SAP Records Management

Automatically Inserting Business Objects in Records Management Records and Setting Up Navigation Between Business Objects and Records

May 7, 2004

Contents

1 Introduction

The following tutorial is aimed at consultants who are implementing Records Management. Prerequisites are a working knowledge of Records Management and ABAP objects.

Customers often want records to be filled automatically, to save users extra work. One typical requirement is for business objects to be inserted in records automatically. This tutorial uses a concrete example to show you how to meet this requirement in a user-friendly way.

Take the following scenario: When you create an instance of the *Sales Order* business object type, you want this sales order to be inserted in the corresponding customer record automatically. You then want the option of navigating to the customer record from the interface of the sales order.

This can be realized as follows: When the sales order is created, the *SalesOrder.created* event is triggered. We want to use this event to enable the automatic insertion of the business object in the appropriate customer record. At the same time, we want to use the generic object services to create a link between the sales order and the customer record. This link enables the user to navigate directly to the record from the sales order.

2 Prerequisites

- You have created an element type for record models.
- You have created a record model in which you have integrated a model node for the business objects.
- You have created a content model for records and have created a CUSTOMER_NO (or similar) attribute for this content model. You have typed this attribute according to the field in the business object (for *SalesOrder*, this is the KUAGV- KUNNR field).
- You have created an element type for records. You have entered the new record model for the connection parameter MODEL_ID; you have entered the new content model for the connection parameter DOCUMENT_CLASS.

3 Setting Up the Event-Receiver Linkage

When you link an event and a receiver, you specify which event causes a reaction for which business object.

You link events and receivers in transaction SWETYPV (from the SAP Easy Access screen, choose *Tools* → *Business Workflow* → *Development* → *Definition Tools* → *Events* → *Event Linkages* → *Type Linkages*). Make the following entries:

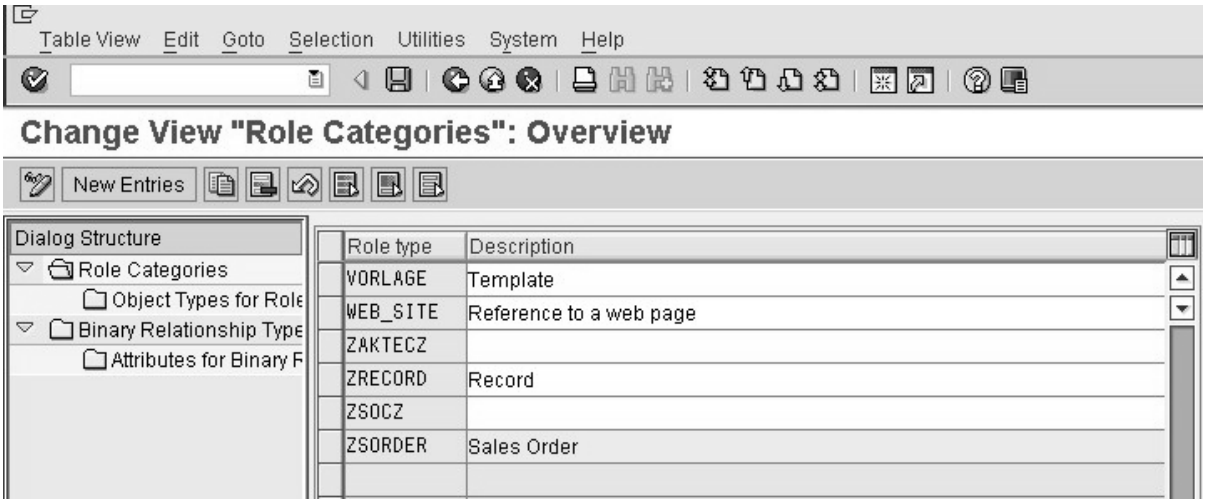
Object Category	BOR object type
Object Type	BUS2032
Event	CREATED
Receiver call	Function module
Receiver Function Module	Name of a function module that executes the reaction to the event The receiver function module must be created as a copy of the template function module SWE_TEMPLATE_REC_FB. This function module is included in the function group SWE_TEMPLATE. The interface is described in the documentation about the template function module. The *Remote-Enabled Module* flag must be set in the attributes of the receiver function module.
Check Function Module	Optional: name of a function module that you need to develop

	You enter a check function module to decide whether the receiver function module needs to be called. You can make use of the data in the event container. If an exception is triggered when a check function module is executed, then the event is not linked to the receiver, and the receiver function module is not executed. The check function module must be created as a copy of the template function module SWE_TEMPLATE_CHECK_FB. This function module is included in the function group SWE_TEMPLATE. The interface is described in the documentation about the template function module.
Linkage Activated	Activate the flag.

