     VALUE (DATE_TO) LIKE SCAPPT-DATE_TO  
* "     VALUE (TIME_TO) LIKE SCAPPT-TIME_TO
```

```
* "      VALUE(DESC) LIKE  SCAPPT-TXT_SHORT
* "      EXPORTING
* "      VALUE(RETURN) LIKE  BAPIRET2 STRUCTURE  BAPIRET2
* "      TABLES
* "      PARTICIPANT_LIST STRUCTURE  SCSPARINC
* " -----
```

```
ENDFUNCTION.
```

Processing Code

The generated BAPI-ALE interface for the inbound FM will call this code explicitly and pass the IDoc values directly to it. In this example, we will forward these values directly to a standard FM that will actually create the appointments – “APPT_CREATE”.

The bespoke i/b/ FM should look something like this: -

```
FUNCTION ZBAPI_CREATE_APPOINTMENT .  
  
* "-----  
* "Local interface:  
* " IMPORTING  
* "     VALUE( DATE_FROM ) LIKE SCAPPT-DATE_FROM  
* "     VALUE( TIME_FROM ) LIKE SCAPPT-TIME_FROM  
* "     VALUE( DATE_TO ) LIKE SCAPPT-DATE_TO  
* "     VALUE( TIME_TO ) LIKE SCAPPT-TIME_TO  
* "     VALUE( DESC ) LIKE SCAPPT-TXT_SHORT  
* " EXPORTING  
* "     VALUE( RETURN ) LIKE BAPIRET2 STRUCTURE BAPIRET2  
* " TABLES  
* "     PARTICIPANT_LIST STRUCTURE SCSPARINC  
* "-----  
  
CALL FUNCTION 'APPT_CREATE'  
EXPORTING  
    date_from           = date_from  
    date_to             = date_to  
    time_from           = time_from  
    time_to             = time_to  
    description         = desc  
* IMPORTING  
* ERROR_MESSAGE         =  
TABLES  
    participant_list    = participant_list.  
  
ENDFUNCTION.
```

Business Object's BAPI

We must now build a BO, to generate the BAPI-ALE Interface Objects, i.e. Message Type, IDoc, and FMs.

Create a new BO and Method for the BAPI – SWO1

Give appropriate values and create as a “Local Object”.

Create the BO's Method.

Click on the “Methods” node, press F5, and select “Yes” to “Create with function module as template?”

The supply the bespoke **i/b** FM earlier created

Rename other fields appropriately, e.g.

Name	Create Appointment
Description	Create Appointment

Change the Property of the Method to “Instance-independent” (not mandatory) and continue by pressing the “Forward” icon

Name in function module	Method parameter	Name
DATE_FROM	DateFrom	Start Date
TIME_FROM	TimeFrom	From Time
DATE_TO	DateTo	End Date
TIME_TO	TimeTo	To Time
DESC	Desc	Short text
RETURN	Return	Return Parameter
PARTICIPANT_LIST	ParticipantList	Interface / Include

Leave parameter defaults, and continue forward

Press “Yes” to confirm the “Extend Program” popup, “Method ... not yet implemented”, generation for the Method’s missing section

The Method is now created

Double click the Method, and navigate to the “ABAP” tab, then select the “API function” option

“Save” the BO

“Implement” the BO so that it can be *generated*, Menu path

Edit->Change release status ->Object type ->To implemented

“Generate” the BO

BAPI-ALE Interface

Now generate the BAPI-ALE interface – BDBG.

Because the bespoke i/b FM supporting the BAPI has the identical interface as the bespoke o/b FM, when the BAPI-ALE interface gets generated, it will generate compatible objects.

Make sure the Message Type is named the same as in the Source System. Otherwise, the inbound Message Type will not be recognized, and the Target System will not know how to process it.

For completeness, select the same “Data Filtering Allowed”, “Call in Update Task”, and “Packet Processing Allowed” options as in the Source System.

