

# Desktop Office Integration (BC-CI)



**Release 4.6C**



## Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.

HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

## Contents

<b>Desktop Office Integration (BC-CI)</b> .....	<b>10</b>
Desktop Office Integration.....	10
Desktop Office Integration - Architecture.....	10
Instances for SAP Desktop Office Integration .....	11
Office Integration Programming .....	13
Using Controls in a WAN.....	14
Special Considerations for Desktop Office Integration .....	15
<b>Error Handling after Calling a Method .....</b>	<b>15</b>
Error Messages and Their Meanings .....	17
raise_message.....	18
get_message.....	19
<b>Central Instance for Managing SAP DOI.....</b>	<b>19</b>
Office Integration Implementation .....	19
Generic Parameters.....	22
init_control.....	22
destroy_control .....	24
link_control.....	24
release_all_documents .....	25
set_window_properties .....	25
set_inplace_border_space.....	26
get_document_proxy .....	27
get_registered_doc_types.....	28
check_registered_doc_types .....	29
get_link_server.....	29
get_table_collection .....	29
get_extensions.....	30
set_focus.....	30
get_control_object.....	31
<b>Documents .....</b>	<b>31</b>
Working With Documents.....	31
Methods of the Interface i_o_i_document_proxy.....	34
Generic Parameters.....	35
create_document.....	35
view_document.....	36
view_document_from_table.....	37
open_document.....	38
open_document_from_table .....	39
reopen_document.....	40
open_activex_document.....	42
play_document .....	42
play_document_from_table .....	43
save_document_to_url .....	44

save_document_to_table.....	44
save_as.....	45
save_copy_as.....	45
close_document.....	46
close_activex_document.....	46
has_changed.....	47
release_document.....	47
is_open.....	47
is_destroyed.....	48
has_standard_interface.....	48
add_menu_bar.....	49
add_menu_popup.....	49
add_menu_item.....	50
delete_menu_item.....	51
select_all.....	51
clear_selection.....	51
cut_selection.....	52
copy_selection.....	52
paste_clipboard.....	52
paste_special.....	52
page_setup.....	53
print_document.....	53
execute_macro.....	53
has_wordprocessor_interface.....	54
get_wordprocessor_interface.....	55
get_document_type.....	55
get_document_handle.....	55
get_application_property.....	56
update_document_links.....	57
has_spreadsheet_interface.....	57
get_spreadsheet_interface.....	57
has_form_interface.....	58
get_form_interface.....	58
has_mail_merge_interface.....	58
get_mail_merge_interface.....	59
has_pivot_interface.....	59
get_pivot_interface.....	59
get_script_collection.....	60
get_document_cookie.....	60
Processing Office Application Events.....	61
Event Handling Using Static Methods.....	61
Event Handling Using Instance Methods.....	62
<b>Exchanging Data Between R/3 and the Desktop Application.....</b>	<b>63</b>
The Link Server.....	64
Working With the Link Server.....	65

Generic Parameters.....	66
start_link_server .....	66
stop_link_server.....	67
add_string_item .....	67
add_table_item2 .....	68
add_table_item .....	69
add_binary_item .....	70
remove_link_item.....	71
copy_link_item_to_clipboard .....	71
execute_copy_link_dialog.....	72
Strategies for Working with the Link Server.....	72
The Table Collection .....	73
Using the Table Collection .....	74
Generic Parameters.....	76
add_table .....	76
add_table_by_url .....	78
get_table .....	78
get_table_by_url .....	79
remove_table .....	80
remove_all_tables.....	80
<b>The Word Processor Interface .....</b>	<b>80</b>
Working With The Word Processor Interface.....	81
Generic Parameters.....	82
get_version.....	82
install_template .....	82
get_version.....	83
set_template.....	83
get_template_info .....	84
remove_template .....	84
search .....	85
replace .....	85
insert_table.....	86
insert_table2.....	88
get_table_info .....	90
clear_table.....	91
<b>The Form Interface .....</b>	<b>91</b>
Working With The Form Interface .....	93
Generic Parameters.....	94
set_data .....	94
get_data .....	95
add_field.....	96
delete_fields.....	96
disconnect_fields .....	97
get_form_fields .....	97
protect .....	97

set_modus .....	98
<b>The Mail Merge Interface .....</b>	<b>98</b>
Using the Mail Merge Interface .....	99
Generic Parameters .....	99
set_data_source .....	100
get_fields .....	101
merge_one .....	101
merge_range .....	102
print .....	102
view .....	102
shutdown .....	102
view_field_codes .....	103
<b>The Script Collection .....</b>	<b>103</b>
Using the Script Collection .....	104
Generic Parameters .....	104
add_script .....	105
add_script_from_table .....	105
remove_all_scripts .....	106
remove_script .....	106
<b>The Spreadsheet Interface .....</b>	<b>107</b>
Using the Spreadsheet Interface .....	107
Generic Parameters .....	109
The Updating Parameter .....	110
get_active_sheet .....	110
get_dimension .....	111
get_protection .....	111
get_sheets .....	112
get_ranges_data .....	112
get_ranges_names .....	113
get_cell_format .....	114
get_cell_formats_table .....	114
get_selection .....	115
get_selected_areas .....	115
select_range .....	116
set_selection .....	116
select_sheet .....	117
set_color .....	117
set_font .....	118
set_format .....	119
set_format_string .....	120
set_frame .....	120
set_ranges_format .....	121
cell_format .....	121
protect .....	122
protect_range .....	122

add_sheet .....	123
set_sheet_name .....	123
delete_sheet.....	124
insert_range_dim .....	124
change_range .....	125
insert_range .....	126
insert_ranges .....	126
insert_one_table .....	127
set_ranges_data .....	127
clear_range .....	128
clear_all_ranges.....	129
delete_content_ranges .....	129
delete_ranges .....	129
hide_columns.....	130
show_columns .....	131
hide_rows.....	131
show_rows .....	131
set_hierarchy.....	132
set_hierarchy_table.....	133
clear_hierarchy.....	133
print .....	133
set_zoom .....	134
fit_widest .....	134
screen_update .....	134
load_lib.....	135
version.....	136
Important Table Structures.....	136
<b>The Pivot Interface .....</b>	<b>142</b>
Using the Pivot Interface .....	143
Generic Parameters.....	144
set_source_table.....	145
set_fieldtype .....	147
get_allfields .....	148
get_allpivottables .....	149
drill_down.....	149
drill_up.....	150
Important Table Structures .....	150
<b>Test Tools for Desktop Office Integration .....</b>	<b>152</b>
DOI Installation Test Using a Standalone Executable Program .....	153
DOI Installation Test Within Word .....	154
DOI Installation Test Within Excel.....	154
Test for the TableFactory OCX .....	155
<b>The Document Viewer.....</b>	<b>155</b>
<b>Using the Document Viewer.....</b>	<b>157</b>
Coding Example.....	158
<b>Methods of the Document Viewer.....</b>	<b>159</b>

---

init_viewer .....	159
view_document_from_url .....	160
view_document_from_table .....	161
close_document .....	162
destroy_viewer .....	162

## Desktop Office Integration (BC-CI)

### Desktop Office Integration

#### Purpose

You can use Desktop Office Integration to connect any application that supports OLE2 to your R/3 System. You can trigger office applications from the R/3 System and react to office application events.

#### Implementation considerations

The applications must support the OLE2 interface. The presentation servers must be running under Windows95 or Windows NT.

#### Integration

You should store documents on the [Business Document Server \[Ext.\]](#).

Ensure that SAPgui is installed with all of its components (especially the OCX modules)

#### Features

Desktop Office Integration is an ABAP Objects interface that you can use to open, close, and control special desktop Office applications using the OLE2 interface. You can start the Office application either in a separate window or within the R/3 window.

You can store the documents in any way. For example, you can use the R/3 database, HTTP and FTP servers, or the local file system on your frontend. You address documents using URLs, and load them into the office application using a special data retrieval component.

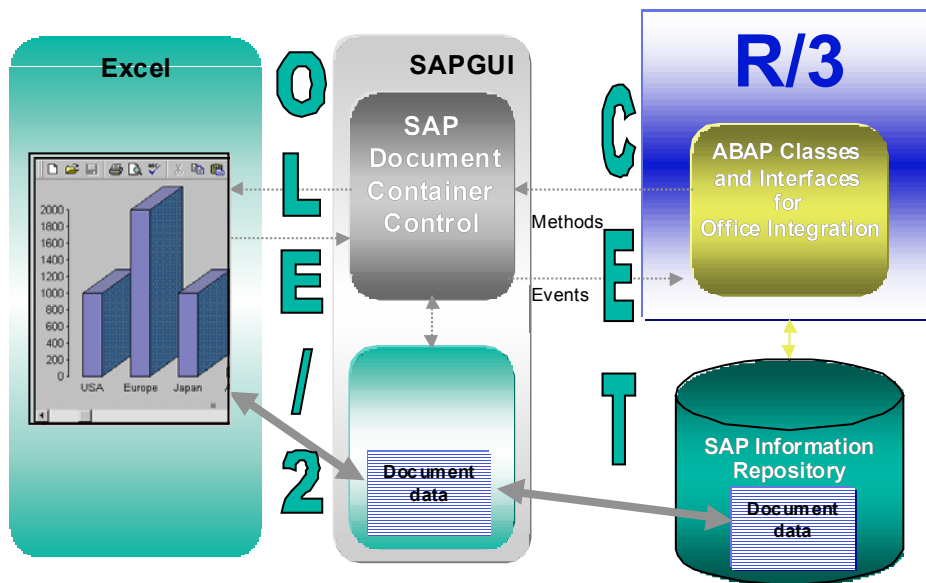
You can react in the ABAP program to events that are triggered in the Office application. You can also import data from the R/3 System (fields, tables, pictures, RTF texts) into the document using links. You can ensure that everything in the document is up to date by updating the links.

### Desktop Office Integration - Architecture

The R/3 integration technology allows you to integrate any OLE-compatible desktop application such as Microsoft Office, Lotus SmartSuite, CorelOffice, or Visio into the R/3 System.

Communication and data transfer work as follows:

Instances for SAP Desktop Office Integration



Communication between the SAPgui (presentation server) and the R/3 application server was extended in Release 4.0. This made it possible to include ActiveX controls in SAPgui ([Control Framework \[Ext.1\]](#)).

Desktop Office Integration uses this interface for special controls (SAP Document Container Control and SAP Data Provider) that are the communication channel between the R/3 System and the Desktop Office Application.

The SAP Document Container Control uses methods and attributes to communicate with a desktop application using the OLE2 interface.

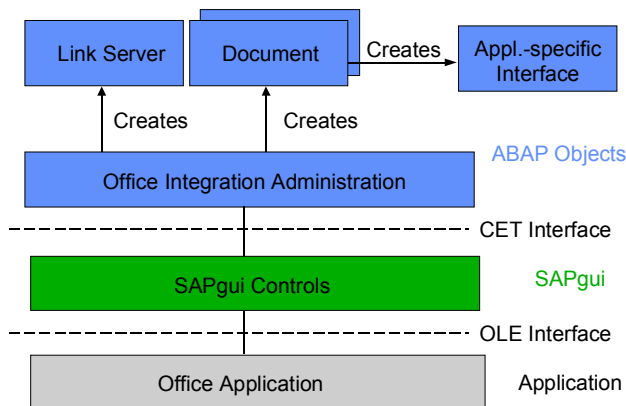
The SAP Data Provider is a container and converter for data in different formats. Data that you import into an application is converted into a readable format based on the Internet standard MIME. The SAP Data Provider buffers document data that is administered in the R/3 System. It also buffers R/3 data, which can be inserted into documents using R/3 links.

Both controls have an interface to the R/3 System. The interface to the SAP Data Provider is implemented using function modules. ABAP classes and interfaces for Desktop Office Integration provide the interface to the SAP Document Container Control.

## Instances for SAP Desktop Office Integration

The interface for SAP Desktop Office Integration is implemented using ABAP Objects (as described in [Desktop Office Integration - Architecture \[Page 10\]](#)). The following diagram shows the most important instances:

## Instances for SAP Desktop Office Integration



At runtime, the first step is to create a central instance to manage the Office integration. Its methods generate further objects that are required for control and communication during the session.

- The management instance contains technical data required to initialize the communication with the SAPgui using the CET interface. The CET interface communicates with special SAPgui controls that are connected to the office application by means of an OLE2 interface.
- The instance for document administration contains information about the URL address of documents. You can trigger office applications functions using the implemented methods. The instance also records office application events that you can react to in your ABAP program.
- There are two ways of transferring data between the R/3 System and the desktop application:
- Using the link server instance to administer embedded links in the office application to objects in the R/3 System (fields, tables, and so on)
- Using the Table Collection instance. This is particularly suitable for working with tables. You can also transfer data from the desktop application to the R/3 System.
- If the office application has a standard interface, you can use its functions to edit the document (select, copy, delete, ...). You may also create additional toolbars.
- If there are also specific interfaces available for a particular application, you can create instances of these as well. Desktop Office Integration contains the following interfaces:
- [The Word Processor Interface \[Page 80\]](#)
- [The Form Interface \[Page 91\]](#)
- [The Mail Merge Interface \[Page 98\]](#)
- [The Script Collection \[Page 103\]](#)
- [The Spreadsheet Interface \[Page 106\]](#)

## Office Integration Programming

### 1. Initialize the Desktop Office Integration:

This starts the SAP Document Container Control in the SAPgui.

When you initialize the Desktop Office Integration, you specify whether the document should be opened in a separate window or within the R/3 window (in-place).

Declare an instance for the central object of the Office Integration with reference to the interface *i\_oi\_containercontrol*.

### 2. Create one or more documents:

Create an instance with reference to the interface *i\_oi\_document\_proxy* for every document to be opened simultaneously. This starts a document container in the SAPgui, and the corresponding office application is started.

### 3. Start the data transfer instance (optional):

This allows you to work with the link server and table collection.

**Link Server:** The link server is used for managing data from the ABAP program and transporting it to the Desktop application. You define it with reference to the interface *i\_oi\_link\_server*.

Objects that you define or store in the R/3 System (fields, internal tables, RTF files, pictures...) and want to insert in a document must be declared to the link server before you open the document.

