Applies to:
SAP CRM 7.0. For more information, visit the Customer Relationship Management homepage.

Summary
This document is aimed at understanding how to create custom BOL object and assign it as a relationship to BP. The document will also look at how to implement GenIL class for the custom BOL object.

Author: Ashish Walke
Company: Deloitte Consulting India Pvt. Ltd.
Created on: 5 August, 2011

Author Bio
Ashish Walke is a Consultant at Deloitte consulting. He has worked on multiple implementations in SAP CRM across versions.
Table of Contents

Extend object Model for Business Partner ................................................................. 3
  • Create custom BOL object and assign it to BOL object – BulHeader ............................. 3
  • Create new entry for custom BOL object (ZCUSTOM) .................................................... 3
  • Assign attribute structure and key structure for BOL object ........................................ 4
  • Create relationship with BOL object BulHeader .......................................................... 4

Create BOL structures and custom table. ........................................................................ 5
  • Attribute structure. ........................................................................................................ 5
  • Key structure .................................................................................................................. 5
  • Custom table .................................................................................................................. 6

BOL Model for BP ........................................................................................................... 7

Implement GenIL class for BOL object ZCUSTOM ........................................................... 8
  • Create class – ZCL_CUSTOM_BOL ............................................................................... 8
  • Interaction between CRM UI and GenIL class methods .................................................. 9
  • Handle data READ ....................................................................................................... 9
  • Handle data MODIFY .................................................................................................. 11
  • Handle data SAVE ...................................................................................................... 13
  • Clear global buffers ..................................................................................................... 13

Related Content ............................................................................................................. 14

Disclaimer and Liability Notice ........................................................................................ 15
Extend object Model for Business Partner

- Create custom BOL object and assign it to BOL object – BuiltHeader

SPRO -> SAP Reference IMG -> Customer relationship management -> CRM cross application components -> Generic Interaction layer/Object layer -> Component-Specific settings -> Extend object model for Business partner

Display IMG

- Create new entry for custom BOL object (ZCUSTOM)

(Choose Implementation class from Dialog structure on left hand pane and click on New Entries)

Change View "Implementation Class Definition": Overview
• **Assign attribute structure and key structure for BOL object**

Every BOL object can have:
- An attribute structure which contains all fields related to that BOL object.
- A key structure to uniquely identify entities.

*(Choose Object definition from Dialog structure on left hand pane and click on New Entries)*

**New Entries: Details of Added Entries**

![Diagram showing object definition with attribute and key structure]

- **Create relationship with BOL object BuilHeader**

BOL object BuilHeader corresponds to BP header data. Hence, we create a relation between BuilHeader and ZCUSTOM. For every BP we may or may not have custom data hence we maintain the cardinality as 1 : 0..n.

*(Choose Model definition from Dialog structure on left hand pane and click on New Entries)*

**New Entries: Overview of Added Entries**

![Diagram showing model definition with relation between BuilHeader and ZCUSTOM]

Save all the changes made so far.
Create BOL structures and custom table.

- **Attribute structure.**
  The attribute structure defines all fields for custom BOL object – ZCUSTOM.

**Dictionary: Maintain Structure**

<table>
<thead>
<tr>
<th>Component</th>
<th>RTx</th>
<th>Component type</th>
<th>Data Type</th>
<th>Length</th>
<th>Decim</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP GUID</td>
<td></td>
<td>BU PARTNER GUID</td>
<td>RAW</td>
<td>16</td>
<td></td>
<td>Business Partner GUID</td>
</tr>
<tr>
<td>PROD GUID</td>
<td></td>
<td>COMT PRODUCT GUID</td>
<td>RAW</td>
<td>16</td>
<td></td>
<td>Internal Unique ID of Product</td>
</tr>
<tr>
<td>PROD ID</td>
<td></td>
<td>COMT PRODUCT ID</td>
<td>CHAR</td>
<td>49</td>
<td></td>
<td>Product ID</td>
</tr>
<tr>
<td>VALID FROM</td>
<td></td>
<td>COMT BSP TIME SC</td>
<td>DATS</td>
<td>8</td>
<td></td>
<td>Start Date for Rule Period</td>
</tr>
<tr>
<td>VALID TO</td>
<td></td>
<td>COMT BSP TIME SC</td>
<td>DATS</td>
<td>8</td>
<td></td>
<td>End Date for Rule Period</td>
</tr>
</tbody>
</table>

- **Key structure**
  The key structure defines the key fields that will be used to uniquely identify entities. The key fields match with the first two fields of attribute structure.

Note: Key structure may consist of one or multiple fields depending on the scenario.
• **Custom table**

The BOL object ZCUSTOM would be assigned to an assignment block in Account overview page. Since it may contain multiple entries we would create a custom table for the same. In this example we are storing multiple product data for an account (BP).

