SAP
Records Management

Integration of Records Management as a GUI Control

Developer Documentation

May 13, 2004

Contents:

1 Introduction

To make the best use of this document, you require a good working knowledg e of the technical terminology and architecture of the Records Management Framework. For more information, see the Records Management reference documentation for developers.

You can implement a record or any other Records Management element as a GUI contro l in an SAP application transaction. Records Management provides the classes CL_SRM_BASE_CONTROL and CL_SRM_STACKED_CONTROL for this.

2 RM Base Control

RM base control is realized using the class CL_SRM_BASE_CONTROL. To call a Records Management element in RM base control, proceed as follows:

1. Generate the base control object using CREATE OBJECT.
2. For the attribute *SRM* of the RM base control object, call the method GET_SRM_OBJECT_FACTORY. As a returning parameter, you receive a reference to the factory object.
3. For the factory object, call the method IF_SRM_SRM_CLIENT_OBJ_FACTORY~CREATE_REQUEST. A reference to an empty request object is returned as a returning parameter.
4. Call the method IF_SRM_REQUEST~SET_DEST_POID for the request object, and enter the POID of the element to be displayed. You must inform the application in advance of the POID of the element to be displayed.
5. For the request object, call the method IF_SRM_REQUEST~SET_ACTIVITY and enter the activity that will be used to call the element.
6. For the RM b ase control object, call the method IF_SRM_BASE_CONTROL~DISPATCH_REQUEST, and enter the request object for RM base control. The element is displayed in the control.

Note: If you do not embed the control directly in the application, but use it within a serv ice provider instead (which means the service provider element contains further screen elements in addition to the control), you need to enter the interface reference to IF_SRM in the constructor. Otherwise the root object is created twice.

2.1 Example Flow

The graphic below shows an overview of an example flow when two RM base controls are called in a transaction in active mode. (Information about active and passive mode you can find in the Records Management Reference Documentation, chapter *Calling a Service Provider in Passive Mode*.)