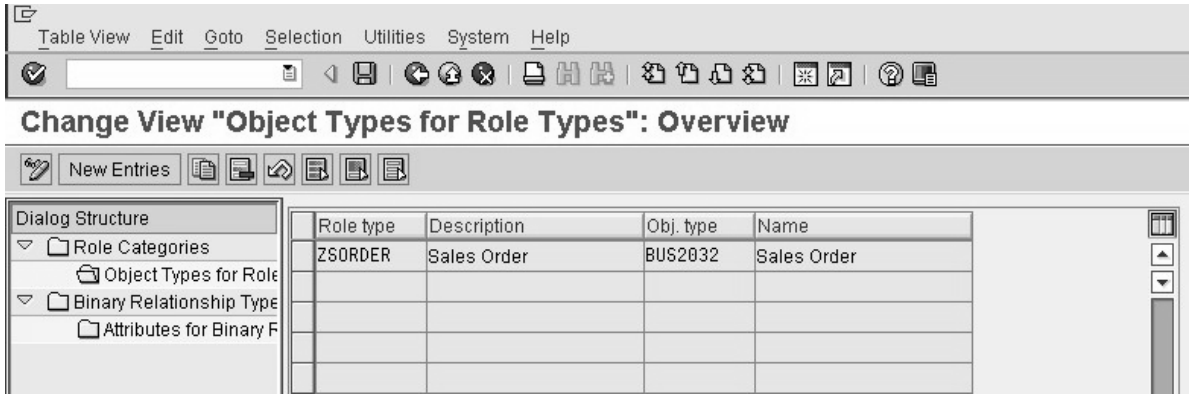
4 Setting Up the Linkage Service for the Generic Object Services

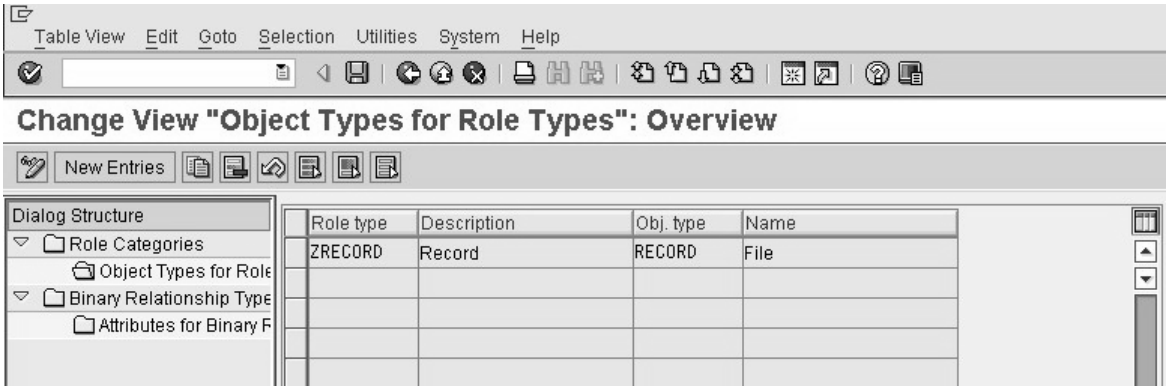
Before you can use the generic object services to link business objects (in our case, *Sales Order* and *Customer Record*), y ou must make some Customizing settings. You make these settings in the VRBINRELATION view cluster in transaction SM34. Proceed as follows:

1) Create two role types for the business object types that you want to link.