**Table Collection:** The table collection is particularly useful for passing tables. You can also use it to pass tables to the R/3 System that have been filled in the desktop application. You create the table collection with reference to the interface *i\_oi\_table\_collection*.

### 4. If you want to use application-specific functions, instantiate the corresponding interface or interfaces:

- [The Word Processor Interface \[Page 80\]](#)
- [The Form Interface \[Page 91\]](#)
- [The Mail Merge Interface \[Page 98\]](#)
- [The Script Collection \[Page 103\]](#)
- [The Spreadsheet Interface \[Page 106\]](#)

### 2. Interpret events for processing (optional):

You can react to events that are triggered by the user in the office application or by the application itself.

### 3. Close the individual documents or the entire office application:

You can destroy objects that you no longer need by calling special methods.

## Using Controls in a WAN

# Using Controls in a WAN

When you use controls in your programs, you place an extra load on the communication channel between the frontend and backend. In a LAN, and particularly in a WAN environment, this can be a critical factor.

The problem is alleviated somewhat by buffering mechanisms (see also [Automation Queue \[Ext.\]](#)). Use these points as a guideline to using controls in a WAN.

The documentation for the individual controls also contains more specific notes about using that control in a WAN.

## Using CL\_GUI\_CFW=>FLUSH

The method [CL\\_GUI\\_CFW=>FLUSH \[Ext.\]](#) synchronizes the automation queue and the ABAP variables in it. Calling it often generates a synchronous RFC call from the application server to the frontend. To optimize the performance of your application, you should call this method as little as possible.

It is often a good idea to read all control attributes in a single automation queue (for example, at the beginning of the PAI) and retrieve them in a single synchronization. You should, in particular, do this when you read attributes that are not necessary in your event handlers or the PAI/PBO cycle.

You do not need to include a "safety flush" at the end of the PBO to ensure that all method calls are transported to the frontend. A flush at the end of the PBO is guaranteed. Consequently, you cannot construct an automation queue spread over several screens.

**There is no guarantee that an automation queue will be sent when you call CL\_GUI\_CFW=>FLUSH. The queue recognizes whether it contains any return values. If this is not the case, it is not sent.**

If you have a queue with no return values, and want to ensure that it is synchronized, you can use the Control Framework method [CL\\_GUI\\_CFW=>UPDATE\\_VIEW \[Ext.\]](#). You should only use this method if you absolutely need to update the GUI. For example, you might have a long-running application in which you want to provide the user with regular updates on the status of an action.

After you have read the attributes of a control, the contents of the corresponding ABAP variables are not guaranteed until after the next flush. The contents of the ABAP variables remain undefined until this call occurs. In the future, there will be cases in which this flush is unnecessary. They will be recognized by the automation queue and the corresponding flush call will be ignored.

## Creating Controls and Passing Data

Creating controls and passing data to them is normally a one-off procedure, which in comparison to using normal screen elements can be very runtime-intensive. You should therefore not use any unnecessary controls, or pass unnecessary data to the controls that you are using.

A typical example is a tabstrip control with several tab pages. If the pages contain controls, you should consider using application server scrolling instead of local scrolling, and not loading the controls until the corresponding page is activated by the user. The same applies to passing data to the controls on tab pages.

If you want to differentiate between LAN and WAN environments when you pass data to a control, you can use the function module `SAPGUI_GET_WANFLAG`. In some applications, you may need to pass different amounts of data or use a complete fallback in a WAN application. The

## Special Considerations for Desktop Office Integration

environment affects, for example, the number of same-level nodes that you can transfer to a tree control without having to introduce artificial intermediate levels.

Unlike screen elements, controls only have to be created and filled with data once. From a performance point of view, this means that they become more profitable the longer they exist. In applications that are called repeatedly, and therefore initialized repeatedly, controls can have a negative effect on performance. In applications that use the same screen for a long time, on the other hand, you may find that using controls results in improved performance.

You can always use the [performance tools \[Ext.\]](#) to check the advantages and disadvantages in terms of network load that using a control brings.

### Storing Documents, Picture, and Other Data

Release 4.6A sees the introduction of a frontend cache for accessing documents from the Business Document Service (BDS). You are strongly recommended to store desktop documents, images, and other data in the BDS and not in the R/3 database. Documents from the BDS can be cached at the frontend, and therefore only have to be loaded over the network once.

## Special Considerations for Desktop Office Integration

In addition to the general considerations that apply to all controls, you should also note the following:

In a WAN environment, you should use Desktop Office Integration with caution, especially if you will be loading documents from the R/3 application server or a web server connected to the WAN.

There is also no way of reducing the quantities of data with which Desktop Office Integration works. The exception to this are documents stored in the Business Document Server that do not often change: it is possible to cache these documents at the frontend.

## Error Handling after Calling a Method

### Purpose

Error handling in Desktop Office Integration can be a complex matter. The methods that you use can return a wide range of error codes. Furthermore, since Desktop Office Integration is being developed further, new error codes will appear in further releases. The following notes should make Desktop Office Integration easier to use.

After calling a method, you must find out if the call was successful. All methods contain a return parameter called *ret\_code*, which contains the value 'OK' if the call was successful, or an error code if an error occurred. The class *c\_oi\_errors* contains symbolic constants for the 'OK' value and for all other error codes. Each method also stores additional error information. You can use the class method *raise\_message* in class *c\_oi\_errors* to display a message in the status bar containing the precise cause of the error. The *raise\_message* method executes the ABAP statement MESSAGE when an internal error occurs, and displays a message from message class SOFFICEINTEGRATION.

## Error Handling after Calling a Method

Use the following two processes as guidelines for your error handling. The first process does not investigate any errors in detail. The second process deals explicitly with some errors, and uses the *raise\_message* method to display the rest.

If you do not synchronize the [automation queue \[Ext.\]](#) when you call methods (see the [generic parameter \[Page 144\]](#) *no\_flush*), the value of the *ret\_code* parameter is meaningless. This is because the value can only be set after the automation queue has been synchronized. Each method call returns a reference to an error object (*error*). Process III explains how to use the error object.

## Process I

1. When you have called a method of the SAP Desktop Office Integration, call the class method *show\_message* of class *c\_oi\_errors*:

```
CALL METHOD C_OI_ERRORS=>RAISE_MESSAGE
EXPORTING TYPE = type
```

Use the type parameter to pass the message type (A, E, W, I, S).

If an error occurred in the execution of the SAP Desktop Office Integration method, the system displays a message of the corresponding type. The correct message text is determined automatically by the class method *raise\_message*.

## Process II

1. When you call a method of the SAP Desktop Office Integration, pass a variable *ret\_code* in the call, in which the error code can be stored.
2. Check first whether an error has occurred at all. If no error occurred, branch directly to the section of the program that carries on with 'normal' processing.
3. Next, check whether an error occurred that you want to handle explicitly. In this case, branch to the section of the program that handles that error.
4. For all other error codes, you must call the class method *raise\_message* of class *c\_oi\_errors* as described in process I.



Example of process II:

```
IF ret_code EQ c_oi_errors=>ret_ok.
  " Document opened successfully
ELSEIF ret_code EQ c_oi_errors=> ret_document_already_open.
  " Special error handling, e.g. dialog box.
ELSE.
  CALL METHOD c_oi_errors=>raise_message
    EXPORTING type = 'E'.
ENDIF.
```

### Process III

1. Create an internal table to hold references to any error objects:  
DATA: errors TYPE REF TO i\_o\_i\_error OCCURS 0 WITH HEADER LINE.
2. After each method call in which you set the parameter no\_flush = 'X', insert the object reference into the internal table:  
CALL METHOD control->get\_link\_server  
    EXPORTING server\_type = server\_type  
              no\_flush = 'X'  
    IMPORTING link\_server = link\_server  
              retcode = retcode  
              error = errors.  
APPEND errors.
3. Once you have synchronized the automation queue (either by setting the parameter no\_flush = '' or by calling the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#)), you can process the references in the error table:  
LOOP AT errors.  
    CALL METHOD errors->raise\_message  
        EXPORTING type = 'E'  
        EXCEPTIONS message\_raised = 1  
                    flush\_failed = 2.  
ENDLOOP.  
FREE errors.

## Error Messages and Their Meanings

The following list contains the possible values of the `retcode` parameter following a method call in SAP Desktop Office Integration:

Value	Description
RET_OK	Success
RET_ERROR	An error occurred while trying to execute the method.  You can display a detailed error message using C_OI_ERRORS=>RAISE_MESSAGE.
RET_CALL_NOT_FLUSHED	Automation Controller queue not flushed.
RET_CONTROL_NOT_INITIALIZED	Document Container Control not initialized.
RET_OBJECT_NOT_INITIALIZED	Object in ABAP Objects not initialized.
RET_OBJECT_ALREADY_INITIALIZE D	Object in ABAP Objects already initialized.
RET_INPLACE_DISABLED	You cannot open documents in an R/3 window in this mode.
RET_DOC_TYPE_NOT_REGISTERED	No application registered on the PC for the specified document type.

**raise\_message**

<b>RET_OPEN_DOCUMENT_FAILED</b>	Error opening a document.
<b>RET_OPEN_DOCUMENT_FILE_FAILED</b>	Error opening a document from a temporary file.
<b>RET_SAVE_DOCUMENT_FAILED</b>	Error saving a document.
<b>RET_SAVE_DOCUMENT_FILE_FAILED</b>	Error saving a document in a temporary file.
<b>RET_DOCUMENT_NOT_OPEN</b>	Document not opened
<b>RET_DOCUMENT_ALREADY_OPEN</b>	Document already open.
<b>RET_DOCUMENT_NO_VIEW_DATA</b>	Document does not contain any presentation data.
<b>RET_LINK_SERVER_NOT_ACTIVE</b>	SAP Link Server not active.
<b>RET_LINK_SERVER_ALREADY_ACTIVE</b>	Another SAP Link Server is already active under the specified name.
<b>RET_NO_LINK_ITEM_SELECTED</b>	No item selected in CopyLinkItemToClipboard.
<b>RET_NO_STANDARD_INTERFACE</b>	No standard interface implemented for this document type.
<b>RET_INTERFACE_NOT_SUPPORTED</b>	No application-specific interface (for example, word processor interface) implemented for this document type.
<b>RET_METHOD_NOT_SUPPORTED</b>	Method not supported for this document type.
<b>RET_CONTROL_INTERNAL_ERROR</b>	An internal error occurred while trying to execute a method.  You can display a detailed error message using <code>C_OI_ERRORS=&gt;RAISE_MESSAGE</code> .

**raise\_message**

You use this method to display a message.

```
CALL METHOD error->raise_message
  EXPORTING type = type
  EXCEPTIONS message_raised = 1
             flush_failed = 2.
```

**Description of parameters**

Parameter	Optional	Description
type		Message type (A, E, W, I, S).

## get\_message

You use this method to retrieve the system message generated automatically by the SAP Desktop Office Integration.

```
CALL METHOD error->get_message
  IMPORTING message_id = message_id
           message_number = message_number
           param1      = param1
           param2      = param2
           param3      = param3
           param4      = param4.
```

### Description of parameters

Parameter	Optional	Description
message_id		Message class
message_number		Message number
param1 to param4		Variable part of message

## Central Instance for Managing SAP DOI

### Definition

You define this instance with reference to the interface *i\_oi\_container\_control*.

### Use

The interface *i\_oi\_container\_control* has methods that you can call for the instance control. You use these methods to create additional instances that you need for Desktop Office Integration.

## Office Integration Implementation

When you program office applications to start in the R/3 window, the interface *i\_oi\_container\_control* is used to create and manage any further objects for Desktop Office Integration. To create an instance for this object:

### Declaration

1. Your data declaration section must begin with the following declarations:

```
TYPE-POOLS soi.
CLASS c_oi_errors DEFINITION LOAD.
```

The type pool contains important constants and type definitions. The class definition is used for error handling from the method calls of the Desktop Office Integration.

**Office Integration Implementation**

2. Declare an instance for the central object of the Office Integration with reference to the interface *i\_oi\_container\_control*:  
DATA control TYPE REF TO i\_oi\_container\_control.
3. Declare an object variable for all of the documents that you want to have open at once.  
DATA: document TYPE REF TO i\_oi\_document\_proxy.
4. **Declare an object variable for the data transfer (optional):**  
**If you are working with the Link Server:**  
DATA: link\_server TYPE REF TO i\_oi\_link\_server.  
**If you are working with the Table Collection:**  
DATA: table\_coll TYPE REF TO i\_oi\_table\_collection.

**Initialization**

The following steps must only occur once in your program.

5. Create the instance *control*.  
CALL METHOD c\_oi\_container\_control\_creator=>get\_container\_control  
IMPORTING control = control  
retcode = retcode.
6. If you want to use Desktop Office Integration in-place, you also need to create a [container \[Ext.\]](#):  
DATA: container TYPE REF TO cl\_gui\_custom\_container.  
CREATE OBJECT container  
EXPORTING container\_name = 'CONTAINER'.
7. Call the method *init\_control*.  
You have now created the central object for Desktop Office Integration and the connection to the relevant GUI control.  
CALL METHOD control->init\_control  
EXPORTING r3\_application\_name = 'Demo Document Container'  
inplace\_enabled = 'X'  
parent = container  
IMPORTING retcode = retcode.
8. Create an instance document for each document that you want to open:  
CALL METHOD control->get\_document\_proxy  
EXPORTING document\_type = document\_type  
document\_format = document\_format  
IMPORTING document\_proxy = document

retcode = retcode.

9. Create the instance for data transfer (Link Server: method *get\_link\_server*; Table Collection: method *get\_table\_collection*) (optional).

10. Create any application-specific interfaces:

- [The Word Processor Interface \[Page 80\]](#)
- [The Form Interface \[Page 91\]](#)
- [The Mail Merge Interface \[Page 98\]](#)
- [The Script Collection \[Page 103\]](#)
- [The Table Interface \[Page 106\]](#)

## Processing

5. Define the processing steps for your documents (open, close, save, ...)

## Closing

6. When you have finished editing a document, close it, save it, and release the memory area.

7. When you no longer need the link server, close it and release the memory area (optional).

8. Destroy the container instance (see item 6):

CALL METHOD container->free.