Accept the default name values or change to suit. Make sure the “Function group”, and “Package” defaults are appropriate.

Confirm/create any “Workbench Request”

Again, as the BO does not have any key fields, the following popup appears, click “No” to continue.

All the necessary objects have now been created.

Click on each of the generated Segment’s **Yellow nodes**, to jump to WE31, where each segment will require “Releasing”. It is not evident from the Object creation summary that this needs to be performed. Do the Segments first.

WE31 – Menu path

Edit->Set release

Do the same for the IDoc type, and confirm prompt.

WE30 – Menu path

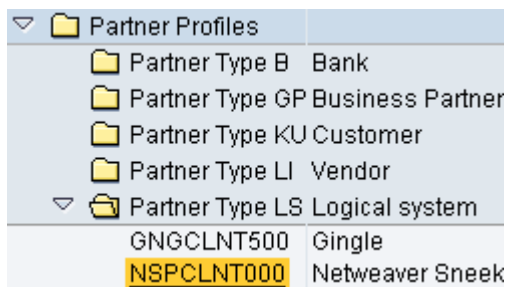
Edit->Set release

Return to the main screen of BDBG, and perform a “Check interface”.

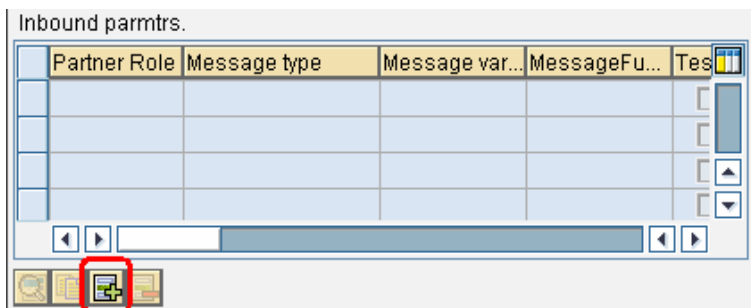
Partner Profiles

Inbound processing does not utilize the Distribution Model, therefore, we will create the Partner Profile configuration manually using transaction WE20. Sometimes it is useful to distribute the Model to the Target Systems for automatically generating the Partner Profile settings, however, as this is a simple setting, we shall do this manually.

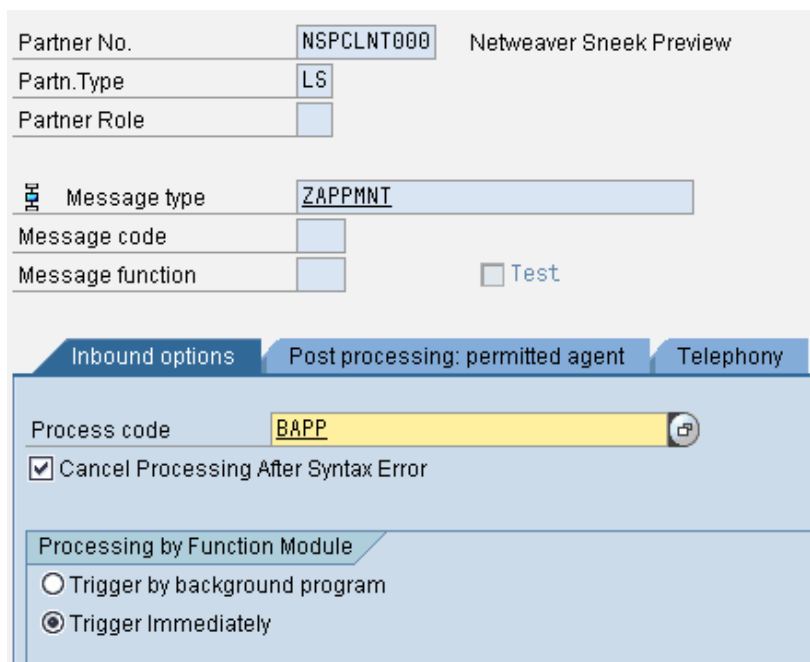
In Partner Profiles - WE20 select the Source System from the Logical System (Partner Type LS) choices.