### Dictionary: Maintain Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Key</th>
<th>Data element</th>
<th>Data Type</th>
<th>Length</th>
<th>Decim</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYMandi</td>
<td>✓</td>
<td>SYMANDI</td>
<td>CLNT</td>
<td>3</td>
<td>0</td>
<td>Client ID of Current User</td>
</tr>
<tr>
<td>BP GUID</td>
<td>✓</td>
<td>BU_PARTNER GUID</td>
<td>RAW</td>
<td>16</td>
<td>0</td>
<td>Business Partner GUID</td>
</tr>
<tr>
<td>PROD GUID</td>
<td>✓</td>
<td>COMT_PRODUCT GUID</td>
<td>RAW</td>
<td>10</td>
<td>0</td>
<td>Internal Unique ID of Product</td>
</tr>
<tr>
<td>PROD ID</td>
<td></td>
<td>COMT_PRODUCT ID</td>
<td>CHAR</td>
<td>48</td>
<td>0</td>
<td>Product ID</td>
</tr>
<tr>
<td>VALID FROM</td>
<td></td>
<td>COMT_BSP_TIME SC</td>
<td>DATS</td>
<td>8</td>
<td>0</td>
<td>Start Date for Rule Period</td>
</tr>
<tr>
<td>VALID TO</td>
<td></td>
<td>COMT_BSP_TIME SC</td>
<td>DATS</td>
<td>8</td>
<td>0</td>
<td>End Date for Rule Period</td>
</tr>
</tbody>
</table>
BOL Model for BP

In transaction GENIL_MODEL_BROWSER, under Component set BP_APPL, we can view the custom BOL object ZCUSTOM related to standard BOL object BuilHeader.

Note: Expand node BuilHeader and go to Relations. The above screenshot has been adjusted in order to move the BOL object ZCUSTOM to the top of the BOL hierarchy. In the actual scenario the custom BOL object would be the last relation displayed i.e. after all standard BOL relations.
Implement GenIL class for BOL object ZCUSTOM

- **Create class – ZCL_CUSTOM_BOL**

The class ZCL_CUSTOM_BOL will handle data transfer between BOL and Database layer.

While creating this class, specify the super class as CL_BUIL_ABSTR which contains methods READ, MODIFY, EXECUTE_WITH_SAVE and EXECUTE_WITH_INIT that shall be re-defined.

Define a Global internal table to store custom BOL data.

*We will see the functionality of these methods in detail in subsequent steps*
• Interaction between CRM UI and GenIL class methods

Interaction diagram:
- Read:
  - Get key
  - Identify new/edit/delete based on DELTA_FLAG
  - New/Edit
  - Delete
- Modify:
  - Update changes to Global internal table
  - Remove entry from Global internal table
- Execute with save:
  - Save data from Global internal table to database
- Execute with init:
  - Clear Global internal table and any other Global buffers

• Handle data READ

Redefine method READ from CL_BUIL_ABSTR. This method would be called each time the Account overview page is loaded. The logic in this method would be structured as follows:-

- Assign importing data to object.

  **Sample code:**
  ```
  DATA: lv_obj TYPE REF TO if_genil_container_object.
  TRY.
  lv_obj ?= iv_ref.
  CATCH: cx_sy_assign_cast_error.
  * else exit: should never happen
  EXIT.
  ENDSAMPLE.
  ```

  Note: LV_OBJ refers to custom BOL object – ZCUSTOM.

- Check if attributes were requested. If not, then exit.

  **Sample code:**
  ```
  CHECK lv_obj->check_attr_requested( ) = abap_true.
  ```

- Fetch key for current BOL entity

  **Sample code:**
  ```
  DATA: lwa_key TYPE ZCUSTOM_KEY.
  lv_obj->get_key( IMPORTING es_key = lwa_key ).
  ```
- **If key is not found** that means data has to be fetched from database (Table ZCUSTOM_TABLE). Loop through the data and store it in the object. Fetch data from first record and set it as key.

```plaintext
Sample code:-
LOOP AT lt_obj1 INTO lwa_obj1 FROM 2.
  MOVE-CORRESPONDING lwa_obj1 TO lwa_key.

  MOVE-CORRESPONDING lwa_obj1 TO lwa_global_bol_data.

  TRY.
    lv_obj_new = lv_obj->copy_self_with_structure( is_object_key = lwa_key ).

    CATCH cx_crm_genil_model_error.           "#EC NO_HANDLER
      CATCH cx_crm_cic_duplicate_entry.        "#EC NO_HANDLER
    ENDTRY.
  ENDDO.
ENDLOOP.

READ TABLE lt_obj1 INTO lwa_obj1 INDEX 1.
IF sy-subrc EQ 0.
  MOVE-CORRESPONDING lwa_obj1 TO lwa_key.

  TRY.
    lv_obj->set_key( lwa_key ).

    CATCH cx_crm_genil_model_error.           "#EC NO_HANDLER
      CATCH cx_crm_cic_duplicate_entry.        "#EC NO_HANDLER
    ENDTRY.
  ENDDO.
ENDIF.
```

Assumption: Before processing the LOOP the data from table ZCUSTOM_TABLE has been fetched into internal table LT_OBJ1 based on key values (LWA_KEY).