9. At the end of the program, use the method *destroy\_control* to delete the instance control. Before doing this, however, you should delete all of the other instances that you had generated.



Remember that you should include [error handling \[Page 15\]](#) after each method call.

When you destroy object and interface instances in Desktop Office Integration, remember that ABAP Objects, unlike other object-oriented languages, has no destructors. This means that when you release the *i\_oi\_container\_control* instance using the FREE statement, only the memory space on the application server is released. The SAPgui controls and documents on the presentation server are not destroyed. For this reason, you should always call the methods that destroy the SAPgui controls and close the documents.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'x': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'x'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## init\_control

You use this method to create the object that manages the link to the OLE interface.

CALL METHOD control->init\_control

```

EXPORTING parent          = parent
  r3_application_name     = r3_application_name
  inplace_enabled        = inplace_enabled
  inplace_show_toolbars  = inplace_show_toolbar
  inplace_scroll_documents = inplace_scroll_documents
  inplace_resize_documents = inplace_resize_documents
  shell_style            = shell_style
  autoalign              = autoalign
  no_flush                = no_flush
  register_on_close_event = register_on_close_event
  register_on_custom_event = register_on_custom_event
  rep_id                  = rep_id
  dynpro_nr               = dynpro_nr
  gui_container           = gui_container
  inplace_mode            = inplace_mode
  parent_id               = parent_id
IMPORTING retcode         = retcode
  error                   = error.

```

### Description of parameters

Parameter	Optional	Description
parent		Name of the container in which you want to start the desktop office application.
r3_application_name		Heading
inplace_enabled	X	'X': Allows a document to be opened in-place. ' ': No documents may be opened in-place.
inplace_show_toolbar	X	Show the toolbar of the desktop application (when inplace_enabled = 'X')
inplace_scroll_documents	X	Show the scrollbars of the desktop application (when inplace_enabled = 'X')
inplace_resize_documents	X	Resize the document display when the R/3 window is resized (when inplace_enabled = 'X')
autoalign		'X': Fill the entire control container with the office application.
shell_style	X	The display format of the container control in the R/3 screen Possible values are contained in the include <SAPOLECONTAINERCONTROL>. You do not normally have to change the default value of the parameter.
register_on_close_event	X	Registers the event triggered when the office application user closes the document. If you want to react to this event in your ABAP program, register with "X".
register_on_custom_event	X	Registers additional events in the office application (for example, macro calls) Pass the value "X" to react to this event in your ABAP program.
rep_id	X	<b>Obsolete.</b> Previous use: Names the program or the module pool of the screen in which the office application will be displayed. (If you have not specified a container but want to process the events from the desktop application).
dynpro_nr	X	<b>Obsolete.</b> Previous use: Number of the screen on which you want to create the SAP Document Container Control. (If you have not specified a container but want to process the events from the desktop application).
gui_container	X	<b>Obsolete.</b> Previous use: Name of the container defined in the Screen Painter for which the SAP Document Container Control should be created.
parent_id	X	<b>Obsolete.</b> Previous use: Dialog box level at which the control appears: dynpro_0 to dynpro_9: The control is displayed at dialog box level 0 (on the R/3 screen) to level 9 (ninth-level dialog box). dynpro_default : Determines automatically the active level at the point where the method is called. This is the level that is then used. This only works if the call occurs in the PBO event.

**destroy\_control**

<code>inplace_mode</code>	X	This parameter is obsolete. Do not use it in new programs.
---------------------------	---	--



For further information about event handling, refer to [Processing Office Application Events \[Page 60\]](#).

**destroy\_control**

You use this method to destroy the central Office Integration instance.

```
CALL METHOD control->destroy_control
  EXPORTING no_flush = no_flush
  IMPORTING retcode = retcode
           error   = error.
```



Before calling the method `destroy_control`, make sure that you have saved all changes to your documents. When you call this method, the method [release\\_all\\_documents \[Page 25\]](#) is also called automatically.

After this call, release the ABAP Objects instance:

```
FREE control.
```

**link\_control**

This method has become obsolete with the introduction of the [container concept \[Ext.\]](#) in Release 4.6. Do not use this method any longer. Instead, use the [LINK \[Ext.\]](#) method on the container instance.

When the office application is initialized, the connection is set for the current program (SY-REPID) and screen (SY-DYNPRO). The SAP Document Container Control is created on this screen. If you want to specify a different screen after the initialization, you can use the method `init_control`, specifying the new program and screen.

```
CALL METHOD control->link_control
  EXPORTING rep_id      = rep_id
           dynpro_nr   = dynpro_nr
           gui_container = gui_container
           no_flush    = no_flush
  IMPORTING retcode    = retcode
           error       = error.
```

**Description of parameters**

Parameter	Optional	Description
-----------	----------	-------------

release\_all\_documents

rep_id	X	<b>Obsolete.</b> Previous use: Names the program or the module pool of the screen in which the office application will be displayed. (If you have not specified a container but want to process the events from the desktop application).
dynpro_nr	X	<b>Obsolete.</b> Previous use: Number of the screen on which you want to create the SAP Document Container Control. (If you have not specified a container but want to process the events from the desktop application).
gui_container	X	<b>Obsolete.</b> Previous use: Name of the container defined in the Screen Painter for which the SAP Document Container Control should be created.
rep_id	X	Name of program or module pool containing the screen on which you want to create the SAP Document Container Control.
dynpro_nr	X	Number of the screen on which you want to create the SAP Document Container Control.
gui_container	X	Name of the container defined in the Screen Painter for which the SAP Document Container Control should be created.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## release\_all\_documents

This method closes all documents on the presentation server. It does not first save their data.

```
CALL METHOD control->release_all_documents
EXPORTING no_flush = no_flush
IMPORTING retcode = retcode
error = error.
```



Use this method carefully, otherwise changes to documents may be lost. Ensure that you save all of the changes before calling the method.



To release the memory required to manage the data object document on the application server, use the ABAP statement FREE document.

## set\_window\_properties



This method has become obsolete with the introduction of the [container concept \[Ext.\]](#) in Release 4.6. Do not use this method any longer.

You use this method to change the display format of the office application in the R/3 window.

**set\_inplace\_border\_space**

```
CALL METHOD control->set_window_properties
EXPORTING metric = metric
         left  = left
         top   = top
         width = width
         height = height
         align = align
         no_flush = no_flush
IMPORTING retcode = retcode
         error   = error.
```

**Description of parameters**

Parameter	Optional	Description
<code>metric</code>	X	Initial: spaces must be given in pixels Screen: spaces correspond to screen lines or screen columns
<code>left</code>	X	Sets the left margin in the office application window
<code>top</code>	X	Sets the top margin in the office application window
<code>width</code>	X	Sets the width of the office application window
<code>height</code>	X	Sets the height of the office application window
<code>align</code>	X	Sets window attributes This parameter allows you to fix the window in the screen or to make its size dependent on the size of the current window. Possible values are contained in the include <CTLDEF>.



You can only use this method if you have not specified a GUI container in *init\_control* or *link\_control*.

**set\_inplace\_border\_space**

This method has become obsolete with the introduction of the [container concept \[Ext.\]](#) in Release 4.6. Do not use this method any longer.

You use this method to determine an area in which no documents can be placed. Use this method when you do not want to display the whole of the desktop application when you call it. It places the application within a fixed border. The size of the border depends on the desktop application you are using.

```
CALL METHOD control->set_inplace_border_space
EXPORTING left   = left
         top     = top
         right   = right
         bottom  = bottom
         no_flush = no_flush
IMPORTING retcode = retcode
         error   = error.
```

**Description of parameters**

Parameter	Optional	Description
left	X	Sets the left margin
top	X	Sets the top margin
right	X	Sets the right margin
bottom	X	Sets the bottom margin

## get\_document\_proxy

You use this method to create an instance for [Managing a Document \[Page 31\]](#)

```
CALL METHOD control->get_document_proxy
    EXPORTING document_type = document_type
             document_format = document_format
             register_container = register_container
             no_flush = no_flush
    IMPORTING document_proxy = document_proxy
             retcode = retcode
             error = error.
```

### Description of parameters

Parameter	Optional	Description
document_type		The type of document. The corresponding application is started using this identifier.
document_format	X	One of the following formats: OLE: Microsoft's Compound Document Format NATIVE: Application-specific format RTF: Rich Text Format
register_container	X	You need to register a container for office applications that do not support a container property (see note).
document_proxy		Creates an instance for document management. You define this instance with reference to the interface i_oj_document_proxy:



Do not hard-code the document type. Instead, use customizing tables or a user dialog to set it. To view possible document types, use the method *get\_registered\_doc\_types*.



Examples of document types:

Word.Document.8:	Word97
Excel.Sheet.8	Excel97 Worksheet
Excel.Chart.8	Excel97 Chart
Word.Document	The current latest version of Word.

**get\_registered\_doc\_types**



Many office applications either do not have a hierarchic object model for internal objects (application, document, sheet, cell, and so on) or, if they do have one, do not have a property in the document object that points to the OLE document container. However, this is required for message communication with the R/3 System and for data exchange using the Table Collection.

To allow you to use these desktop applications, you can register the SAP Document Container for the relevant document using the optional *register\_container* parameter. You can then access it using the GetObject() function, as demonstrated in the following VBA example:

```
Sub R3Test
    Dim cont as Object
    Set cont = GetObject(, "SAP.DocumentContainer")
    cont.SendCustomEvent("ON_DATA_CHANGE")
End Sub
```

**get\_registered\_doc\_types**

You use this method to list all the types of document that are registered for a particular interface type.

```
CALL METHOD control->get_registered_doc_types
    EXPORTING interface_type = interface_type
             no_flush      = no_flush
    IMPORTING descr_list   = descr_list
             retcode      = retcode
             error        = error.
```

**Description of parameters**

Parameter	Optional	Description
interface_type	X	One of the following interface types: ' ': all possible document types are displayed in an internal table 'BASIC': only document types supported by the standard interface are displayed. 'WORDPROCESSOR': only document types supported by the standard interface are displayed.
descr_list		Possible document types in an internal table with type soi_document_type_descr_list.



You can call this method without previously having used the [init\\_control \[Page 22\]](#) method. It allows you to check whether a desktop application exists on the frontend.

## check\_registered\_doc\_types

This method checks whether there is a list of desktop applications on the presentation server, and returns a list of existing applications. List the document types you want to check in the DOCUMENT\_TYPE column. The information about whether the application exists is returned in the AVAILABLE column.

```
CALL METHOD control->check_registered_doc_types
  EXPORTING no_flush      = no_flush
  IMPORTING error         = error
  CHANGING descr_list    = descr_list
  retcode                = retcode.
```

### Description of parameters

Parameter	Optional	Description
descr_list	X	Possible document types in an internal table with type soi_document_type_descr_list.



You can call this method without previously having used the [init\\_control \[Page 22\]](#) method. It allows you to check whether a desktop application exists on the frontend.

## get\_link\_server

You use this method to create an instance for the [link server \[Page 64\]](#).

```
CALL METHOD control->get_link_server
  EXPORTING server_type = server_type
  no_flush      = no_flush
  IMPORTING link_server = link_server
  retcode      = retcode
  error        = error.
```

### Description of parameters

Parameter	Optional	Description
server_type	X	Not used at present
link_server		Instance for the link server You define it with reference to the interface i_oi_link_server.



You can only start one link server for each instance control.

## get\_table\_collection

Generate an instance for [Table Collection \[Page 73\]](#).

**get\_extensions**

```
CALL METHOD control->get_table_collection
  EXPORTING no_flush = no_flush
  IMPORTING table_collection = table_collection
           retcode = retcode
           error = error.
```

**Description of parameters**

Parameter	Optional	Description
table_collection	X	Generated instance for the table collection, created with reference to the interface i_oi_table_collection.



You can only create one Table Collection instance for each Container Control. Any subsequent calls return a reference to the existing object. All of the open documents expose the same tables in the property tables.

**get\_extensions**

Use this method to return a list of file extensions valid for a particular application.

```
CALL METHOD control->get_extensions
  EXPORTING document_type = document_type
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode
  CHANGING extensions = extensions.
```

**Description of parameters**

Parameter	Optional	Description
document_type		The Program ID of the application as it appears in the Registry (such as 'Word.Document.8')
extensions		A single-column internal table containing the list of valid file extensions for the application. The method reads these from the Registry of your frontend computer. Define it with reference to <code>soi_colnames_table</code> .

**set\_focus**

You use this method to give the desktop application the focus.

```
CALL METHOD control->set_focus
  EXPORTING no_flush = no_flush
  IMPORTING retcode = retcode
           error = error.
```

## get\_control\_object

Use this method to get a reference to the Desktop Office Integration control. You can use this reference to address the [methods of the control framework \[Ext.\]](#).

```
CALL METHOD control->get_control_object
  EXPORTING no_flush = no_flush
  IMPORTING control = control_2
           retcode = retcode
           error   = error.
```

### Description of parameters

Parameter	Optional	Description
control_2	X	Reference to the Desktop Office Integration control. The data object must be created with reference to the class <code>cl_gui_control</code> .

## Documents

### Definition

You define this instance with reference to the interface `i_oj_document_proxy`:

### Use

Each instance document controls a document that you open in the office application. If you want to open more than one document at once, you must define a separate instance for each document.

You can also use methods to open, close, save, and edit documents in your office application programs.

### Integration

Before you create the instance document , you must create the central instance control for managing the Desktop Office Integration.

