SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

Making BDBG Easy

Applies To:

Generating an ALE Interface in SAP

Summary

Transaction BDBG is used to generate an ALE interface for SAP standard or custom developed BAPIs. Once a BAPI is given an ALE interface, it can be used in LSMW for migrating data (either the BAPI or IDoc option can be selected) or just as another IDoc. Quite a few SAP standard BAPIs cannot be used in LSMW as they do not come with an ALE interface. OSS note – Documentation on Transaction BDBG (note number 125776) – details the necessary steps to use transaction BDBG but it does not comprehensively cover the technical issues that one may encounter in order to create an ALE interface.

By: Arijit Kumar Das

Company: Roads and Traffic Authority (RTA)

Date: 31 Aug 2005

Table of Contents

Making BDBG Easy

Introduction

Transaction BDBG is used to generate an ALE interface for SAP standard or custom developed BAPIs. Once a BAPI is given an ALE interface, it can be used in LSMW for migrating data (either the BAPI or IDoc option can be selected) or just as another IDoc. Quite a few SAP standard BAPIs cannot be used in LSMW as they do not come with an ALE interface. OSS note – Documentation on Transaction BDBG (note number 125776) – details the necessary steps to use transaction BDBG but it does not comprehensively cover the technical issues that one may encounter in order to create an ALE interface.

Example

BAPI_ACTIVITYCRM_CREATEMULTI, which is used to create Activities in CRM, will be used to illustrate a step-by-step approach to using transaction BDBG.

Go to Object Navigator (transaction SE80) and create a bespoke function group, we will call ours Z_CRM_ACTIVITY. Activate it.

Then go to Function Builder (transaction SE37), and copy the BAPI to a bespoke function module. Thus, BAPI_ACTIVITYCRM_CREATEMULTI is copied to Z_BAPI_ACTIVITYCRM_CREATEMULTI and assigned to the function group that we created in the previous step – Z_CRM_ACTIVITY. Edit the bespoke function module, go to the Attributes tab, and release the function module (Menu Path – Function Module >> Release >> Release). Now activate the function module.

IDoc segment types have a maximum length of 27 characters. Each associated type for the Parameters (tabs – Import, Export, Changing, and Tables) in a BAPI becomes a segment in the IDoc. Thus, an associated type in a BAPI cannot exceed 27 characters. During the ALE generation process anything beyond the 27th character gets truncated. This leads to more than 1 segment being called the same, and as a result an error occurs when the ALE interface is generated in the final step as an IDoc cannot have the same segment twice.

Eg: Table parameters DATE and DATEX correspond to types BAPIBUS20001_APPOINTMENT_INS (28 characters long) and BAPIBUS20001_APPOINTMENT_INSX (29 characters) respectively.

Also, a developer does not have the option to reduce the segment name during the ALE generation process. So in order to have an error free ALE generation happening in the end, bespoke associated types need to be created and assigned to our bespoke BAPI.

Go to ABAP Dictionary (transaction SE11), and create ZXXXXXXXXXXXXXXX data types for the structures in the BAPI that are greater than 27 characters. Activate all these bespoke structures.

Bespoke structures – ZBAPIBUS20001_APP_INS and ZBAPIBUS20001_APP_INSX can be used in lieu of the standard SAP ones for parameters DATE and DATEX. The same correction method needs to take place for other excessively long types, like the types being referenced by LOCATION and LOCATIONX.

Go to Business Object Builder (transaction SWO1). A search for *Activity* as the short description produces a few hits. After examining all the objects returned in the hit, we see that BAPI_ACTIVITYCRM_CREATEMULTI belongs to 2 Business Objects - BUS2000110 (CRM Activity) and BUS2000126 (CRM Business Activity).

Copy either of these objects to a bespoke object ZXXXXXXXXX, we will copy BUS2000110 and rename it to ZBUS200010, program RBUS2000110 should also be copied to a Z version (ZRBUS2000110), and the object name can be changed to Company Name (eg: XXX) followed by ActivityCRM (XXXActivityCRM).

Now we edit our bespoke object, and create a method for it by placing the mouse cursor on methods in the object type tree and clicking the create button on the menu bar.

A pop-up appears on the screen with the following text –

Create with function module
as template?

After clicking on Yes, another pop-up asks for a function module to be entered, enter the bespoke function module Z_BAPI_ACTIVITYCRM_CREATEMULTI here and click the tick button.

Accept all the default values in the next pop-up – Create Method: Method Properties, and do the same for the next pop-up – Create Method: Create Parameters. In the next pop-up click Yes to the question –

Method ZBAPIACTIVITYCRMCREATEMULTI not
yet implemented
Do you want to generate a template
automatically for the missing section?

Double-click on the newly created method and go to the ABAP tab, the Function Module option will be selected as the default, make the API function as the default and click OK. Generate the business object.

Now, click on the method of the object and release the method (Menu Path – Edit >> Change Release Status >> Object Type Component >> To Released). Double-click on the method to confirm that the Status is released.

Save all the changes made, and Back out of the Edit mode and go the main Business Object Builder screen. Re-generate the business object. First implement, then release the main object ZBUS200010 (Menu Path – Object Type >> Change Release Status To >> Implemented, then follow the same menu path and click on Released).

Go to the BAPI Explorer (transaction BAPI), click on the filter icon on the menu bar, and select radiobutton – All. Click on the alphabetical tab. Look for XXXActivityCRM in the list. Release status should be Released for this object.

Now we should be able to generate the ALE interface for the BAPI. Execute transaction BDBG, enter the bespoke object (ZBUS200010), do a drop down on the method field and select the bespoke method from the list. Click on the create button on the menu bar. In the first pop-up accept Message Type ZBAPIACTIVITYCRM and click OK. In the next pop-up the following values should be there by default –

IDoc type – ZBAPIACTIVITYCRM01,

ALE Outbound Processing
Function Module – ZZ_ALE_ACTIVITYCRM_CREATEMULTI,
Function Group – Z_CRM_ACTIVITY, and

ALE Inbound Processing
Function Module – ZIDOC_INPUT_ZBAPIACTIVITYCRM,
Function Group – Z_CRM_ACTIVITY.

The option – Call in update task – should be ticked.

As the final step, we need to release the IDoc and the segments (Menu Path – Edit >> IDoc Type and Segment Type >> Release).

Now our BAPI has an ALE Interface to it and can be used in conjunction with either the BAPI or IDoc option in LSMW. The default values for an IDoc Inbound Processing should be maintained in order to use the bespoke BAPI / IDoc.

Author Bio

I have been an ABAP programmer for the past 4.5 years. I started my career in SAP with Toyota, Melbourne as a final year Bachelor's work experience student. After staying with Toyota for close to 4 months, I got my first big break by scoring a full-time position with BOC in Sydney. Here, main duties were programming and configuring in the logistics area – SD, MM, FI-CO, PP, and PM. I also got a chance to dabble in HR and PS. I remained with BOC for a little more than 3 years. Then I moved on (to greener pastures), and joined the RTA in Sydney (Roads and Traffic Authority). In the 6 months that I have been with the RTA, I have had the good fortune (or misfortune) to dirty my hands with customizing (read hacking) standard HR applications. Currently, I'm involved in the RTA's CRM 4 implementation. My main areas of responsibility are to configure various bits of the system for Business Partners, Products, and Activities.