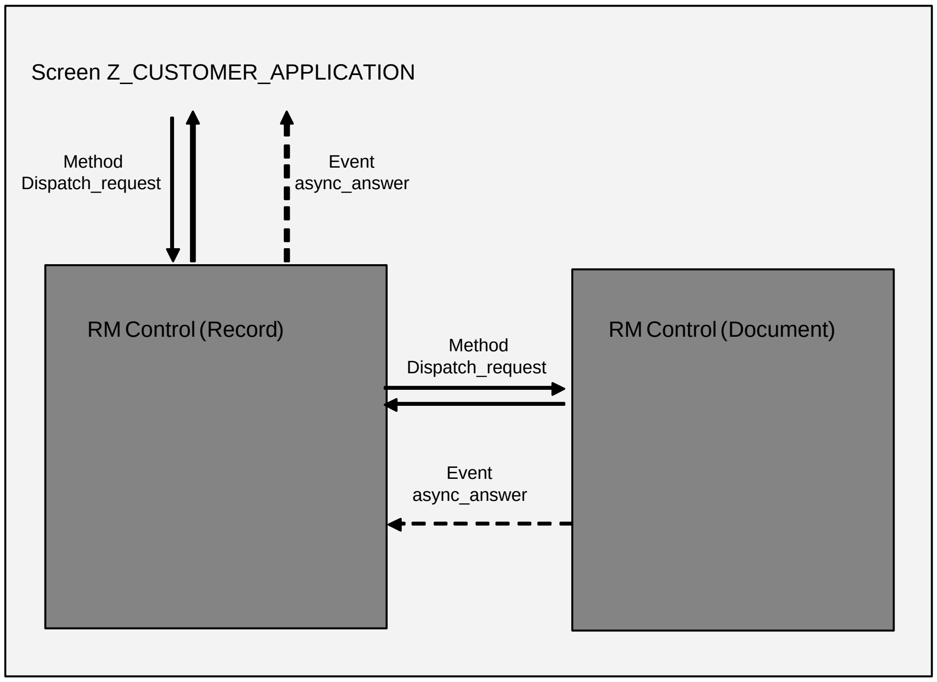


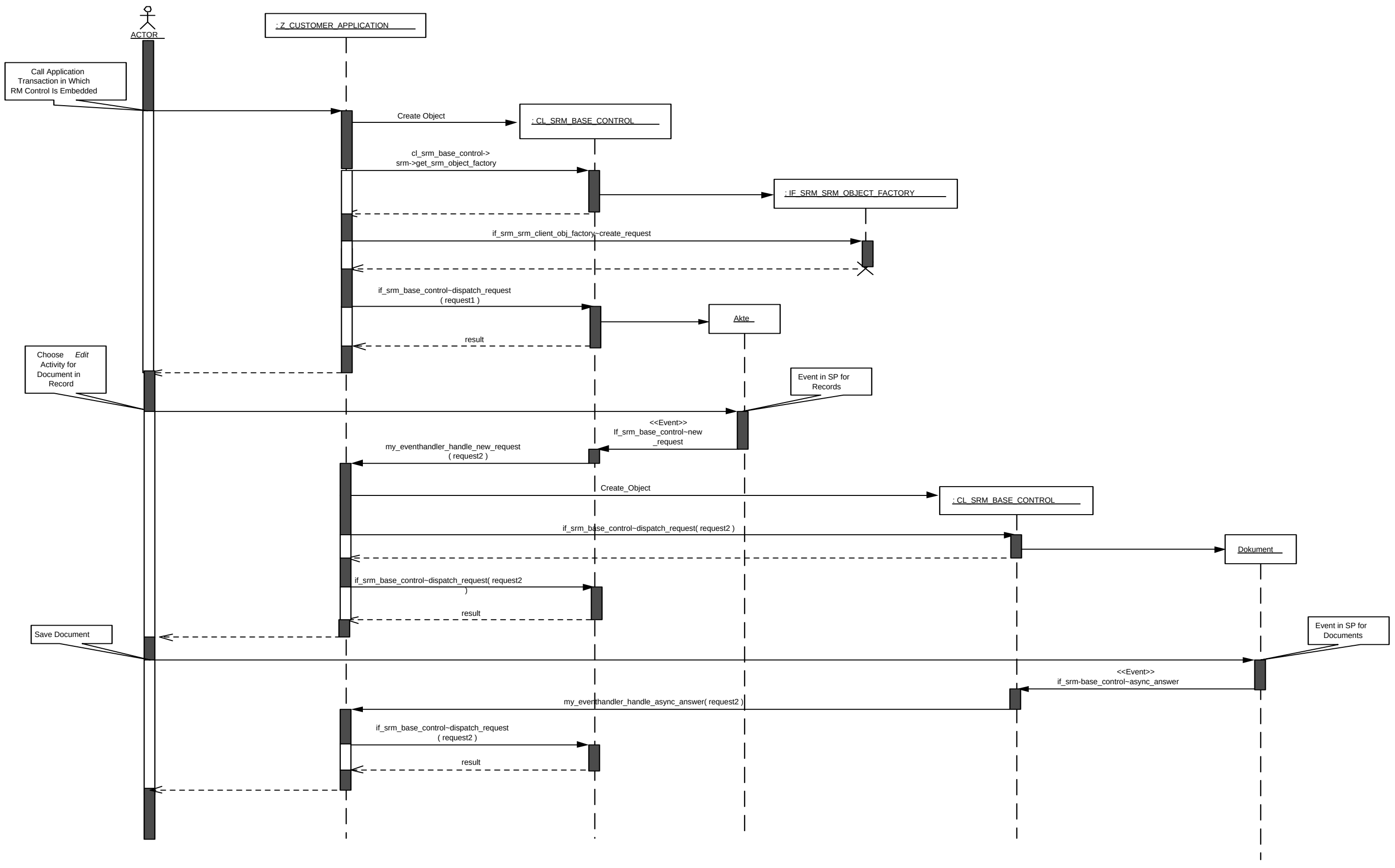
Figure 1: Integration of Two RM Base Controls in an Application Transaction

The flow diagram below illustrates this example in more detail.

In the example flow, two RM base controls are used: In the first control, a record is displayed. The second control is called if the user wants to display a document (event NEW_REQUEST). The event returns a new request object, for which the POID and the activity of the new element are set. A new control must be ge nerated in the event handler method, and the new request must be transferred to the new control using the method F_SRM_BASE_CONTROL~DISPATCH_REQUEST. The request object that is returned as a returning parameter must then be transferred to the first control (the record), to report the activity status. In the example in the flow diagram, this is an asynchronous activity (Edit). If this were a synchronous activity (for example, Display), the process would already be completed.

For asynchronous activities, the event ASYNC_ANWER is triggered when the user saves the document. The event returns the request object, for which the current POID and the current activity status of the changed document are set. In the event handler method, the request must be sent back to the first control (record).

The flow diagram shows the methods implemented by the RM base control and the RM framework, as well as the methods that you must implement as a user of the controls. The methods you implement yourself are on the “lifeline” of t he example object Z_CUSTOMER_APPLICATION.