## Working With Documents

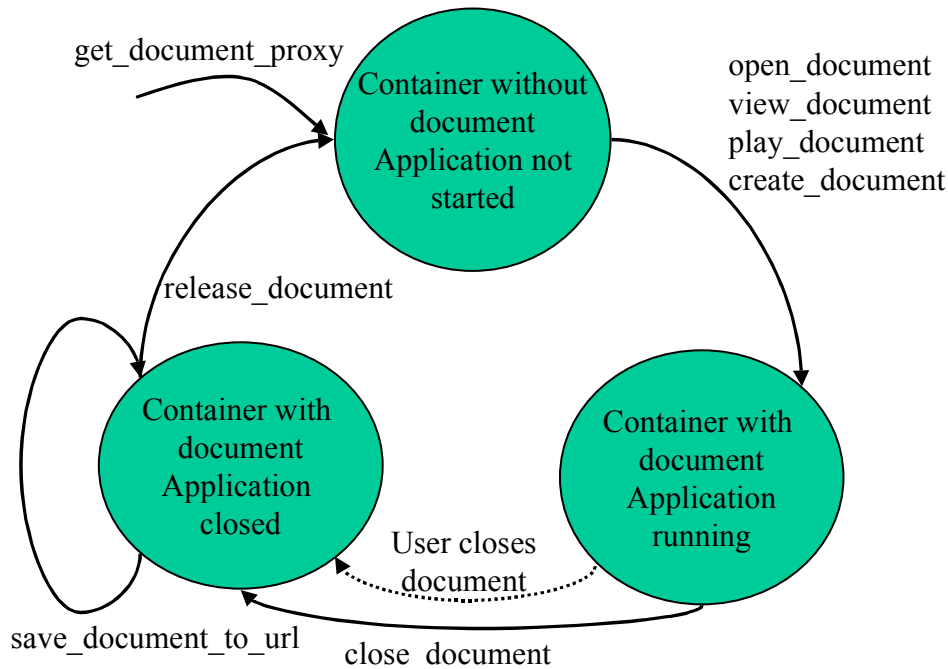
### Prerequisites

Before proceeding, you must create the central instance control for managing Office Integration. Refer to [Instance for Managing Office Integration \[Page 19\]](#).

Working With Documents

**Process flow**

The following diagram shows the process of editing a document:



**Declaration**

1. Declare an instance with reference to interface *i\_oi\_document\_proxy*:  
DATA: document TYPE REF TO I\_OI\_DOCUMENT\_PROXY.

**Initialization**

2. Create the instance for the document by using the method *get\_document\_proxy* on the central administration instance for Desktop Office Integration *control*.
3. In your application program, assign a handler method to the events that could be triggered in the document.
4. Open a document. For existing documents, use the *open\_document* (change mode), *view\_document* (display mode for metadata), or *play\_document* methods. To create a new document, use the *create\_document* method.

**Editing**

5. Edit your document and save any changes. You can use the following methods during the editing phase:

**Methods for editing a document**

Method	Function
<i>print_document</i>	Prints the document
<i>clear_selection</i>	Deletes marked text

Working With Documents

<i>paste_clipboard</i>	Inserts the contents of the clipboard
<i>cut_selection</i>	Cuts marked text
<i>paste_special</i>	Inserts a link
<i>select_all</i>	Marks all text
<i>copy_selection</i>	Copies marked text
<i>execute_macro</i>	Executes a macro
<i>save_as</i>	Saves document locally (on the presentation server)
<i>save_copy_as</i>	Saves the document under another name
<i>page_setup</i>	Determines the page layout

- You can find out the attributes of the instance document using the following methods:

**Methods for determining the attributes of a document**

Method	Function
<i>is_open</i>	Document is open in the office application
<i>is_destroyed</i>	Document is not open (empty container)
<i>get_document_type</i>	Type of document
<i>has_changed</i>	Document has changed
<i>get_document_handle</i>	Determines the handle of a document
<i>has_standard_interface</i>	Determines whether the document has a standard interface

**Closing**

- When you have edited the document, use the method *close\_document* to close the document or react to the event *on\_close\_event* that is triggered when the office application user closes the document.
- Save the changes in the document document using the method *save\_document\_to\_url*. Then release the memory occupied using the method *release\_document*.



When you develop your application, you must bear in mind that the user can always close the office application while the R/3 System is busy with other tasks. If this happens, the event *on\_close\_event* is lost. Therefore, when you exit the transaction, you should always check whether there are documents that are still open, and whether these have changed.

Make sure that you save the changes in the document before releasing the document with the *release\_document* method. Use the relevant methods on the instance document.



If you want to use special methods of the desktop application, you can use one or more of the following interfaces:

[The Word Processor Interface \[Page 80\]](#)

[The Form Interface \[Page 91\]](#)

[The Mail Merge Interface \[Page 98\]](#)

[The Script Collection \[Page 103\]](#)

[The Tables Interface \[Page 106\]](#)



Remember that you should include [error handling \[Page 15\]](#) after each method call.

Methods of the Interface `i_oi_document_proxy`

Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Methods of the Interface `i_oi_document_proxy`

The following sections describe the methods of the interface `i_oi_document_proxy`. They are divided into the methods that all OLE2 desktop applications provide, and those that can also be used through a standard interface. The following tables provide an overview of these methods:

Methods of all OLE2 desktop applications

<a href="#">create_document [Page 35]</a>	<a href="#">view_document [Page 36]</a>
<a href="#">view_document_from_table [Page 37]</a>	<a href="#">open_document [Page 38]</a>
<a href="#">open_document_from_table [Page 39]</a>	<a href="#">open_activex_document [Page 42]</a>
<a href="#">reopen_document [Page 40]</a>	<a href="#">play_document [Page 42]</a>
<a href="#">play_document_from_table [Page 43]</a>	<a href="#">save_document_to_url [Page 44]</a>
<a href="#">save_document_to_table [Page 44]</a>	<a href="#">close_document [Page 46]</a>
<a href="#">has_changed [Page 47]</a>	<a href="#">release_document [Page 47]</a>
<a href="#">is_open [Page 47]</a>	<a href="#">is_destroyed [Page 48]</a>
<a href="#">has_standard_interface [Page 48]</a>	<a href="#">has_wordprocessor_interface [Page 54]</a>
<a href="#">get_wordprocessor_interface [Page 54]</a>	<a href="#">get_document_type [Page 55]</a>
<a href="#">get_document_handle [Page 55]</a>	

Methods of the standard interface:

<a href="#">save_as [Page 45]</a>	<a href="#">save_copy_as [Page 45]</a>
<a href="#">add_menu_bar [Page 49]</a>	<a href="#">add_menu_popup [Page 49]</a>
<a href="#">add_menu_item [Page 50]</a>	<a href="#">delete_menu_item [Page 51]</a>
<a href="#">select_all [Page 51]</a>	<a href="#">clear_selection [Page 51]</a>
<a href="#">cut_selection [Page 51]</a>	<a href="#">copy_selection [Page 52]</a>
<a href="#">paste_clipboard [Page 52]</a>	<a href="#">paste_special [Page 52]</a>
<a href="#">page_setup [Page 53]</a>	<a href="#">print_document [Page 53]</a>
<a href="#">update_document_links [Page 56]</a>	<a href="#">get_application_property [Page 56]</a>
<a href="#">execute_macro [Page 53]</a>	



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'x': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'x'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## create\_document

You use this method to open a new document in the office application.

```
CALL METHOD document->create_document
EXPORTING open_inplace   = open_inplace
          create_view_data = create_view_data
          onsave_macro     = onsave_macro
          startup_macro    = startup_macro
          document_title   = document_title
          no_flush         = no_flush
IMPORTING error          = error
          retcode         = retcode.
```

### Description of parameters

Parameter	Optional	Description
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter <code>inplace = 'X'</code> was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
create_view_data	X	Set this option if you want to save presentation data with the document. This is necessary if you want to use the <i>view_document</i> method.

**view\_document**

onsave_macro	X	<p>The name of a macro that is called if the document has been changed <b>and</b> one of the following conditions applies:</p> <ul style="list-style-type: none"> <li>• The user closes the application (if it is not running in-place)</li> <li>• You use one of the following methods: <ul style="list-style-type: none"> <li>– save_document</li> <li>– reopen_document with <code>do_save = 'X'</code></li> <li>– close_document with <code>do_save = 'X'</code></li> </ul> </li> </ul> <p>The Macro must be stored in the document.</p> <p><b>Note:</b> What constitutes a change to the document differs from application to application:</p> <ul style="list-style-type: none"> <li>• MS Project and MS Excel almost always consider the document to have been changed, so the macro is called even if you have not made any physical changes</li> </ul> <p>In MS Word and MS Powerpoint, you must physically change the document in order for the macro to be called</p>
startup_macro	X	Macro to be run when you open the document.
document_title	X	Title to be displayed in the title bar of the desktop application.



You can only start one document in the R/3 window (in place).



If you open a document as a presentation, the application used to create it does not have to be installed on the presentation server. Due to the restrictions of OLE, only the first page of a document can be saved as a presentation.

**view\_document**

Displays the presentation data saved in a document.

```
CALL METHOD document->view_document
  EXPORTING document_title = document_title
            document_url   = document_url
            open_inplace   = open_inplace
            startup_macro   = startup_macro
            user_info       = user_info
            no_flush        = no_flush
  IMPORTING error          = error
            retcode         = retcode.
```

**Description of parameters**

view\_document\_from\_table

Parameter	Optional	Description
document_url		The URL address of the document
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
startup_macro	X	Macro to be run when you open the document.
document_title	X	Title to be displayed in the title bar of the desktop application.
user_info	X	Logon and proxy data for accessing an HTTP or FTP server.



You can only start one document in the R/3 window.



If you want to use the *view\_document* method to display documents, you must set the *create\_view\_data* flag when you use the *create\_document* method to create them. To display other documents, use the *open\_document* method with the *open\_readonly* flag. If you want to do this, the desktop application used to create the document must be installed on the presentation server.

URLs of documents in the local file system must be prefaced with 'file://'.

## view\_document\_from\_table

Displays the presentation data saved in a document. The document must be stored in an internal table.

```
CALL METHOD document->view_document_from_table
EXPORTING document_table = document_table
          document_size   = document_size
          document_title  = document_title
          open_inplace    = open_inplace
          startup_macro   = startup_macro
          no_flush        = no_flush
IMPORTING error          = error
          retcode         = retcode.
```

### Description of parameters

Parameter	Optional	Description
document_table		Internal table containing the document.
document_size		File size.
document_title	X	Title to be displayed in the title bar of the desktop application.

**open\_document**

open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
startup_macro	X	Macro to be run when you open the document.



You can only start one document in the R/3 window.



If you want to use the *view\_document* method to display documents, you must set the *create\_view\_data* flag when you use the *create\_document* method to create them. To display other documents, use the *open\_document* method with the *open\_readonly* flag. If you want to do this, the desktop application used to create the document must be installed on the presentation server.

## open\_document

You use this method to open an existing document:

```
CALL METHOD document->open_document
EXPORTING document_title = document_title
          document_url    = document_url
          open_inplace    = open_inplace
          open_readonly   = open_readonly
          onsave_macro    = onsave_macro
          startup_macro   = startup_macro
          protect_document = protect_document
          user_info       = user_info
          no_flush        = no_flush
IMPORTING error          = error
          retcode         = retcode.
```

### Description of parameters

Parameter	Optional	Description
document_title	X	Title to be displayed in the title bar of the desktop application.
document_url		The URL address of the document
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
open_readonly	X	Open in display mode (the close_event event is not triggered when the document is closed, and the parameter <a href="#">has_changed [Page 47]</a> is always initial).

**open\_document\_from\_table**

onsave_macro	X	<p>The name of a macro that is called if the document has been changed <b>and</b> one of the following conditions applies:</p> <ul style="list-style-type: none"> <li>• The user closes the application (if it is not running in-place)</li> <li>• You use one of the following methods: <ul style="list-style-type: none"> <li>– save_document</li> <li>– reopen_document with <b>do_save = 'X'</b></li> <li>– close_document with <b>do_save = 'X'</b></li> </ul> </li> </ul> <p>The Macro must be stored in the document.</p> <p><b>Note:</b> What constitutes a change to the document differs from application to application:</p> <ul style="list-style-type: none"> <li>• MS Project and MS Excel almost always consider the document to have been changed, so the macro is called even if you have not made any physical changes</li> <li>• In MS Word and MS Powerpoint, you must physically change the document in order for the macro to be called</li> </ul>
startup_macro	X	Macro to be run when you open the document.
protect_document	X	Flags whether to protect document against changes. It is still possible to select within the document.
user_info	X	Logon and proxy data for accessing an HTTP or FTP server.



You can only start one document in the R/3 window (in place).

URLs of documents in the local file system must be prefaced with 'file://'.

## open\_document\_from\_table

You use this method to open an existing document. The document must be stored in the ABAP program in an internal table.

```
CALL METHOD document->open_document_from_table
EXPORTING document_table = document_table
          document_size   = document_size
          document_title  = document_title
          open_inplace    = open_inplace
          open_readonly   = open_readonly
          onsave_macro    = onsave_macro
          startup_macro   = startup_macro
          protect_document = protect_document
          no_flush        = no_flush
IMPORTING error          = error
          retcode         = retcode.
```

**reopen\_document****Description of parameters**

Parameter	Optional	Description
document_table		Internal table containing the document.
document_size		File size.
document_title	X	Title to be displayed in the title bar of the desktop application.
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
open_readonly	X	Open in display mode (the close_event event is not triggered when the document is closed, and the parameter <a href="#">has_changed [Page 47]</a> is always initial).
onsave_macro	X	The name of a macro that is called if the document has been changed <b>and</b> one of the following conditions applies: <ul style="list-style-type: none"> <li>The user closes the application (if it is not running in-place)</li> <li>You use one of the following methods: <ul style="list-style-type: none"> <li>save_document</li> <li>reopen_document with do_save = 'X'</li> <li>close_document with do_save = 'X'</li> </ul> </li> </ul> The Macro must be stored in the document. <b>Note:</b> What constitutes a change to the document differs from application to application: <ul style="list-style-type: none"> <li>MS Project and MS Excel almost always consider the document to have been changed, so the macro is called even if you have not made any physical changes</li> </ul> In MS Word and MS Powerpoint, you must physically change the document in order for the macro to be called
startup_macro	X	Macro to be run when you open the document.
protect_document	X	Flags whether to protect document against changes. It is still possible to select within the document.



You can only start one document in the R/3 window (in place).