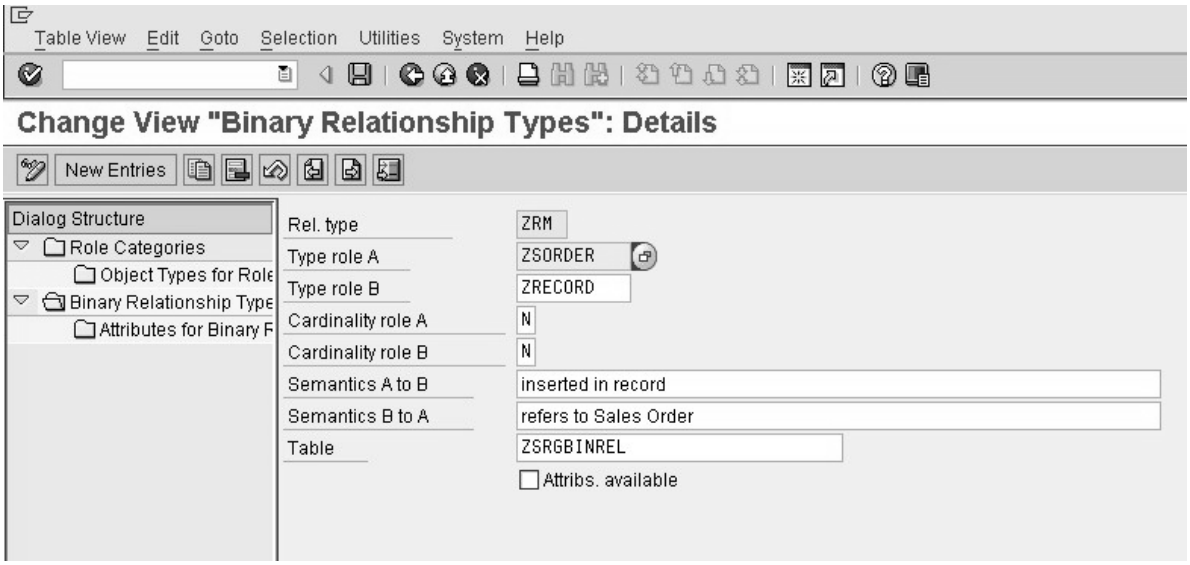
2) Assign a business object type to each of the role types.





3) In transaction SE11, create a table for storing the linkage information between the business objects of both role types. Create this new table as a copy of the SRGBINREL table.

4) In the VRBINRELATION view cluster, create a binary relationship type for creating the relationship between the business object types. In this case, objects of the type BUS2032 are linked to objects of the type RECORD. In the *Table* field, enter the table you created in step 3.



For more information, see the SAP Library under Relationship Service (BC-SRV-GBT).

5 Implementing the Receiver Function Module

Copy the template function module SWE_TEMPLATE_REC_FB and give it a new name. The following documents the steps that you need to implement.

5.1.1 Finding the Correct Record

To search for the correct record, use the function module BAPI_RECORD_GETLIST. The correctness of the record is determined by the customer number, which is a unique attribute. (You must already have created this attribute in the content model of the record; see above under *Prerequisites*.)

You can determine the customer number as follows:

The import parameter OBJKEY of your receiver function module gives you the document number of the sales order. You can use this information to instantiate the object and extract additional information. The

BOR provides you with a range of macros for this. To use this macros, you must integrate the `<cntain>` include.

In this scenario, you use the following macros:

- swc_create_object (You instantiate the object with the type BUS2032.)
- swc_get_property (You get the object-type attribute *OrderingParty*.)
- swc_get_object_key (You get the *Customer Number* key from the *Customer* object.)

For more information about these macros, see the SAP Library under *SAP NetWeaver Components → SAP Web Application Server → Business Management → WebFlow Engine → Reference Documentation → Business Object Builder → Programming in the Implementation Program → Macro Instructions for Accessing Objects, Attributes and Methods*.

As well as the table with the search parameters, the BAPI_RECORD_GETLIST interface also requires the RMS ID and SPS ID of the record you want to find. You must define this information and specify it in the code.

5.1.2 Inserting the Business Object

To insert the business object, use the SRM_RECORD_ADDELEMENT function module. The SRM_RECORD_ADDELEMENT function module wraps the BAPI_RECORD_ADDELEMENT function module; unlike the BAPI, however, it raises exceptions. You require these exceptions, since you are calling the function module in background mode. Here, you cannot extract a return structure, but you can register exceptions.

Calling the function module in background mode is important in those cases where the record is locked. Therefore, call the SRM_RECORD_ADDELEMENT function module with the IN BACKGROUND TASK addition. If the record is locked, the `container_is_locked` exception is raised, and the failed call is recorded in transaction SM58, with the error message. You can call the function module again later; to do this, choose *Edit → Execute LUW* in transaction SM58. (We recommend that you schedule a regular background job for this.) Note: After calling the IN BACKGROUND TASK function module, you must execute a COMMIT WORK.

When you call SRM_RECORD_ADDELEMENT, you must specify the following parameters to identify the element you want to insert:

- SP POID table: You can specify the information for the SP POID of the business object in the following ways:
  - SP POID parameter BOR_OBJECT_TYPE: You get this value from the OBJTYPE import parameter of your receiver function module.
  - SP POID parameter OBJECT_ID: You get this value from the OBJKEY import parameter of your receiver function module.
- SPS_ID (element type of the business object that you want to insert): You specify this in the code.
- ANCHOR (anchor in the record model the business object that you want to insert): You specify this in the code.