ACTOR
: Z_CUSTOMER_APPLICATION
Call Application Transaction in Which RM Control Is Embedded
Create Object
: CL_SRM_BASE_CONTROL
cl_srm_base_control-> srm->get_srm_object_factory
: IF_SRM_SRM_OBJECT_FACTORY
if_srm_srm_client_obj_factory~create_request
if_srm_base_control~dispatch_request (request1)
Akte
result
Choose Edit Activity for Document in Record
Event in SP for Records
<<Event>> If_srm_base_control~new _request
my_eventhandler_handle_new_request (request2)
Create_Object
: CL_SRM_BASE_CONTROL
if_srm_base_control~dispatch_request(request2)
Dokument
if_srm_base_control~dispatch_request(request2)
result
Save Document
Event in SP for Documents
<<Event>> if_srm-base_control~async_answer
my_eventhandler_handle_async_answer(request2)
if_srm_base_control~dispatch_request (request2)
result

Fig

ure 1 Example Process Flow when Calling Two RM base controls

2.2 IF_SRM_BASE_CONTROL

The RM base control has two interfaces: IF_SRM_BASE_CONTROL and IF_SRM_BASE_CO NTROL_INT. The interface IF_SRM_BASE_CONTROL_INT is only for internal purposes, only the interface IF_SRM_BASE_CONTROL is relevant. This interface has the methods and events described below:

IF_SRM_BASE_CONTROL~DISPATCH_REQUEST( )

IMPORTING	IM_REQUEST	TYPE	IF_SRM_REQUEST
IMPORTING	IM_MODE_PASV	TYPE	SRMBOOLEAN
RETURNING	RE_REQUEST	TYPE	IF_SRM_REQUEST

This method executes a new request or returns a response to the SP that is in the control.

Use the flag IM_MODE_PASV to determine whether the call is active or passive. The returning parameter returns the event request object, for which you can read the activity status.

IF_SRM_BASE_CONTROL~UNLOAD_CURRENT( )

IMPORTING	IM_UNLOAD_MODE	TYPE	I
IMPORTING	IM_FORCE_UNLOAD	TYPE	SRMBOOLEAN

This method deletes the current SP from the control.

The import parameter IM_UNLOAD_MODE describes the unload mode. It can have the following values:

- IF_SRM_BASE_CONTROL=>UNLOAD_MODE_DESTROY_ONLY: The SP element is unloaded immediately.
- IF_SRM_BASE_CONTROL=>UNLOAD_MODE_NOTIFY: A mes sage is sent to the SP that unloads the element. You can use this interface to include an intermediate step before the element is unloaded. As soon as you want to actually unload the SP element, you must call the method UNLOAD_CURRENT and enter one of the other two constants.
- IF_SRM_BASE_CONTROL=>UNLOAD_MODE_NOTIFY_AND_DESTROY: A notification is sent to the SP and the SP element is unloaded immediately.

You can use the import parameter IM_FORCE_UNLOAD to determine whether or not the SP offers the user a *Cancel* option (When an SP is unloaded, it normally displays a dialog box in which it offers the user the options *Save, Do Not Save, Cancel*.)

IF_SRM_BASE_CONTROL~FINISH_ASYNC( )

IMPORTING	IM_KEEP_STATE	TYPE	SRMBOOLEAN

The method informs the SP that is in the control that the asynchronous activity should be ended, that is, the SP should save its data. It is only possible to call this method in passive mode, because in this case, the SP save procedure is triggered externally.

Use the import parameter IM_KEEP_STATE to determine whether or not the SP should be stored in a changeable state. The values have the following meanings:

- IF_SRM=>TRUE: The SP remains in change mode
- IF_SRM=>FALSE: After saving, the SP changes to display mode

IF_SRM_BASE_CONTROL~GET_REQUEST_TYPE( )

IMPORTING	IM_REQUEST	TYPE	IF_SRM_REQUEST
IMPORTING	IM_PASV_MODE	TYPE	SRMBOOLEAN
RETURNING	RE_REQUEST_TYPE	TYPE	STRING

This method returns the execution mode of a request. It is purely for information purposes.