**reopen\_document**

When you close a document, it remains in memory on the presentation server. You can reopen it using the *reopen\_document* method.

```
CALL METHOD document->reopen_document
EXPORTING open_inplace = open_inplace
          open_readonly = open_readonly
          do_save       = do_save
          onsave_macro  = onsave_macro
          startup_macro = startup_macro
          no_flush      = no_flush
IMPORTING has_changed  = has_changed
          error         = error
          retcode       = retcode.
```

**Description of parameters**

Parameter	Optional	Description
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
open_readonly	X	Open in display mode (the close_event event is not triggered when the document is closed, and the parameter <a href="#">has_changed [Page 47]</a> is always initial).
do_save	X	Determines whether the document should be saved
onsave_macro	X	The name of a macro that is called if the document has been changed <b>and</b> one of the following conditions applies: <ul style="list-style-type: none"> <li>The user closes the application (if it is not running in-place)</li> <li>You use one of the following methods: <ul style="list-style-type: none"> <li>save_document</li> <li>reopen_document with do_save = 'X'</li> <li>close_document with do_save = 'X'</li> </ul> </li> </ul> <p>The Macro must be stored in the document.</p> <p><b>Note:</b> What constitutes a change to the document differs from application to application:</p> <ul style="list-style-type: none"> <li>MS Project and MS Excel almost always consider the document to have been changed, so the macro is called even if you have not made any physical changes</li> </ul> <p>In MS Word and MS Powerpoint, you must physically change the document in order for the macro to be called</p>
startup_macro	X	Macro to be run when you open the document.
has_changed		Flags whether the office application document was changed If you set the open_readonly parameter when you opened the document, the value of has_changed is always initial.



You can only start one document in the R/3 window (in place).

**open\_activex\_document**

## open\_activex\_document

ActiveX document technology is an extension of OLE technology. It allows you to display documents from any office application that supports the ActiveX documents interface in Microsoft Internet Explorer. You can also use Visual Basic 5.0 to design forms as ActiveX documents, which you can then open in Internet Explorer.

The method *open\_activex\_document* allows you to open an ActiveX document, created using Visual Basic, in the office application.

```
CALL METHOD document->open_activex_document
  EXPORTING document_type = document_type
            document_url  = document_url
            user_info     = user_info
            no_flush      = no_flush
  IMPORTING error        = error
            retcode      = retcode.
```

**Description of parameters**

Parameter	Optional	Description
document_type	X	Document type ProgId of the Visual Basic application
document_url	X	The URL address of the document
user_info	X	Logon and proxy data for accessing an HTTP or FTP server.

## play\_document

You use this method to play a presentation.

```
CALL METHOD document->play_document
  EXPORTING document_title = document_title
            document_url   = document_url
            open_inplace   = open_inplace
            startup_macro  = startup_macro
            protect_document = protect_document
            user_info      = user_info
            no_flush       = no_flush
  IMPORTING error         = error
            retcode       = retcode.
```

**Description of parameters**

Parameter	Optional	Description
document_title	X	Title to be displayed in the title bar of the desktop application.

**play\_document\_from\_table**

document_url		The URL address of the document
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
startup_macro	X	Macro to be run when you open the document.
protect_document	X	Flags whether to protect document against changes. It is still possible to select within the document.
user_info	X	Logon and proxy data for accessing an HTTP or FTP server.



You can only start one document in the R/3 window (in place).

URLs of documents in the local file system must be prefaced with 'file://'.

## play\_document\_from\_table

You use this method to play a presentation. The presentation must be stored in an internal table.

```
CALL METHOD document->play_document_from_table
EXPORTING document_table = document_table
          document_size   = document_size
          document_title   = document_title
          open_inplace     = open_inplace
          startup_macro     = startup_macro
          protect_document = protect_document
          no_flush         = no_flush
IMPORTING error           = error
          retcode          = retcode.
```

### Description of parameters

Parameter	Optional	Description
document_table		Internal table containing the document.
document_size		File size.
document_title	X	Title to be displayed in the title bar of the desktop application.
open_inplace	X	'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <a href="#">get_document_proxy [Page 27]</a> ). ' ': Document is opened in a separate window.
startup_macro	X	Macro to be run when you open the document.
protect_document	X	Flags whether to protect document against changes. It is still possible to select within the document.



You can only start one document in the R/3 window (in place).

**save\_document\_to\_url****save\_document\_to\_url**

You use this method to save a document to a URL address. It enables you to store documents in a central repository.

```
CALL METHOD document->save_document_to_url
EXPORTING url          = url
          user_info    = user_info
          no_flush     = no_flush
IMPORTING error        = error
          retcode      = retcode
CHANGING document_size = document_size.
```

**Description of parameters**

Parameter	Optional	Description
url		The URL address of the document to be saved
user_info	X	Logon and proxy data for accessing an HTTP or FTP server.
document_size	X	The length of the saved document is placed in this field.



Close the document in the office application (using the method *close\_document*) before calling this method. This ensures that the document cannot be changed after you have saved it. You can also use this method to make a backup copy of a document.

After saving, release the memory occupied by the document by calling the *release\_document* method.

URLs of documents in the local file system must be prefaced with 'file://'.

**save\_document\_to\_table**

Saves an office document in an internal table.

```
CALL METHOD document->save_document_to_table
EXPORTING no_flush    = no_flush
IMPORTING error       = error
          retcode     = retcode
CHANGING document_table = document_table
          document_size = document_size.
```

**Description of parameters**

Parameter	Optional	Description
document_table		Internal table in which you want to save the document.
document_size		The length of the saved document is placed in this field.



Close the document in the office application (using the method *close\_document*) before calling this method. This ensures that the document cannot be changed after you have saved it. You can also use this method to make a backup copy of a document.

After saving, release the memory occupied by the document by calling the *release\_document* method.

## save\_as

You use this method to save the document on the local presentation server

```
CALL METHOD document->save_as
  EXPORTING file_name = file_name
           prompt_user = prompt_user
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
file_name	X	The name of the file to be created
prompt_user	X	When you execute the method, it starts the relevant dialog box in the office application.



You should only use this method to create a local copy of the document on the presentation server. Note that this method only works while the document is open. To save the document in a document repository, use the method *save\_document\_to\_url*.

## save\_copy\_as

You use this method to save an office document locally under another name

```
CALL METHOD document->save_as
  EXPORTING file_name = file_name
           prompt_user = prompt_user
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
file_name	X	The name under which the office document is to be saved

**close\_document**

prompt_user	X	When you execute the method, it starts the relevant dialog box in the office application.
-------------	---	---



You should only use this method to create a local copy of the document on the presentation server. Note that this method only works while the document is open. To save the document in a document repository, use the method *save\_document\_to\_url*.

**close\_document**

You use this method to close a document in the office application

```
CALL METHOD document->close_document
EXPORTING do_save   = do_save
          no_flush  = no_flush
IMPORTING has_changed = has_changed
          error     = error
          retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
do_save	X	Determines whether the document should be saved
has_changed	X	Flags whether the office application document was changed



When you have executed the method *close\_document* without saving the document, you must always execute the method *release\_document* to initialize the document.

You can also call the *close\_document* method even if the document is already closed. This does not return an error, and in this way you can retrieve a correct value for the *has\_changed* parameter. This is safer and quicker than calling *is\_open* followed by *close\_document*.

**close\_activex\_document**

Use this method to close an ActiveX document that you created using Visual Basic.

```
CALL METHOD document->close_activex_document
EXPORTING no_flush = no_flush
IMPORTING error    = error
          retcode  = retcode.
```

## has\_changed

You use this method to query whether the document of the instance document was changed in the office application:

```
CALL METHOD document->has_changed
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode   = retcode
           ret_value = ret_value.
```

### Description of parameters

Parameter	Optional	Description
ret_value		Flags whether the document was changed This value is only set if the user has changed the document and the open_readonly parameter was set to initial when you opened the document.



Since the *close\_document* method also returns information about whether the document has changed, you will very rarely have to use the *has\_changed* method.

## release\_document

Even when you have closed a document, its memory area remains occupied. You use this method to release this memory area.

```
CALL METHOD document->release_document
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode   = retcode.
```



After this call, release the ABAP Objects instance:

```
FREE document.
```

## is\_open

You use this method to query whether a document has already been opened for the instance document:

```
CALL METHOD document->is_open
  EXPORTING no_flush = no_flush
  IMPORTING ret_value = ret_value
           error      = error
           retcode    = retcode.
```

### Description of parameters

**is\_destroyed**

Parameter	Optional	Description
ret_value		Indicates whether the document is open



You should use a flag in your application program to mark whether the application is still open. Only use this method if you are unsure whether the desktop application is still open (for example, if a *close\_event* event might have been lost).

You can also call the *close\_document* method even if the document is already closed. The method does not return any errors. This is safer and quicker than calling *is\_open* followed by *close\_document*.

**is\_destroyed**

You use this method to find out whether the instance document has already been destroyed using the [release\\_document \[Page 47\]](#) method.

```
CALL METHOD document->is_destroyed
  IMPORTING ret_value = ret_value.
```

**Description of parameters**

Parameter	Optional	Description
ret_value		Flags whether the document has been destroyed

**has\_standard\_interface**

You use this method to query whether the instance document has a standard interface. In addition to basic functions (open, close, save), the standard interface also supports additional functions (select, copy, ...).

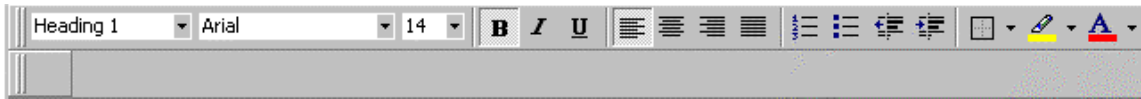
```
CALL METHOD document->has_standard_interface
  EXPORTING no_flush = no_flush
  IMPORTING is_available = is_available
           error = error
           retcode = retcode.
```

**Description of parameters**

Parameter	Optional	Description
is_available		Flags whether a standard interface is available

## add\_menu\_bar

You use this method to generate an extra menu bar in the office application:



The diagram shows the result of the method call. A new menu bar has been added beneath the application-specific toolbar. Currently, this is empty. You can fill it using the method *add\_menu\_popup*.

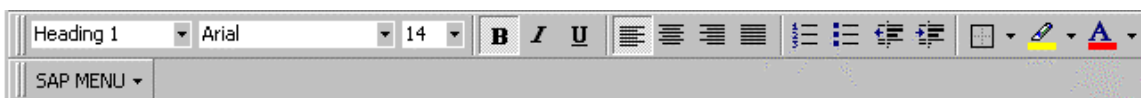
```
CALL METHOD document->add_menu_bar
  EXPORTING name = name
           pos   = pos
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		The name of the new menu bar
pos	X	Not currently used

## add\_menu\_popup

You use this method to add a new entry to an existing menu bar that you have defined using the *add\_menu\_bar* method.



```
CALL METHOD document->add_menu_popup
  EXPORTING name = name
           menu_bar_name = menu_bar_name
           insert_before_popup = insert_before_popup
           parent = parent
           popup_before = popup_before
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		The name of the new menu

**add\_menu\_item**

menu_bar_name	X	The name of the menu bar where the menu will be added
insert_before_popup	X	The position of the item before which the new menu will be added (0: Place at end)
parent	X	Not used
popup_before	X	Not used



If you do not specify the name of a menu bar, the menu is placed in the main menu bar of the desktop application.

## add\_menu\_item

You use this method to generate a new menu entry in a user-defined menu. You must assign this menu entry to an existing macro in the desktop application.



```
CALL METHOD document->add_menu_item
EXPORTING item_name      = item_name
          item_macro     = item_macro
          menu_bar_name  = menu_bar_name
          menu_popup_name = menu_popup_name
          insert_before_item = insert_before_item
          item_pos       = item_pos
          item_before    = item_before
          no_flush       = no_flush
IMPORTING error          = error
          retcode        = retcode.
```

**Description of parameters**

Parameter	Optional	Description
item_name		The name of the new menu option
item_macro		The name of the macro to be executed when the function is chosen
menu_bar_name	X	Name of the menu bar containing the menu menu_popup_name.
menu_popup_name	X	Name of the menu to which you want to add the menu entry.
insert_before_item	X	Position of the existing menu entry before which you want to add the new entry (0: Add at end).
item_before	X	Not used
item_pos	X	Not used



If you do not specify the name of a menu bar, the menu is placed in the main menu bar of the desktop application.

## delete\_menu\_item

You use this method to delete a menu option in the office application:

```
CALL METHOD document->delete_menu_item
  EXPORTING item_name      = item_name
           menu_bar_name  = menu_bar_name
           menu_popup_name = menu_popup_name
           item_pos       = item_pos
           no_flush       = no_flush
  IMPORTING error         = error
           retcode       = retcode.
```

### Description of parameters

Parameter	Optional	Description
item_name		The name of the new menu option
menu_bar_name	X	Name of the menu bar containing the menu menu_popup_name.
menu_popup_name		Name of the menu from which you want to delete the menu entry.
item_pos	X	This parameter is obsolete. Do not use it in new programs.



If you do not specify the name of a menu bar, the menu is deleted from the main menu bar of the desktop application.

## select\_all

You use this method to mark all text in the office document.

```
CALL METHOD document->select_all
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode   = retcode.
```

## clear\_selection

You use this method to deselect all marked text:

```
CALL METHOD document->clear_selection
  EXPORTING no_flush = no_flush
```

## cut\_selection

```
IMPORTING error = error
         retcode = retcode.
```

## cut\_selection

You use this method to cut all marked text in the office application:

```
CALL METHOD document->cut_selection
  EXPORTING no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

## copy\_selection

You use this method to copy marked text from the office document to the clipboard:

```
CALL METHOD document->copy_selection
  EXPORTING no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

## paste\_clipboard

You use this method to insert the contents of the clipboard at the current cursor position. The clipboard contents are inserted as fixed text; there is no link to the object:

```
CALL METHOD document->paste_clipboard
  EXPORTING no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

## paste\_special

You use this method to call the dialog box for inserting the contents of the clipboard in the office document

```
CALL METHOD document->paste_special
  EXPORTING prompt_user = prompt_user
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
prompt_user	X	Flags whether the method should call the corresponding dialog box in the office application when it is executed.



This method should only be executed when the user responds to the dialog box in the office application. Otherwise, only fixed values and not links will be inserted in the document.

## page\_setup

You use this method to call the dialog box for office application page layout

```
CALL METHOD document->page_setup
  EXPORTING no_flush      = no_flush
  IMPORTING error        = error
           retcode       = retcode.
```

## print\_document

You use this method to print a document that is open in the office application

```
CALL METHOD document->print_document
  EXPORTING prompt_user = prompt_user
           no_flush     = no_flush
  IMPORTING error       = error
           retcode      = retcode.
```

### Description of parameters

Parameter	Optional	Description
prompt_user	X	Flags whether the method should call the corresponding dialog box in the office application when it is executed

## execute\_macro

You use this method to execute a macro in the office application

```
CALL METHOD document->execute_macro
  EXPORTING macro_string = macro_string
           param_count   = param_count
           script_name   = script_name
           param1        = param1
           param2        = param2
           param3        = param3
           param4        = param4
           param5        = param5
           param6        = param6
           param7        = param7
           param8        = param8
```

**has\_wordprocessor\_interface**

```

param9      = param9
param10     = param10
param11     = param11
param12     = param12
no_flush    = no_flush
IMPORTING error      = error
           retcode   = retcode
CHANGING error_string = error_string
           retvalue  = retvalue.
    