In this example it is "NSPCLNT000".



Having the Source System now selected, continue to create the Inbound Message, by clicking on the "Add" button.



Partner No.	NSPCLNT000	Netweaver Sneek Preview
Partn.Type	LS	
Partner Role		
Message type	ZAPPMNT	
Message code		
Message function		<input type="checkbox"/> Test
Inbound options Post processing: permitted agent Telephony		
Process code	BAPP	
<input checked="" type="checkbox"/> Cancel Processing After Syntax Error		
Processing by Function Module		
<input type="radio"/> Trigger by background program		
<input checked="" type="radio"/> Trigger Immediately		

For the Message Type, use the specific Message that was created during the ALE-BAPI interface creation process.

For the Process Code, use BAPP – Packet Processing, as we selected the option "Packet Processing Allowed" when generating the ALE-BAPI interface objects. If the option "Packet Processing Allowed" was not selected, we would use Process Code "BAPI" – Individual Processing.

Save & exit.

On the Target System, run the Calling Program – SE38. Accepting defaults from the example program should work, assuming you have a user id in the Target System.

Start Date	18.05.2005
From Time	11:31:45
End Date	18.05.2005
To Time	11:31:45
Short text	BAPI Test
User	GS007

IDocs	IDoc status	Number
<ul style="list-style-type: none"> IDoc selection <ul style="list-style-type: none"> Changed on is in the range 18.05.2005 to 18.05.2005 Message type is equal to ZAPPMNT 		
<ul style="list-style-type: none"> CRM Test Client <ul style="list-style-type: none"> IDocs in outbound processing: <ul style="list-style-type: none"> Data passed to port OK ZAPPMNT <ul style="list-style-type: none"> IDoc sent to R/3 System or external program 	03	1 1 1 1

IDocs	IDoc status	Number
<ul style="list-style-type: none"> IDoc selection <ul style="list-style-type: none"> Changed on is in the range 18.05.2005 to 18.05.2005 Message type is equal to ZAPPMNT 		
<ul style="list-style-type: none"> R/3 Test Client <ul style="list-style-type: none"> IDoc in inbound processing Application document posted ZAPPMNT BAPI ZCREATE has been called successfully 	53	1 1 1 1

22.05.2005

16.05.2005-22.05.2005

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
08:00						
08:30						
09:00						
09:30						
10:00						
10:30						
11:00			BAPI Test ...			
11:30						
12:00						
12:30						
13:00						

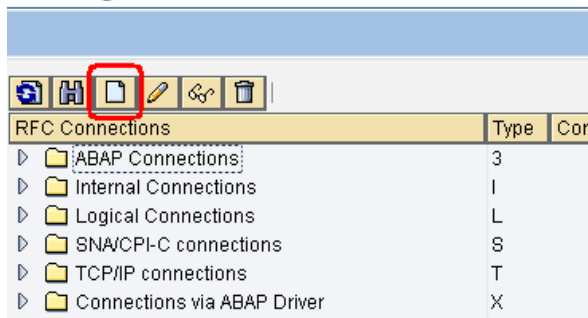
Appendices

RFC Destination

Create an “R/3 connection” RFC Destination – SM59

Configuration of RFC Connections

Select “ABAP Connections”, then press “Create”



RFc Destination: GNGCLNT500
 Connection Type: 3 ABAP Connection Description

Description
 Description 1: Gingle Client 500
 Description 2:
 Description 3:

Administration Technical Settings Logon & Security MDMP & Unicode Special Options

Target System Settings
 Load Balancing Status
 Load Balancing: ☐ Yes ☒ No
 Target Host: ferrari System Number:
 Save to Database as
 Save as: ☐ Hostname ☒ IP Address 10.10.0.10

Gateway Options
 Gateway Host: Delete
 Gateway service:

Provide a name for the “RFC Destination” using SAP convention (<SID>CLNT<xxx> where <SID> is the SAP System ID, and <xxx> is the Client Number. E.g. GNGCLNT500),

Make sure “Connection Type” is “3”.