The parameter RE_REQUEST_TYPE can have the following values:

- IF_SRM_REQUEST_PROCESSOR=>REQUEST_TYPE_INPLACE: The request is executed in -place
- IF_SRM_REQUEST_PROCESSOR=>REQUEST_TYPE_OUTPLACE: The request is executed out - place
- IF_SRM_REQUEST_PROCESSOR=>REQUEST_TYPE_NEWMODE: The request is executed in a new mode
- IF_SRM_REQUEST_PROCESSOR=>REQUEST_TYPE_NONVISUAL: The request is executed in the background or as a popup
- IF_SRM_REQUEST_PROCESSOR=>REQUEST_TYPE_INFO: Execution of the request includes displaying the information popup
- IF_SRM_REQUEST_PROCESSOR=>REQUEST_TYPE_QUERY: Execution of the request is of type Search and takes place in dialog

Event IF_SRM_BASE_CONTROL~NEW_REQUEST

IMPORTING	REQUEST	TYPE	IF_SRM_REQUEST

This event is triggered if the SP that is in the control wants to start a request, for example, if a document is displayed from a record. The event returns a request object that contains the POID to be displayed and the activity to be called. The control user must implement an event handler method that ensures this request is executed. To execute the request in a new screen, you can use the class CL_SRM_START_FRAMEWORK (for more information, see the online documentation for the class). If the request is executed in an additional RM base control, call the method IF_SRM_BASE_CONTR OL~DISPATCH_REQUEST for the new control, and enter the request.

Event IF_SRM_BASE_CONTROL~ASYNC_ANSWER

IMPORTING	REQUEST	TYPE	IF_SRM_REQUEST

This event is triggered when the SP that is in the control saves its data and then sends an asynchronous response. The event returns the request object that contains the current POID and the current activity status of the displayed element. The control user implements an event handler method that, using the method IF_SRM_BASE_CONTROL~DISPATCH_REQUEST, returns the requ est to the control from which it was started.

Event IF_SRM_BASE_CONTROL~SEND_TITLE

IMPORTING	TITLE	TYPE	STRING

This event is triggered if the element in the GUI control wants to set its own GUI title. The event returns the title, which you can display in a separate header bar.

Note:

The report SRM_DEMO_RM_BASE_CONTROL is provided for integration of the RM Base control.

3 RM Stacked Control

You can use the RM stacked control store multiple elements in a stack and implement navigation within the control. The R M stacked control keeps all elements that are in editing mode in a stack. In contrast to RM base control, you must unload the elements explicitly.

RM stacked control is realized using the class CL_SRM_STACKED_CONTROL. The class CL_SRM_STACKED_CONTROL inherits from the class CL_SRM_BASE_CONTROL.

3.1 IF_SRM_STACKED_CONTROL

The RM stacked control has implemented the interface IF_SRM_STACKED_CONTROL in addition to the interface IF_SRM_BASE_CONTROL. This interface has the methods described below:

IF_SRM_STACKED_CONTROL~CHECK_ITEM( )

IMPORTING	IM_POID	TYPE	IF_SRM_POID
RETURNING	RE_IS_IN_STACK	TYPE	SRMBOOLEAN

This method checks whether the referenced element is contained in the stack.

The stack only contains elements that are in change mode. Navigation should therefore be implemented as a new request. If the user wants to use backwards navigation to navigate to an element that was displayed in Display mode, the element is no longer in the stack. The element must be reloaded using the method IF_SRM_BASE_CONTROL~DISPATCH_REQUEST. If the user navigates to an element that was displayed in Change mode, this element is still contained in the stack. You can display this element using the method IF_SRM_STACKED_CONTROL~MOVE_TO_ITEM.

IF_SRM_STACKED_CONTROL~UNLOAD_ALL( )

IMPORTING	IM_UNLOAD_MODE	TYPE	I

This method unloads all elements that have been displayed in the control.

For the import parameter UNLOAD_MODE, you can enter one of the three constants below: IF_SRM_BASE_CONTROL=>UNLOAD_MODE_DESTROY_ONLY, IF_SRM_BASE_CONTROL=>UNLOAD_MODE_NOTIFY or IF_SRM_BASE_CONTROL=>UNLOAD_MODE_NOTIFY_AND_DESTROY (for an explanation of these constants, see the method IF_SRM_BASE_CONTROL~UNLOAD_CURRENT).

Theoretically, the SP can cancel the unload process.

IF_SRM_STACKED_CONTROL~UNLOAD_ITEM( )

IMPORTING	IM_POID	TYPE	IF_SRM_POID
IMPORTING	IM_UNLOAD_MODE	TYPE	I
RETURNING	RE_FOUND	TYPE	SRMBOOLEAN

This method unloads the element identified by the specified POID.

This method corresponds to the method IF_SRM_BASE_CONTROL~UNLOAD_CURRENT, with the following difference: You can use the method UNLOAD_ITEM to delete all elements from the stack. With the method UNLOAD_CURRENT, the top element is always deleted.

IF_SRM_STACKED_CONTROL~MOVE_TO_ITEM( )

IMPORTING	IM_POID	TYPE	IF_SRM_POID

This method makes the element from the stack that is identified by the specified POID visible.