```

**Description of parameters**

Parameter	Optional	Description
macro_string		The name of the macro in the office application
script_name	X	' ': The macro is part of the document <name>: The macro is placed in the script collection under this name.
param_count	X	Number of parameters to be passed to the macro
param1.. param9	X	The parameters to be passed to the macro (not Word 97)
error_string	X	The internal error code from the application macro is returned to the ABAP side, and can be handled there. This suppresses the normal VB dialog box.
retvalue	X	Value generated by the application macro.



If you are using Word, you cannot pass any parameters to the macro.

You can also specify the name of a template, add-in, or VBA module in the form 'template\_name.macro\_name' or 'module\_name.macro\_name'.

## has\_wordprocessor\_interface

You use this method to query whether there is an [application-specific interface \[Page 80\]](#) for the document type of the instance document:

```

CALL METHOD document->has_wordprocessor_interface
EXPORTING no_flush    = no_flush
IMPORTING is_available = is_available
           error      = error
           retcode    = retcode.
    
```

**Description of parameters**

Parameter	Optional	Description
is_available		Flags whether a standard interface is available

## get\_wordprocessor\_interface

You use this method to create the instance wp\_interface to manage the [application-specific interface \[Page 80\]](#) :

```
CALL METHOD document->get_wordprocessor_interface
  EXPORTING no_flush = no_flush
  IMPORTING wp_interface = wp_interface
           error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
wp_interface		The instance created for the application-specific interface

## get\_document\_type

You use this method to query the document types managed by the instance document:

```
CALL METHOD document->get_document_type
  EXPORTING no_flush = no_flush
  IMPORTING document_type = document_type
           error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
document_type		Document type



The actual document type may, in some cases, not be the same as the type returned by *get\_document\_proxy*.

## get\_document\_handle

You use this method to query the handle of the instance document. This allows you to access the document directly in the relevant application using SAP Automation.

```
CALL METHOD document->get_document_handle
  EXPORTING no_flush = no_flush
  IMPORTING handle = handle
           error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
-----------	----------	-------------

**get\_application\_property**

handle		The handle of instance document
--------	--	---------------------------------



The handle returned by the method points to the document object. The function modules CONTROL\_CALL\_METHOD, CONTROL\_GET\_PROPERTY, and CONTROL\_SET\_PROPERTY allow you to call methods of the document object and find out or set its properties. However, when you use these function modules, you are programming directly against the object model of the relevant desktop application. Consequently, you will have to adapt your program to each desktop application that you use.

**get\_application\_property**

Retrieves the properties of a desktop application. This method allows you to find out, for example, user-specific settings in the desktop application , such as the default directory for templates in Word.

```
CALL METHOD document->get_application_property
EXPORTING property_name = property_name
          subproperty_name = subproperty_name
          no_flush = no_flush
IMPORTING error = error
          retcode = retcode
CHANGING retvalue = retvalue.
```

**Description of parameters**

Parameter	Optional	Description
property_name		Name of the property you want to retrieve
subproperty_name	<b>X</b>	Defines the property further
retvalue		Value of the property

Possible values:

Property_Name	Subproperty_Name	Description
'TEMPLATE_PATH'	"	The directory path for user-specific templates in a word processing program
'INTERNATIONAL'	'DECIMAL_SEPARATOR'	Character for decimal separator
'INTERNATIONAL'	'THOUSANDS_SEPARATOR'	Character for thousands separator
'INTERNATIONAL'	'LIST_SEPARATOR'	Separator character for lists
'INTERNATIONAL'	'DATE_SEPARATOR'	Separator character for dates
'INTERNATIONAL'	'TIME_SEPARATOR'	Separator character for times

## update\_document\_links

If the desktop application that you are using does not automatically refresh the links in a document when you open it, you can use this method to refresh them manually.

```
CALL METHOD document->update_document_links
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode   = retcode.
```

## has\_spreadsheet\_interface

You use this method to query whether there is a [tables interface \[Page 106\]](#) for the document type of the instance document:

```
CALL METHOD document->has_spreadsheet_interface
  EXPORTING no_flush = no_flush
  IMPORTING is_available = is_available
           error      = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
is_available		Flags whether a tables interface is available

## get\_spreadsheet\_interface

Creates an instance `sheet_interface` for managing the [tables interface \[Page 106\]](#).

```
CALL METHOD document->get_spreadsheet_interface
  EXPORTING no_flush = no_flush
  IMPORTING sheet_interface = sheet_interface
           error      = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
sheet_interface		Name of the generated instance for the tables interface. Object with reference to interface <code>i_oi_spreadsheet</code> .

**has\_form\_interface****has\_form\_interface**

You use this method to query whether there is a [form interface \[Page 91\]](#) for the document type of the instance document:

```
CALL METHOD document->has_form_interface
  EXPORTING no_flush = no_flush
  IMPORTING is_available = is_available
           error = error
           retcode = retcode.
```

**Description of parameters**

Parameter	Optional	Description
is_available		Flags whether a form interface is available

**get\_form\_interface**

You use this method to create the instance f\_interface to manage the [form interface \[Page 91\]](#) :

```
CALL METHOD document->get_form_interface
  EXPORTING no_flush = no_flush
  IMPORTING f_interface = f_interface
           error = error
           retcode = retcode.
```

**Description of parameters**

Parameter	Optional	Description
f_interface		The instance created for the form interface Object with reference to interface <code>i_oi_form</code> .

**has\_mail\_merge\_interface**

You use this method to query whether there is a [mail merge interface \[Page 98\]](#) for the document type of the instance document:

```
CALL METHOD document->has_mail_merge_interface
  EXPORTING no_flush = no_flush
  IMPORTING is_available = is_available
           error = error
           retcode = retcode.
```

**Description of parameters**

Parameter	Optional	Description
-----------	----------	-------------

**get\_mail\_merge\_interface**

is_available		Flags whether a tables interface is available
--------------	--	---

## get\_mail\_merge\_interface

Creates an instance `mm_interface` for administering the [mail merge interface \[Page 98\]](#).

```
CALL METHOD document->get_mail_merge_interface
  EXPORTING no_flush = no_flush
  IMPORTING mm_interface = mm_interface
           error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
mm_interface		Name of the generated instance for the mail merge interface. Object with reference to interface <code>i_oi_mail_merge</code> .

## has\_pivot\_interface

You use this method to find out whether the document type of the instance document supports the [pivot interface \[Page 142\]](#).

```
CALL METHOD document->has_pivot_interface
  EXPORTING no_flush = no_flush
  IMPORTING is_available = is_available
           error = error
           retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
is_available		Flag to indicate whether the pivot interface is available

## get\_pivot\_interface

You use this method to get an instance `pivot_interface` for administering the [pivot interface \[Page 142\]](#).

```
CALL METHOD document->get_pivot_interface
  EXPORTING no_flush = no_flush
```

**get\_script\_collection**

```

IMPORTING pivot_interface = pivot_interface
          error           = error
          retcode         = retcode.
    
```

**Description of parameters**

Parameter	Optional	Description
pivot_interface		Reference to an instance of the pivot interface <b>I_OI_PIVOT</b> , which you can use to work with Excel pivot tables from within your R/3 application.

**get\_script\_collection**

Creates an instance wp\_interface for administering the [script collection \[Page 103\]](#).

```

CALL METHOD document->get_script_collection
EXPORTING no_flush = no_flush
IMPORTING scripts = scripts
          error    = error
          retcode  = retcode.
    
```

**Description of parameters**

Parameter	Optional	Description
scripts		Name of the generated instance for the script collection. Object with reference to interface <b>i_oi_script_collection</b> .

**get\_document\_cookie**

This method returns the document cookie (ID of the document). You can use this in a VB add-in to check whether the document exists. If it does, you can then use the methods of the add-in to work with the document.

In the data model of SAP Desktop Office Integration, the document ID is found under:

Application.Documents.Container.DocumentCookie

```

CALL METHOD document->get_document_cookie
IMPORTING document_cookie = document_cookie.
    
```

**Description of parameters**

Parameter	Optional	Description
document_cookie		Document ID

## Processing Office Application Events

### Purpose

You can interpret events triggered by the office application or users of the office application in your ABAP program and react to them. The following events exist:

- `on_custom_event`: The event is triggered by the office application user. You must initialize this event when you call the document
- `on_close_event`: This event is triggered when you close the document to be edited in the office application. This allows you still to save the document.

### Process flow

1. Register the events that you want to interpret in your program. You do this in the method call for [init\\_control \[Page 22\]](#) on the instance control.
2. Create the processing logic for the registered events in a class of your own. You can use static methods or object methods here.
3. After initializing the document (`get_document_proxy` method), define the event handling. You can use static methods and object methods to react to the events.
  - Static methods

```
SET HANDLER c_event_handler=>close_event FOR document.
SET HANDLER c_event_handler=>custom_event FOR document.
```
  - Object methods

```
DATA o_event_handler TYPE REF TO class_event_handler.
CREATE OBJECT o_event_handler.
SET HANDLER o_event_handler->close_event FOR document.
SET HANDLER o_event_handler->custom_event FOR document.
```
4. In each PAI module of your program that can be called while the desktop application is active, include the following method call for dispatching control framework events:

```
CALL METHOD cl_gui_cfw=>dispatch
  IMPORTING return_code = return_code.
```

## Event Handling Using Static Methods

### Prerequisites

When you have declared the events of the office application that you want to react to in your ABAP program, you must create the processing logic for events. You create this in a local class

## Event Handling Using Instance Methods

containing a series of methods (either static or instance methods). Each method covers one process relevant for document processing.

### Procedure

1. Create the class definition, in which you declare the class methods for the event interpretation:

```
CLASS c_event_handler DEFINITION.  
  PUBLIC SECTION.  
  CLASS-METHODS: close_event  
    FOR EVENT ON_CLOSE_DOCUMENT OF I_OI_DOCUMENT_PROXY  
    IMPORTING (...).  
  
  CLASS-METHODS: custom_event  
    FOR EVENT ON_CUSTOM_EVENT OF I_OI_DOCUMENT_PROXY  
    IMPORTING (...).  
ENDCLASS.
```

2. Implement the static methods:

```
CLASS c_event_handler IMPLEMENTATION.  
  METHOD close_event.  
<Processing logic>  
  ENDMETHOD.  
  
  METHOD custom_event.  
<Processing logic>  
  ENDMETHOD.  
ENDCLASS.
```

## Event Handling Using Instance Methods

### Prerequisites

When you have declared the events of the office application that you want to react to in your ABAP program, you must create the processing logic for events. You create this in a local class containing a series of methods (either static or instance methods). Each method covers one process relevant for document processing.

---

**Exchanging Data Between R/3 and the Desktop Application****Procedure**

1. Create the class definition, in which you declare the methods for the event interpretation:

```
CLASS class_event_handler DEFINITION.  
    PUBLIC SECTION.  
        METHODS: close_event  
                FOR EVENT ON_CLOSE_DOCUMENT OF I_OI_DOCUMENT_PROXY  
                IMPORTING (...).  
  
        METHODS: custom_event  
                FOR EVENT ON_CUSTOM_EVENT OF I_OI_DOCUMENT_PROXY  
                IMPORTING (...).  
ENDCLASS.
```

2. Implement the methods:

```
CLASS event_handler IMPLEMENTATION.  
    METHOD close_event.  
<Processing logic>  
    ENDMETHOD.  
  
    METHOD custom_event.  
<Processing logic>  
    ENDMETHOD.  
ENDCLASS.
```

**Exchanging Data Between R/3 and the Desktop Application****Use**

To enable you to use Desktop Office Integration in business applications as well as for storing and managing documents, it contains features that allow you to use documents as forms or templates into which you can insert R/3 data at runtime.

The Link Server

## Integration

### Link Server

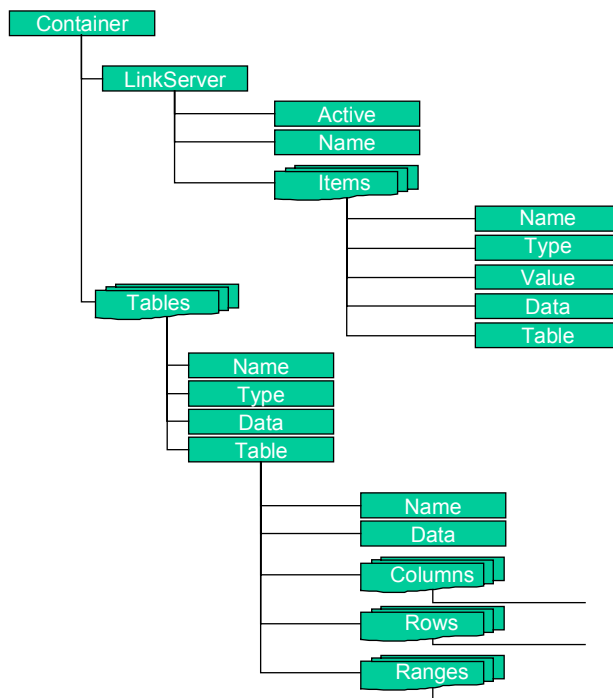
OLE links allow you to insert not only simple text into your documents, but also complex data formats such as RTF, graphics, and internal tables. You can create document templates easily using a 'Paste special' dialog box and without any macro programming knowledge.