Provide a “Description” & “Target Host” details.

Administration Technical Settings Logon & Security MDMP & Unicode Special Options

Security Options
 Trusted System/Logon Screen Status
 Trusted System: ☒ No ☐ Yes ☐ Logon Screen
 Status of Secure Protocol
 SNC: ☒ Inactive ☐ Active
 Authorization for Destination:

Logon
 Language: EN
 Client: 500
 User: SYSTEMUSER ☐ Current User
 PW Status: saved
 Password: *****

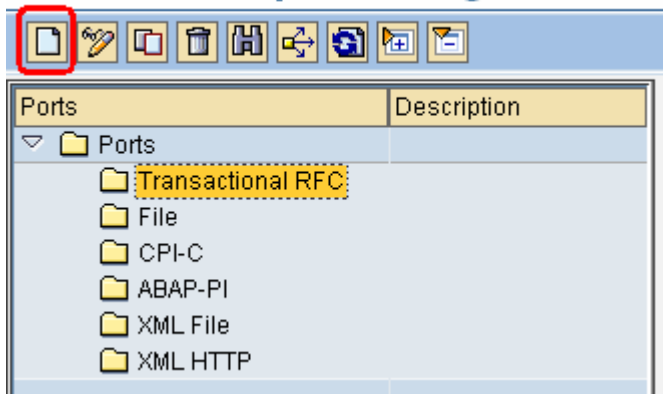
On the “Logon & Security” tab, in the “Logon” area, provide a “Language”, “Client”, “User” and “Password” to access the Target System. The “User” must be defined in the Target System.

“Save” and test the connection with the “Remote Logon” button. If the remote logon user is not type “Dialog”, and nothing appears to happen, the connection is good.

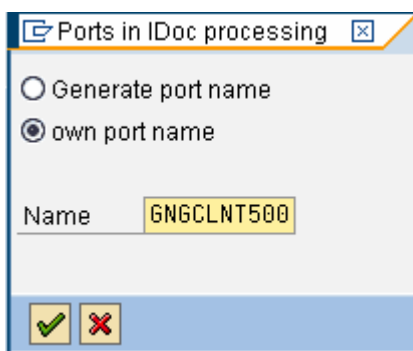
Ports

Create the tRFC Port – WE21.

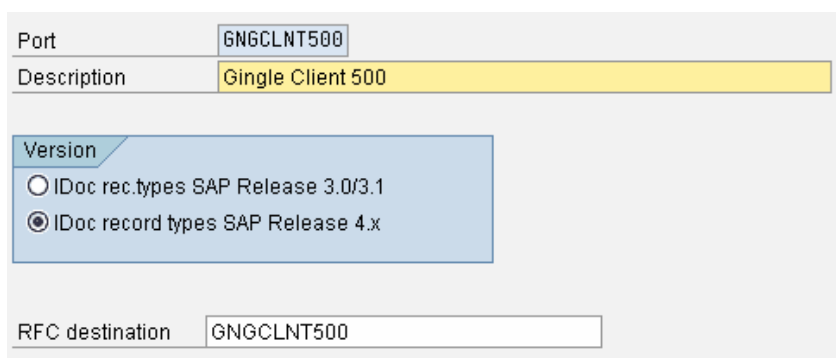
Ports in IDoc processing



Select the “Transactional RFC” node, then “Create” – F7.



Select “own port name” and supply a port description. Give it the same name as the RFC Destination it will utilize to access the Target System.