You must also specify the following parameters. They are used to identify the record:

- OBJECTID: You get this value from the RESULTING_LIST parameter (OBJECTID field) that is imported when BAPI_RECORD_GETLIST is called.
- DOCUMENTCLASS: You get this value from the RESULTING_LIST parameter (DOCCLASS field) that is imported when BAPI_RECORD_GETLIST is called.

5.2 Linking the Sales Order with the Record

To create a concrete link between two business objects whose object types have a relationship model, you must execute the BINARY_RELATION_CREATE function module. You need to specify the business object type and the object key for the interface parameters OBJ_ROLEA and OBJ_ROLEB. For the

RELATIONTYPE interface parameter, you need to specify the relationship type defined in the VRBINRELATION view cluster. After the function module has been called, a COMMIT WORK must be executed.

5.3 Example Code

The following is an example of the code for the receiver function module.

Notes:

- The example does not include the handling of exceptions; you must add this yourself.
- This example has been implemented in WebAS 6.20, and is only guaranteed to be valid for this release.

```
FUNCTION zrm_bo_created_eventconsumer .
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(EVENT) LIKE  SWETYPECOU-EVENT
*"     VALUE(RECTYPE) LIKE  SWETYPECOU-RECTYPE
*"     VALUE(OBJTYPE) LIKE  SWETYPECOU-OBJTYPE
*"     VALUE(OBJKEY) LIKE  SWEINSTCOU-OBJKEY
*"     VALUE(EXCEPTIONS_ALLOWED) LIKE  SWEFLAGS-EXC_OK DEFAULT SPACE
*"  EXPORTING
*"     VALUE(REC_ID) LIKE  SWELOG-RECID
*"  TABLES
*"      EVENT_CONTAINER STRUCTURE  SWCONT
*"  EXCEPTIONS
*"      TEMP_ERROR
*"      ANY_ERROR
*"----------------------------------------------------------------------
  INCLUDE <cntain>.

  DATA: lt_property_selection  TYPE TABLE OF bapipropqy,
        lwa_property_selection TYPE bapipropqy,
        lt_resulting_list      TYPE TABLE OF bapidoctab,
        lwa_resulting_list     TYPE bapidoctab,
        l_anchor               TYPE bapisrmrec-anchor,
        lt_element_sp_poid     TYPE TABLE OF bapiproptb,
        lwa_element_sp_poid    TYPE bapiproptb,
        l_return               TYPE bapiret2,
        lo_bus2032             TYPE swc_object,
        lo_kna1                TYPE swc_object,
        l_customer_no          LIKE kna1-kunnr,
        lwa_obj_rolea          TYPE boriden t,
        lwa_obj_roleb          TYPE boriden t,
        l_record_objkey        TYPE swo_typeid.

** get customer no by application document no
  swc_create_object lo_bus2032 'BUS2032' objkey.
  swc_get_property lo_bus2032 'OrderingParty' lo_kna1.
  swc_get_object_key lo_kna1 l_customer_no.

** set property table
  lwa_property_selection-propname = 'ZRM_CUSTOMER_NO'.
  lwa_property_selection-option = 'EQ'.
  lwa_property_selection-sign = 'I'.
  lwa_property_selection-propval_lo = l_customer_no.
  APPEND lwa_property_selection TO lt_property_selection.

** retrieve correct record
  CALL FUNCTION 'BAPI_RECORD_GETLIST'
    EXPORTING
      rms_id                  = 'S_RMS_DEMO'
      sps_id                  = 'ZRM_DEMO_RECORD'
*     MAX_HITS                =
```

```
   IMPORTING
     return                  = l_return
   TABLES
     property_selection      = lt_property_selection
     resulting_list          = lt_resulting_list.

** set sp poid of BOR object
  READ TABLE lt_resulting_list INTO lwa_resulting_list INDEX 1.

  lwa_element_sp_poid-name = 'BOR_OBJECT_TYPE'.
  lwa_element_sp_poid-value = objtype.
  APPEND lwa_element_sp_poid TO lt_element_sp_poid.

  lwa_element_sp_poid-name = 'BOR_OBJECT_ID'.
  lwa_element_sp_poid-value = objkey.
  APPEND lwa_element_sp_poid TO lt_element_sp_poid.

** insert BOR object in record
  CALL FUNCTION 'SRM_RECORD_ADDELEMENT'
    IN BACKGROUND TASK
    EXPORTING
      objectid              = lwa_resulting_list-objectid
      documentclass         = lwa_resulting_list-docclass
      sps_id                = 'ZRM_DEMO_CUSTOMER_ORDER'
      anchor                = 'ZRM_DEMO_CUSTOMER_ORDER'
    IMPORTING
      return                = l_return
    TABLES
      element_sp_poid       = lt_element_sp_poid
    EXCEPTIONS
      anchor_not_found      = 1
      not_authorized        = 2
      parameter_error       = 3
      container_not_found   = 4
      container_is_locked   = 5
      max_number_of_elements = 6
      poid_is_wrong         = 7
      internal_error        = 8
      OTHERS                = 9.

  COMMIT WORK.

**write connection between SalesOrder and Record for navigation
  lwa_obj_rolea-objkey = objkey.
  lwa_obj_rolea-objtype = objtype.

  CONCATENATE lwa_resulting_list-docclass
              lwa_resulting_list-objectid
              INTO l_record_objkey
              SEPARATED BY space.
  lwa_obj_roleb-objkey = l_record_objkey.
  lwa_obj_roleb-objtype = 'RECORD'.

  CALL FUNCTION 'BINARY_RELATION_CREATE'
    EXPORTING
      obj_rolea      = lwa_obj_rolea
      obj_roleb      = lwa_obj_roleb
      relationtype
      ='ZRM'
    EXCEPTIONS
      no_model       = 1
      internal_error = 2
      unknown        = 3
      OTHERS         = 4.

  COMMIT WORK.
```

```
ENDFUNCTION.
```