### Table Collection

R/3 data in tabular form is made available using the SAP TableFactory OCX. This is part of the SAP Automation Suite, and is used for RFC programming in Visual Basic. This function allows you to re-import data into the R/3 System after you have edited it using the desktop application.

### Features

The following diagram shows the object hierarchy by which R/3 data is exposed. You can access the data using Visual Basic programs.



## The Link Server

### Definition

You define this instance with reference to the interface *i\_oj\_link\_server*.

## Use

The link server manages the objects of the R/3 System (internal tables, fields, documents, graphics,...) that are integrated in the office application when the program is executed. The link server also provides methods for managing and creating links to the office application.

## Integration

Before you can create the instance *link\_server*, you must create the central instance *control* for administering the Desktop Office Integration.

## Working With the Link Server

The link server contains the R/3 System objects that users can incorporate in the office documents. You also save a list of permitted objects on the link server.

## Prerequisites

Before proceeding, you must create the central instance control for managing Office Integration. Refer to [Instance for Managing Desktop Office Integration \[Page 19\]](#).

## Process flow

### Initialization

1. Declare an instance with reference to the interface *i\_oj\_link\_server*:  
data: link\_server type ref to i\_oj\_link\_server.
2. Use the method *get\_link\_server* on the central object for Desktop Office Integration *control* to create an instance for the link server.
3. Use the method *start\_link\_server* in the instance *link\_server* to start the link server

### Document Design Phase

4. Use the methods *add\_string\_item*, *add\_table\_item* and *add\_binary\_item* to add items to the link server. Use the method *remove\_link\_item* to delete items from the link server.



Before opening the document, you must declare all of the links that you will use in it to the link server.

5. The data in the link server is placed on the clipboard of the desktop application using the *copy\_link\_item\_to\_clipboard* method.

### Document Runtime

6. If you have changed the data in your application program, and need to update it in the desktop application, use the methods *add\_string\_item*, *add\_table\_item* and *add\_binary\_item* to update them.
7. Before you leave your program, use the method *stop\_link\_server* to switch off the link server and release the memory area.

## Generic Parameters



The links are only updated when you pass the link that you want to update using one of the methods [add\\_string\\_item \[Page 67\]](#), [add\\_table\\_item2 \[Page 67\]](#), [add\\_table\\_item \[Page 69\]](#), or [add\\_binary\\_item \[Page 70\]](#).



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## start\_link\_server

You use this method to start an instance with reference to the interface `i_oi_link_server`:

```
CALL METHOD link_server->start_link_server
  EXPORTING link_server_mode = link_server_mode
            server_name_suffix = server_name_suffix
            no_flush          = no_flush
  IMPORTING error            = error
            retcode          = retcode.
```

**Description of parameters**

Parameter	Optional	Description
link_server_mode	X	link_server->link_server_inactive: The link server instance is created but not registered (see note). link_server->link_server_standardname: The link server is created with the standard identifier link_server->link_server_customname: The link server is created with the additional identifier server_name_suffix.
server_name_suffix	X	Suffix to name of link server



You should always use *link\_server\_customname* with a suffix. If a link server with the same name (for example, the standard name) is already running, an error occurs when you call the method. See also [Strategies for Working with the Link Server \[Page 72\]](#).

Note that links in all OLE documents opened in the SAP Document Container Control must be created with this server name.



If you specify *link\_server->link\_server\_inactive* as a link server mode, the system creates an instance of the link server in the SAP Document Container Control. You can then add items to the link server using *add\_xxx\_item*. These items are only accessible using automation. However, you can also release the data for linking later on by calling *start\_link\_server* with another session.

## stop\_link\_server

You use this method to destroy the object of the link server.

```
CALL METHOD link_server->stop_link_server
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode  = retcode.
```

## add\_string\_item

You use this method to add a data field or structure to the link server.

```
CALL METHOD link_server->add_string_item
  EXPORTING item_name = item_name
           item_title = item_title
           item_value = item_value
           no_flush   = no_flush
  IMPORTING error    = error
           retcode  = retcode.
```

**Description of parameters**

**add\_table\_item2**

Parameter	Optional	Description
item_name		The technical name in the link server
item_title	X	Describes the field (for translatable texts)
item_value		The value of the data field

## add\_table\_item2

You use this method to add an internal table to the link server.



This method is different from [add\\_table\\_item \[Page 69\]](#) in that, in this case, you do not have to call the Data Provider yourself.

```
CALL METHOD link_server->add_table_item2
EXPORTING item_name = item_name
          item_title = item_title
          ddic_name  = ddic_name
          description = description
          date       = date
          time       = time
          no_flush   = no_flush
IMPORTING error     = error
          retcode    = retcode
CHANGING data_table = data_table
          fields_table = fields_table
          properties  = properties.
```

### Description of parameters

Parameter	Optional	Description
item_name		The name of the internal table in the ABAP program
item_title	X	Describes the table
ddic_name	X	Name of the table or structure in the ABAP Dictionary.
description	X	Optional description
date	X	Date of last change
time	X	Time of last change
data_table		Data table
fields_table	X	Description of the table structure (if you have not already passed it using the <code>ddic_name</code> parameter). Create the table with reference to the type <code>SOI_FIELDS_TABLE</code> .
properties	X	Additional properties Create the table with reference to the type <code>SOI_PROPERTIES_TABLE</code> .



If you specify an internal table whose structure is defined in the ABAP Dictionary, enter the name of the structure as the parameter *ddic\_name*. This transfers the column names to the SAPgui along with the table, so that you can address a column by its ABAP Dictionary name in a macro.

However, if the table structure is defined in a type pool or directly in the program, the system generates numbers for the column names. If you then still want to address a column using a meaningful name, you must program the name transfer yourself. To do this, create an extra table to contain the column information. Then use the function module `DP_GET_FIELDS_FROM_TABLE` to retrieve the column descriptions:

```
CALL FUNCTION 'DP_GET_FIELDS_FROM_TABLE'
  TABLES
    DATA      = SALES_TABLE
    FIELDS     = FIELDS_TABLE.
```

You can then use your own column names, as shown in the following example:

```
READ TABLE FIELDS_TABLE INDEX 1.
FIELDS_TABLE-FIELDNAME = 'Region'. "#EC NOTEXT
MODIFY FIELDS_TABLE INDEX 1.
READ TABLE FIELDS_TABLE INDEX 2.
FIELDS_TABLE-FIELDNAME = 'Sales'. "#EC NOTEXT
MODIFY FIELDS_TABLE INDEX 2.
```

You can now use the table with column information when you call the method *add\_table\_item2*.



You cannot add structures directly to the link server. However, you can create a single-line table and send it to the link server.

## add\_table\_item

You use this method to add an internal table to the link server.

```
CALL METHOD link_server->add_table_item
  EXPORTING item_name  = item_name
            item_title = item_title
            column_count = column_count
            line_count  = line_count
            table_url   = table_url
            no_flush    = no_flush
  IMPORTING error      = error
            retcode    = retcode.
```

**add\_binary\_item****Description of parameters**

Parameter	Optional	Description
item_name		The name of the internal table in the ABAP program
item_title	X	Describes the table
table_url		The location of an internal table when it has been downloaded from the presentation server (location created for example using the function module 'DP_CREATE_URL_FROM_TABLE')
column_count	X	Number of columns
line_count	X	Number of lines



**You must download the contents of the internal table form the presentation server. Use a function module to maintain data in the data provider. The URL address is then transferred to the link server.**

**add\_binary\_item**

You use this method to add an binary object (graphic, document,...) to the link server. You must determine the URL address of the object in advance.

```
CALL METHOD link_server->add_binary_item
  EXPORTING item_name = item_name
            item_title = item_title
            data_type  = data_type
            data_subtype = data_subtype
            user_info  = user_info
            table_url  = table_url
            no_flush   = no_flush
  IMPORTING error      = error
            retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
item_name		The name of the object
item_title	X	Describes the object
table_url		The URL address of the object
data_type	X	Type of data as a MIME type (for example, application)
data_subtype	X	Subtype of data as a MIME type (for example, x-rtf)
user_info	X	Logon and proxy data for accessing an HTTP or FTP server.



The possible data types are contained in the type group **SOI**.

## remove\_link\_item

You use this method to delete items from the link server

```
CALL METHOD link_server->remove_link_item
  EXPORTING item_name = item_name
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
item_name		Technical name of the object to be deleted

## copy\_link\_item\_to\_clipboard

You use this method to create a link to an entry in the link server and store the link in the clipboard of the presentation server. Use the insert function to transfer this link to the open document in the office application. In this way, you can integrate both the fixed value and the link to the object in the document.

```
CALL METHOD link_server->copy_link_item_to_clipboard
  EXPORTING item_name = item_name
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
item_name		Technical name of the object to which a link is to be placed on the clipboard



**The user can create links to individual cells, columns, or rows of tables using the following additions:**

**RnCM:**     **Selects the cell with row number n and column number m.**

RnC\$Name:     Selects the cell with the line number n in column 'name'

**R0Cm:**     **Selects an entire column with column number m.**

R0C\$Name:     Selects an entire column with the name 'name'.

RnC0:         Selects an entire row with row number n.



Refer to the notes under the method *add\_table\_item2* about specifying column names.

execute\_copy\_link\_dialog

## execute\_copy\_link\_dialog

You use this method to call a dialog box in R/3 in which you select a link to the link server. The link is then stored in the office application clipboard.

```
CALL METHOD link_server->stop_link_server
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode   = retcode.
```

## Strategies for Working with the Link Server

Remember that when you work with the link server, it is entered in the list of services in Windows. Since all services must be unique, you cannot insert the link server twice with the same name.

### This Causes the Following Problems:

When you generate an instance of a link server (*start\_link\_server* method), the system checks whether a link server with the same name is already registered. If this is the case, the call fails, and the system returns an error message.

This means that problems can arise with the link server if a user is using several transactions at once that use SAP Desktop Office Integration (and hence the link server).

### Solution 1

When you call the method *start\_link\_server*, always use the parameter `link_server_mode = link_server->link_server_custname` and assign an application-related parameter `<server_name_suffix>`.



```
CALL METHOD link_server->START_LINK_SERVER
  EXPORTING LINK_SERVER_MODE = link_server->link_server_custname
           SERVER_NAME_SUFFIX = 'FI'
  RECEIVING RETCODE          = retcode.
```

This means that the links from the document must have the following format:  
 {LINK SAP.OLELinkServer.ItemObject.<n> "SAPOLELinkServer**FI**" "<field>"  
 \<format>}

This solution has the restriction that the user cannot run the same transaction twice simultaneously, since in this case, the same name would be assigned to the memory area twice.

### Solution 2

When you call the method *start\_link\_server*, always use the parameter `link_server_mode = link_server->link_server_custname` and assign a unique name (for example, GUID or ID of the document in the central repository) to the parameter `server_name_suffix`.

This solution has the disadvantage that existing links in an initial document must be adapted:  
 {LINK SAP.OLELinkServer.ItemObject.<n> "SAPOLELinkServer<**GUID**>" "<field>" \<format>}

The unique name must also be saved with the document, since the same suffix must be used when you subsequently restart the link server to display the document again.

## The Table Collection

### Definition

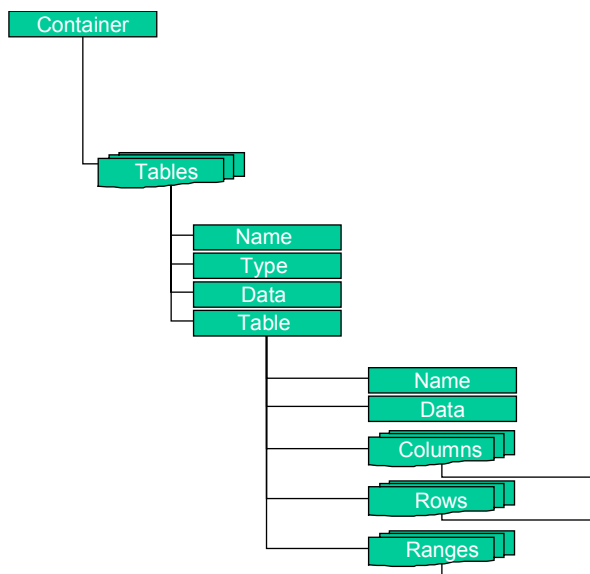
You define this instance with reference to the interface *i\_oi\_table\_collection*.

### Use

The table collection allows you to pass data (especially internal tables) from the R/3 System to the desktop application. The desktop applications can then work with the data and store them in work areas. They can also export them back to the R/3 System.

The data is incorporated in the desktop application using the SAP TableFactory OCX, which is used for RFC programming under Visual Basic.

The object hierarchy of the table collection is as follows:



### Integration

Before you can create the instance `table_collection`, you must create the central instance `control` for administering the Desktop Office Integration.

## Using the Table Collection

# Using the Table Collection

## Purpose

The table collection contains data from the R/3 System that you can include in your office application.

## Prerequisites

Before proceeding, you must create the central instance control for managing Office Integration. Refer to [Instance for Managing Desktop Office Integration \[Page 19\]](#).

## Process flow

1. Declare an instance with reference to the interface *i\_oi\_table\_collection*:  
DATA table\_coll TYPE REF TO i\_oi\_table\_collection.
2. Use the method *get\_table\_collection* on the central object for Desktop Office Integration `control` to create an instance for the table collection.
3. Use the methods *add\_table* and *add\_table\_by\_url* to add data to the table collection.
4. Read and change the table collection data in the desktop application using Visual Basic to access it.
5. Use the methods *get\_table* and *get\_table\_by\_url* to read data from the desktop application back into the R/3 System.
6. Delete the contents of the table collection using the methods *remove\_table* and *remove\_all\_tables*.



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).



The following example shows how you can access data in the table collection using a Visual Basic program.