Provide a “Description”, and the “RFC Destination”

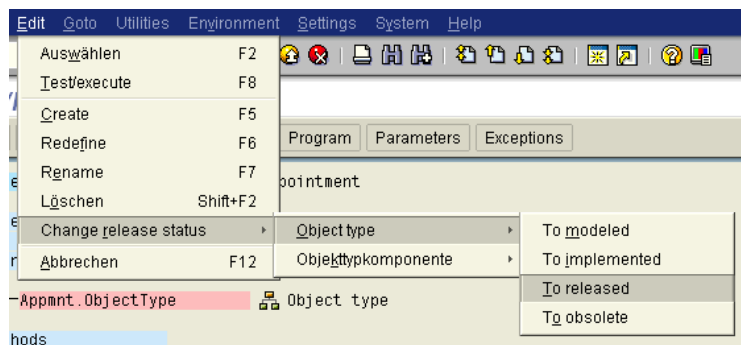
Note: When the RFC Destination, Port and Logical systems are named the same, the Generate Partner Profile functionality in the Distribution Model is automatically able to provide the Message / Partner Profile configuration.

Releasing BAPIs

In some cases, BAPIs need to be “Released”. This is not necessary for the examples here, but for completeness, the procedure is provided below.

Before the BAPI can be “Released”, the BO needs to be “Released”, and also the BAPI’s implementing FM.

Before the BO can be “Released”, it must belong to a “Transport Request”. If not, an error message occurs, ceasing further processing when attempting.

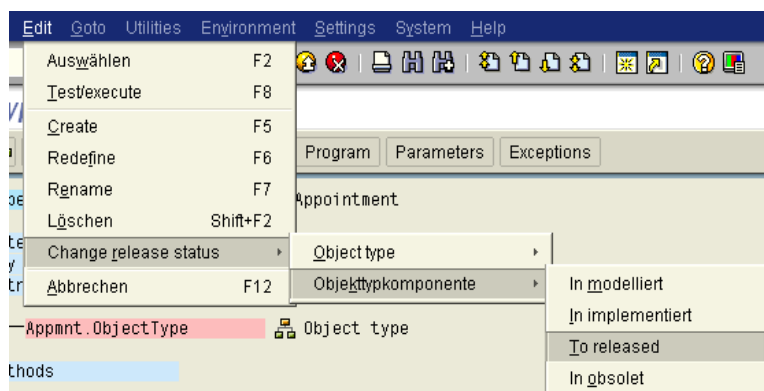


Menu Path:

Edit->Change release status->Object type->To released



Some information popups may occur, however, the BO should become “Released” as indicated by the “tick” graphic, next to the BO description.

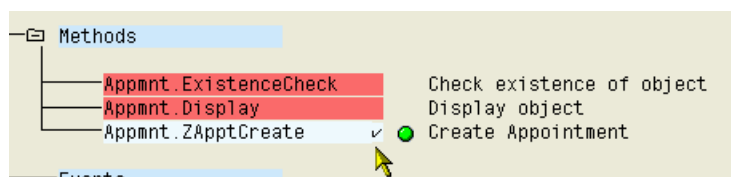


Now the BAPI needs to be released.

Click once on the BAPI.

Then “Release” the BAPI using the menu path

Edit->Change release status->Object component->To released



The BAPI should now be “Released” as indicated by the “Tick” graphic icon, next to the BAPI.

“Generate” the BO, using “Ctrl + F3” or the “Generate” icon.