6 Testing the Scenario

6.1 Inserting the Sales Order

1) Create a record for the element type that is based on the content model for which you created the CUSTOMER_NO attribute.

2) In the record, give the CUSTOMER_NO attribute a customer number.

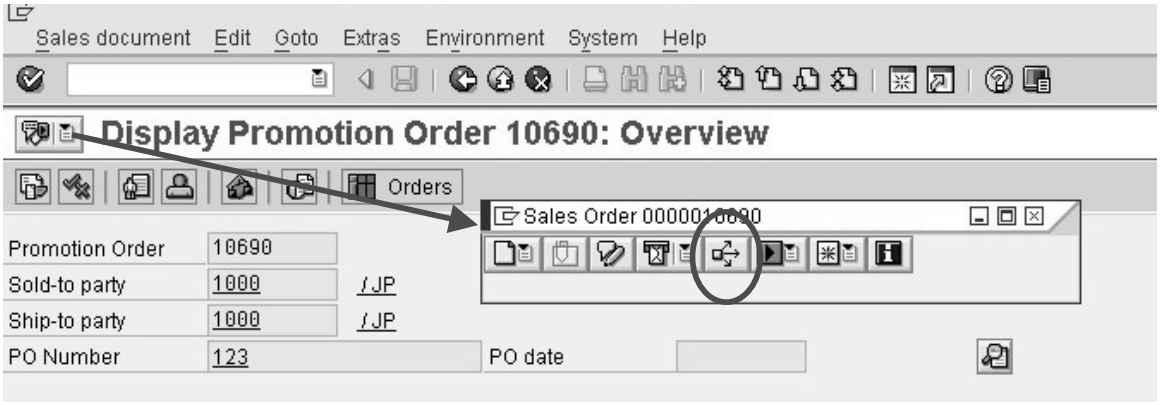
3) Create a sales order (use transaction VA01). Enter the customer number that you entered in the record in step 2.

4) Open the record. The business object has now been inserted.

6.2 Navigation

1) Open the sales order that you have just created (use transaction VA03).

2) Open the toolbox with the generic object services. (If the generic object services are not active, see SAP Note 598073.)



3) Go to the where-used list. A list appears that shows you the record in which the sales order has been integrated.

4) Double-click the entry. The appropriate record appears.

6.3 Notes About Debugging

By default, the receiver function module is called in IN BACKGROUND TASK mod e. To be able to debug the function module, you must activate the debugging mode. To do this, set a breakpoint in the CL_SWF_EVT_STRATEGY_BOR_FB~PROCESS method; when the program runs, assign the value D to the PROCESS_MODE variable.

Alternatively, you can generate the event with transaction SWUE and set the *Trigger Receiver FM Synchr.* flag.

To determine whether an event has been triggered, you can use the event trace (transaction SWEL).