SAP Records Management

# Automatically Inserting ArchiveLink Documents in Records Management Records

May 4, 2004

# Contents

# 1 Introduction

The following tutorial is aimed at consultants who are implementing Records Management. Prerequisites are a working knowledge of Records Management, ArchiveLink, and the ABAP programming language.

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

# 2 Automatically Inserting Inbound ArchiveLink Documents

Take the following scenario: You want to archive an inbound document and assign it to a sales order. As part of this inbound ArchiveLink process, you want the ArchiveLink document to be inserted in the correct customer record automatically.

This can be realized as follows: The *SalesOrder.assigned* event is triggered when the document is assigned to the sales order. We want to use this event to enable the automatic insertion of the document in the record.

Note: The ASSIGNED event is offered by many business object types; this means that the procedure for other business object types matches the following procedure. If a business o bject type does not support the ASSIGNED event, you must implement a subtype and define the ASSIGNED event there (by inserting the IFARCH21 interface).

## 2.1 Prerequisites

- You have created an element type for record models.
- You have created a record model in whi ch you have integrated model nodes for the ArchiveLink documents. We recommend that you create one ArchiveLink element type for each ArchiveLink document type. Give the element type the same name as the ArchiveLink document type. We recommend that you crea te a separate model node for each ArchiveLink element type. Create an anchor for each of these model nodes and give this anchor the same name as the element type.
- You have created a content model for records and have created a CUSTOMER_NO (or similar) attr ibute 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.

## 2.2 Activating the Trigger for the ASSIGNED Event

In the ArchiveLink Customizing settings in transaction OAG1 (in the Implementation Guide (IMG), choose *SAP Web Application Server* → *ArchiveLink* → *Customizing Enhancements* → *Maintain All Basic Settings* ), activate the trigger of the ASSIGNED event.

## 2.3 Setting Up the Event-Receiver Linkage

When you link an event and a receiver, you specify which event causes a re action 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 | ASSIGNED |
| Receiver call | Function module |
| Receiver Function Module | Name of a function module that executes the reaction to the event<br>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<br>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.<br>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. |

## 2.4 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.

### 2.4.1 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 these 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.

### 2.4.2 Inserting the Document

To insert the ArchiveLink document, 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, b ut 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 sched ule 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 ArchiveLink document in the following ways:
  - SP POID parameter CREP -ID (content repository ID): You get this value from the ARCHIVEID parameter in the event container . You get the event co ntainer through the interface of your receiver function module (TABLES parameter EVENT_CONTAINER).
  - SP POID parameter DOC_CLASS (document type): You get this value from the DOCCLASS parameter in the event container.
  - SP POID parameter DOC_ID (document ID): Y ou get this value from the ARCHIVEDOCUMENTID parameter in the event container.
- SPS ID (element type of the ArchiveLink document that you want to insert): You get this value from the DOCUMENTTYPE parameter in the event container. The prerequisite is that yo u have given the element type of the ArchiveLink document the same name as the document type of the document (see above).
- ANCHOR (anchor in the record model for the document that you want to insert): You get this value from the DOCUMENTTYPE parameter in th e event container. The prerequisite is that you have given the model node anchor for the ArchiveLink document that you want to insert the same name as the document type of the document (see above).

You must also specify the following parameters. They are u sed 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.

### 2.4.3 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_aldoc_linked_eventconsumer .
*"----------------------------------------------------------------------
*"*"Local interface:
*"  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,
        l_sps_id               TYPE bapisrmrec-spsid,
        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_crep_id              TYPE bapipropva,
        l_doc_class            TYPE bapipropva,
        l_doc_id               TYPE bapipropva,
        l_return               TYPE bapiret2,
        lo_bus2032             TYPE swc_object,
        lo_kna1                TYPE swc_object,
        l_customer_no          LIKE kna1-kunnr.

** 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'
   IMPORTING
     return                   = l_return
   TABLES
     property_selection       = lt_property_selection
     resulting_list           = lt_resulting_list.

** get the data from event container
  LOOP AT event_container.

    CASE event_container-element.
      WHEN 'DOCUMENTTYPE'.
        l_sps_id = event_container-value.
        l_anchor = event_container-value.
      WHEN 'ARCHIVEID'.
        l_crep_id = event_container-value.
      WHEN 'DOCCLASS'.
        l_doc_class = event_container-value.
      WHEN 'ARCHIVEDOCUMENTID'.
        l_doc_id = event_container-value.
    ENDCASE.

  ENDLOOP.
```

```
** set sp poid of ArchiveLink document
  READ TABLE lt_resulting_list INTO lwa_resulting_list INDEX 1.

  lwa_element_sp_poid-name = 'CREP_ID'.
  lwa_element_sp_poid-value = l_crep_id.
  APPEND lwa_element_sp_poid TO lt_element_sp_poid.

  lwa_element_sp_poid-name = 'DOC_CLASS'.
  lwa_element_sp_poid-value = l_doc_class.
  APPEND lwa_element_sp_poid TO lt_element_sp_poid.

  lwa_element_sp_poid-name = 'DOC_ID'.
  lwa_element_sp_poid-value = l_doc_id.
  APPEND lwa_element_sp_poid TO lt_element_sp_poid.

** insert document in record
  CALL FUNCTION 'SRM_RECORD_ADDELEMENT'
    IN BACKGROUND TASK
    EXPORTING
      objectid              = lwa_resulting_list-objectid
      documentclass         = lwa_resulting_list-docclass
      sps_id                = l_sps_id
      anchor                = l_anchor
    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.

ENDFUNCTION.
```

## 2.5 Testing the Scenario

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

2) In the record, give the CUSTOMER_NO attribute a customer number for which a sales order exists in the system.

3) Save an ArchiveLink document and link it to a sales order with the customer number from step 2 . You can use the administration transaction OAAD to create the link for this test. Choose *Store and Assign*.

4) Open the record. The ArchiveLink document has now been inserted.

### 2.5.1 Notes About Debugging

By default, the receiver function module is called in IN BAC KGROUND TASK mode. 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).

# 3 Automatically Inserting Outbound ArchiveLink Documents

Take the following scenario: You want to archive an outbound document and assign it to a sales order. As part of this outbound ArchiveLink process, you want the ArchiveLink document to be inserted in the correct customer record automatically.

This is realized in exactly the same way as when you insert an inbound ArchiveLink document in the inbound process. The *SalesOrder.assigned* event is triggered when the document is assigned to the sales order. We use this event to enable the automatic insertion of the document in the record.

The outbound process for ArchiveLink documents differs depending on the application, which is why we cannot offer you an example tutorial that applies in every situation.

Note:

If you link inbound and outbound documents with the same business object, the <business object type>.ASSIGNED event is triggered in both cases. For this event, you can save exactly one receiver function module in the type -receiver linkage. This means that you must use the same receiver function module for inbound and outbound documents. However, you can insert inbound and outbound documents at different positions in the record, if you stick to the following guidelines:

Create a separate document type for inbound and outbound ArchiveLink documents. Then create two ArchiveLink element types. Give these element types the same names as the ArchiveLink document types. In the record model, create two model nodes. Assign one of the element types to each of the model nodes. Also create two anchors and give them the same names as the ArchiveLink document types. When the appropriate document type is selected as part of the ArchiveLink inbound or outbound process, the document is inserted at the right position in the record, since the position is determined by the name of the document type.