Short List

- Need a transportable Package
- Package does not need to have Application Component. If Package does contain an Application Component, it appears under hierarchical menu in transaction BAPI
- Implementing FM needs to be “Released”
- Implementing FM parameters need to be passed “by value”
- Implementing FM needs to be RFC
- BO needs to be “Released”
- Method needs to be “Released”

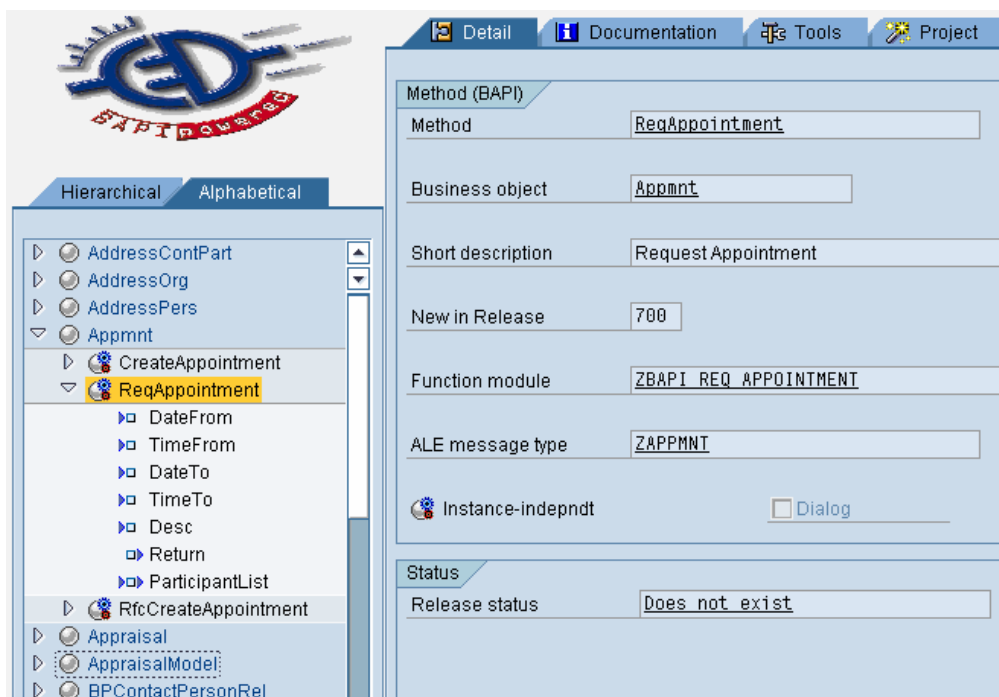
BAPI Explorer – BAPI

Further requirements need to be fulfilled for the BO to generate the BAPI-ALE interface via “BAPI Explorer” – BAPI. Specifically, the BAPI needs to be “Released”, to be visible in the “BAPI Explorer”. Some third party software also requires the BAPIs to be “Released”.

There are no consequences to the BAPI-ALE communication, whether the BAPI is “Released” or not.

Without “Releasing” the BAPI, there is no visibility of the BAPI in the “BAPI Explorer”

To “Release” the BAPI, see appendices



From the left-hand frame, “Alphabetical” tab, locate the BO and BAPI.

On the right hand frame, “Detail” tab, in the “Method (BAPI)” area, notice the “ALE message type” reads, “Does not exist”.

Double click the “Does not exist” text, and the system jumps to “Generate ALE Interface for BAPI” – BDBG, with the necessary fields populated.

From this moment on, it is identical to “Generate ALE Interface for BAPI” – BDBG.

SAP Help

From the SAP Library Help – 6.20

MySAP Technology Components

- ➔ SAP Web Application Server
 - ➔ Middleware (BC-MID)
 - ➔ Application Link Enabling (BC-MID-ALE)
 - ➔ ALE Programming Guide

Distribution Using BAPIs

Author Bio



Graduated as a mature student in 1994 with a BSc (Hons), I began work in a company programming MS Access applications. After 1 year, I began contracting as a VB, MS Access, and Excel Application Programmer in, and around, London. 3 years later, in 1997, I started a permanent job working as an SAP Technical Consultant for an SAP Implementation Partner working on various projects with an array of SAP technologies. In 2001, I decided to leap into contracting. More recently, I have extended my skills by instructing courses at SAP UK. It was teaching the BC300 – Integration Technology, which led me to write this paper. I found limited resources on the BAPI-ALE mechanism and no real hard examples.

After a small amount of time using this technology, I found the BAPI-ALE method very fast to develop and flexible throughout. I wish I could see more examples and implementation of this technology in real-life systems.

Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.