- **If key is found** then fetch data from global internal table and set attributes.

```plaintext
Sample code:-
READ TABLE gt_global_bol_data INTO lwa_global_bol_data
WITH KEY bu_guid = lwa_key-bu_guid
  prod_guid = lwa_key-prod_guid.
IF sy-subrc NE 0.
  CLEAR lwa_global_bol_data.
ENDIF.

  MOVE-CORRESPONDING lwa_global_bol_data TO lwa_bol_data.
  lv_obj->set_attributes( lwa_bol_data ).
  me->set_attr_properties( lv_obj ).
```

Note: The global internal table (GT_GLOBAL_BOL_DATA) is declared as an attribute in class ZCL_CUSTOM_BOL and is filled in the MODIFY method.
Handle data MODIFY

Redefine method MODIFY from CL_BUIL_ABSTR. This method would be called while navigating back from the “Edit / New” page. The logic in this method would be structured as follows:

- Assign importing data to object. (Refer to sample code above for method READ)
- Fetch key for current entity. (Refer to sample code above for method READ)
- Fetch Delta flag. The delta flag would enable us to determine whether the operation was NEW / EDIT / DELETE.

Sample code:-

```
DATA: lv_delta TYPE crmt_delta.
lv_delta = lr_obj->get_delta_flag( ).

IF NOT lv_delta IS INITIAL.
  * Delete messages for entity
    lv_msg = lr_obj->get_message_container( ).
    lv_msg->delete_messages( iv_object_name = me->object_name ).
ENDIF.
```

- If current operation is creation i.e. NEW then fetch data entered on screen and set the key. Also, store the data into global internal table.

Sample code:-

```
IF lv_delta EQ if_genil_cont_simple_object=>delta_created.
  *
  Set the key structure
    lwa_key-bu_guid = lv_partner_guid.
    lwa_key-prod_guid = lv_prod_guid.
TRY.
  CALL METHOD lr_obj->set_key EXPORTING
    is_object_key = lwa_key.
  CATCH cx_crm_genil_duplicate_key.
    #EC NO_HANDLER
ENDTRY.
  *
  Store data into global internal table
    lr_obj->get_attributes( IMPORTING es_attributes = lwa_bol_data ).
    APPEND lwa_bol_data to gt_global_bol_data.
ENDIF.
```

- If current operation is change i.e. EDIT then fetch the modified fields and update the change(s) into global internal table.
Sample code:-
DATA: lr_obj_prop TYPE REF TO if_genil_obj_attr_properties,
lt_changed_fields TYPE crmt_attr_name_tab.

IF lv_delta EQ if_genil_cont_simple_object=>delta_changed.
  lr_obj->get_attributes( IMPORTING es_attributes = lwa_bol_data_chg ).

  lr_obj_prop = lr_obj->get_attr_props_obj( ).
  CALL METHOD lr_obj_prop->get_name_tab_4_property
  EXPORTING
    iv_property = if_genil_obj_attr_properties=>modified
  IMPORTING
    et_names    = lt_changed_fields.

  CLEAR lwa_global_bol_data.
  READ TABLE gt_global_bol_data INTO lwa_global_bol_data
  WITH KEY bu_guid   = lwa_key-bu_guid
              prod_guid = lwa_key-prod_guid.
  IF sy-subrc EQ 0.
    LOOP AT lt_changed_fields INTO lwa_changed_fields.
      CASE lwa_changed_fields.
        WHEN 'VALID_FROM'.
          lwa_global_bol_data-valid_from = lwa_bol_data_chg-valid_from.
        WHEN 'VALID_TO'.
          lwa_global_bol_data-valid_to    = lwa_bol_data_chg-valid_to.
        WHEN OTHERS.
          "do nothing"
        END_CASE.
      END_LOOP.
  ELSE.
    MOVE-CORRESPONDING lwa_bol_data_chg TO lwa_global_bol_data.
  ENDIF.

  APPEND lwa_global_bol_data to gt_global_bol_data.
ENDIF.

In order to activate the “Save” button on account overview page we need to register the save handler.

Sample code:-
DATA: lwa_obj_inst TYPE crmt_genil_obj_instance,
me->register_save_handler( lr_obj ).
  lwa_obj_inst-object_name = me->object_name.
  lwa_obj_inst-object_id   = cl.crm_genil_container_tools=>build_object_id( lwa_key ).
  APPEND lwa_obj_inst TO ct_changed_objects.
If there are errors in processing then transfer them to screen.

**Sample code:**

```plaintext
DATA : lt_return TYPE bapiret2_t.
CALL METHOD cl_crm_buil_services=>bol_add_messages
  EXPORTING
    iv_cont_obj = lr_obj
    iv_object_name = me->object_name
    it_bapi_messages = lt_return.
```

- **Handle data SAVE**
  Redefine method EXECUTE_WITH_SAVE from CL_BUIL_ABSTR. This method would be called on clicking of “Save” button on Account overview page. Data can be saved to database either with a standard BAPI or custom Function module. Direct interaction with the database layer would be performed in this method.

- **Clear global buffers**
  Redefine method EXECUTE_WITH_INIT from CL_BUIL_ABSTR. This method would be called when Account overview page is loaded.
Related Content

www.sdn.sap.com

Technical Data Environment

SAP CRM WebUI blogs

For more information, visit the Customer Relationship Management homepage.
Disclaimer and 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.