```
Private Sub Workbook_Activate()  
    Dim R3Table As Object  
    Dim ExcelRange As Excel.Range  
  
    Set R3Table = ThisWorkbook.Container.Tables("SALES_OUT").Table  
  
    Set ExcelRange = Sheet1.Range(Sheet1.Cells(1, 1),  
        Sheet1.Cells(R3Table.Rows.Count, R3Table.Columns.Count))  
  
    ExcelRange.Value = R3Table.Data  
End Sub  
  
Private Sub Chart_SeriesChange (ByVal SeriesIndex As Long, ByVal PointIndex As Long)  
    Dim R3Table As Object  
    Dim Row As Object  
  
    Set R3Table = ThisWorkbook.Container.Tables("SALES_IN").Table  
  
    R3Table.Rows.RemoveAll  
  
    For i = 1 To 6  
        Set Row = R3Table.Rows.Add  
  
        For j = 1 To R3Table.Columns.Count  
            Row.Cell(j) = Sheet1.Cells(i, j).Value  
        Next j  
    Next i  
  
    Call ThisWorkbook.Container.SendCustomEvent ("ON_CHANGE")  
End Sub
```

## Generic Parameters

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## add\_table

You use this method to add an internal table to the table collection.

```
CALL METHOD table_coll->add_table
  EXPORTING table_name = table_name
            table_type = table_type
            ddic_name  = ddic_name
            description = description
            date       = date
            time       = time
            no_flush   = no_flush
  IMPORTING error      = error
            retcode    = retcode
  CHANGING data_table = data_table
            fields_table = fields_table
            properties  = properties.
```

### Description of parameters

Parameter	Optional	Description
table_name		The name of the internal table in the ABAP program

add\_table

table_type		Table type: <b>table_coll-&gt;table_type_input</b> : Passes an empty table to the desktop application. This is used to load data from the application into the R/3 System. <b>table_coll-&gt;table_type_output</b> : Passes a filled table to the desktop application. This is used to pass data from the R/3 System to the application.
ddic_name	X	Link to the ABAP Dictionary
description	X	Description of the table
date	X	Date of last change
time	X	Time of last change
data_table		Table containing data
fields_table	X	Description of the table structure (if you have not already passed it using the <b>ddic_name</b> parameter). Create the table with reference to the type <b>SOI_FIELDS_TABLE</b> .
properties	X	Additional properties Create the table with reference to the type <b>SOI_PROPERTIES_TABLE</b> .



If you specify an internal table whose structure is defined in the ABAP Dictionary, enter the name of the structure as the parameter *ddic\_name*. This transfers the column names to the SAPgui along with the table, so that you can address a column by its ABAP Dictionary name in a macro.

However, if the table structure is defined in a type pool or directly in the program, the system generates numbers for the column names. If you then still want to address a column using a meaningful name, you must program the name transfer yourself. To do this, create an extra table to contain the column information. Then use the function module `DP_GET_FIELDS_FROM_TABLE` to retrieve the column descriptions:

```
CALL FUNCTION 'DP_GET_FIELDS_FROM_TABLE'
  TABLES
    DATA      = SALES_TABLE
    FIELDS     = FIELDS_TABLE.
```

You can then use your own column names, as shown in the following example:

```
READ TABLE FIELDS_TABLE INDEX 1.
FIELDS_TABLE-FIELDNAME = 'Region'. "#EC NOTEXT
MODIFY FIELDS_TABLE INDEX 1.
READ TABLE FIELDS_TABLE INDEX 2.
FIELDS_TABLE-FIELDNAME = 'Sales'. "#EC NOTEXT
MODIFY FIELDS_TABLE INDEX 2.
```

## add\_table\_by\_url

You can now use the table with column information when you call the method `add_table_item2`.

## add\_table\_by\_url

You use this method to add an internal table to the table collection.

```
CALL METHOD table_coll->add_table_by_url
  EXPORTING table_name = table_name
            table_title = table_title
            table_type = table_type
            table_url  = table_url
            no_flush  = no_flush
  IMPORTING error      = error
            retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
table_name		The name of the internal table in the ABAP program
table_title	X	Describes the table
table_type		Table type: <b>table_coll-&gt;table_type_input</b> : Passes an empty table to the desktop application This is used to load data from the application into the R/3 System. <b>table_coll-&gt;table_type_output</b> : Passes a filled table to the desktop application This is used to pass data from the R/3 System to the application.
table_url		The location of an internal table when it has been downloaded to the presentation server (location created for example using the function module 'DP_CREATE_URL_FROM_TABLE')



You must download the contents of the internal table from the presentation server. Use a function module to maintain data in the data provider. The URL address is then transferred to the link server.

## get\_table

You use this method to read a table from the table collection back into an ABAP program.

However, you can only use the method for tables with type `table_coll->table_type_input`.

```
CALL METHOD table_coll->get_table
  EXPORTING table_name = table_name
            no_flush  = no_flush
  IMPORTING description = description
            date       = date
```

**get\_table\_by\_url**

```

time      = time
error     = error
retcode   = retcode
CHANGING ddic_name = ddic_name
data_table = data_table
fields_table = fields_table
properties = properties.
    
```

**Description of parameters**

Parameter	Optional	Description
table_name		The name of the internal table in the ABAP program
description	X	Description of the table
date		Date of last change
time		Time of last change
ddic_name	X	Link to the ABAP Dictionary
data_table		Table containing data
fields_table	X	Description of the table structure (if you have not already passed it using the <b>ddic_name</b> parameter). Create the table with reference to the type <b>SOI_FIELDS_TABLE</b> .
properties	X	Additional properties Create the table with reference to the type <b>SOI_PROPERTIES_TABLE</b> .

## get\_table\_by\_url

You use this method to read a table from the table collection back into an ABAP program. However, you can only use the method for tables with type **table\_coll->table\_type\_input**.

```

CALL METHOD table_coll->get_table_by_url
  EXPORTING table_name = table_name
            table_url  = table_url
            no_flush   = no_flush
  IMPORTING error     = error
            retcode    = retcode.
    
```

**Description of parameters**

Parameter	Optional	Description
table_name		The name of the internal table in the ABAP program
table_url		Memory address of the internal table

**remove\_table****remove\_table**

You use this method to delete a given table from the table collection.

```
CALL METHOD table_coll->remove_table
  EXPORTING table_name = table_name
            no_flush   = no_flush
  IMPORTING error      = error
            retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
item_name		Name of the internal table in the table collection

**remove\_all\_tables**

You use this method to delete the entire contents of table collection.

```
CALL METHOD link_server->remove_all_tables
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
            retcode  = retcode.
```

**The Word Processor Interface****Definition**

You define this instance with reference to the interface *i\_oi\_word\_processor\_document*.

**Use**

You can use the instance *wp* to activate application-specific functions of a word processor.

**Integration**

Before creating the *wp* instance, you must create the *document* instance for managing the document.

## Working With The Word Processor Interface

### Purpose

The word processor interface allows you to address application-specific word processing functions in the office application (if it has a word processor interface) from your ABAP program (refer also to [has\\_wordprocessor\\_interface \[Page 54\]](#))

### Prerequisites

You must create the instance *document* for document management before you can use the interface.

### Process flow

1. Declare an instance with reference to the interface *i\_oi\_word\_processor\_document*:  

```
data: wp type ref to i_oi_word_processor_document.
```
2. Use the method [get\\_wordprocessor\\_interface \[Page 54\]](#) on the document instance for document management to create the instance for the word processor interface.
3. Use the application-specific functions:

#### Methods of the word processor interface

Method	Function
<i>get_version</i>	Returns the version of the word processor application
<i>install_template</i>	Installs a template locally
<i>set_template</i>	Calls an existing template
<i>remove_template</i>	Deletes an existing template
<i>get_template_info</i>	Returns the name and location of the template
<i>search</i>	Searches for a string
<i>replace</i>	Replaces a search string with a new term
<i>insert_table</i>	Inserts the contents of an internal table into a table in the document
<i>insert_table2</i>	Inserts the contents of an internal table into a table in the document
<i>clear_table</i>	Deletes the contents of a table in the document
<i>get_table_info</i>	Returns the size of each table in the word processor document



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## get\_version

Use this method to find out the installed version of the word processor application.

```
CALL METHOD wp->get_version
  EXPORTING no_flush = no_flush
  IMPORTING version  = version
           error     = error
           retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
version		The version of the word processor application

## install\_template

You use this method to install a template for the office application on the presentation server:

**get\_version**

```
CALL METHOD wp->install_template
EXPORTING template_name = template_name
          url           = url
          user_info    = user_info
          no_flush     = no_flush
IMPORTING retcode     = retcode
          error       = error.
```

**Description of parameters**

Parameter	Optional	Description
template_name		The name of the file to be created on the presentation server for the template you are installing
url		The URL address of the template
user_info	<b>X</b>	Logon and proxy data for accessing an HTTP or FTP server.



The name of the template must not include path or extension data. The template is saved in the default template directory for the current user.



Remember always to use the format that applies to the respective office application (MS Word 97, Lotus WordPro, ...).

## get\_version

Use this method to find out the installed version of the word processor application.

```
CALL METHOD wp->get_version
EXPORTING no_flush = no_flush
IMPORTING version  = version
          error     = error
          retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
version		The version of the word processor application

## set\_template

You use this method to apply the template to the office document that is currently open.

## get\_template\_info

```
CALL METHOD wp->set_template
  EXPORTING template_name = template_name
            attach_styles = attach_styles
            no_flush      = no_flush
  IMPORTING retcode      = retcode
            error         = error.
```

### Description of parameters

Parameter	Optional	Description
template_name		The name of the template to be attached (in accordance with the name given during installation) If you do not specify a name, the standard template (NORMAL.DOT) is applied
attach_styles		'X': Formatting of the template is used in the document. ' ': The formatting of the template is ignored (that is, the existing formatting in the document is retained).



The name of the template must not include path or extension data.

## get\_template\_info

This method returns the name of the template attached to the word processor document currently open.

```
CALL METHOD wp->get_template_info
  EXPORTING no_flush      = no_flush
  IMPORTING retcode      = retcode
            error         = error
  CHANGING template_info = template_info
```

Parameter	Optional	Description
template_info		An internal table containing the name, path, and full name (path + name) of the template currently applied to the document. You should declare your actual parameter with reference to type <b>SOI_TEMPLATE_INFO</b> .

## remove\_template

You use this method to remove a template that has been installed locally on the presentation server:

```
CALL METHOD wp->remove_template
  EXPORTING template_name = template_name
            no_flush      = no_flush
```

```
IMPORTING retcode = retcode
error = error.
```

**Description of parameters**

Parameter	Optional	Description
template_name		The name of the template to be deleted (in accordance with the name given during installation)



The name of the template must not include path or extension data.

If you have used the template in the document, you must call the [set template \[Page 83\]](#) method with template name ' ' (space) before deleting.

## search

You use this method to activate the search function in the office document:

```
CALL METHOD wp->search
EXPORTING search_string = search_string
pos = pos
flag = flag
no_flush = no_flush
IMPORTING retcode = retcode
error = error.
```

**Description of parameters**

Parameter	Optional	Description
search_string		Searches for a character string
pos		global: searches the entire text up: searches from the current position to the start of the document down: searches from the current position to the end of the document
flag		' ': searches for character string word: searched for word exact: case-sensitive search

## replace

You use this method to replace a character string in the open office document

```
CALL METHOD wp->replace
EXPORTING replace_string = replace_string
search_string = search_string
pos = pos
flag = flag
no_flush = no_flush
```

**insert\_table**

```

IMPORTING retcode    = retcode
          error      = error.

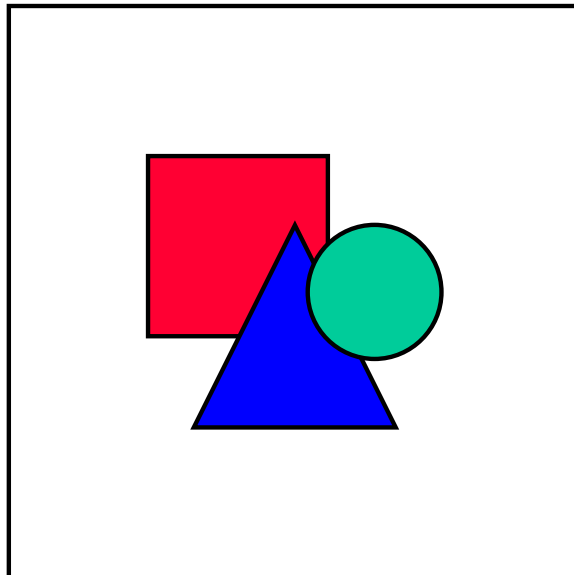
```

**Description of parameters**

Parameter	Optional	Description
search_string		Searches for a character string
replace_string		The character string that replaces the search string
pos		global: searches the entire text up: searches from the current position to the start of the document down: searches from the current position to the end of the document
flag		' ': searches for character string word: searched for word exact: case-sensitive search

**insert\_table**

Use this method to insert the contents of an internal table from your ABAP program into a table in the word processor document. The table in the document **must already exist** before you execute the method.



In this method, you use the index numbers of columns to indicate the order in which they should appear in the table in the document. If you want to use the column names instead, use the method [insert\\_table2 \[Page 88\]](#).

```

CALL METHOD wp->insert_table
  EXPORTING data_table    = data_table
            info_table    = info_table
            lowerbound    = lowerbound

```

insert\_table

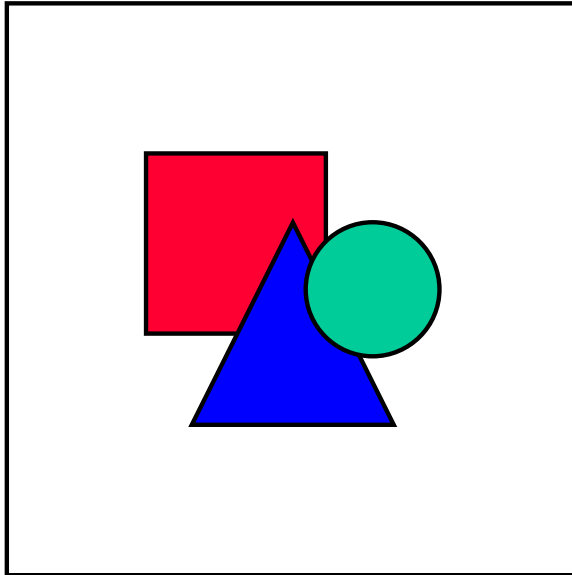
```

upperbound      = upperbound
doctable_number = doctable_number
clearoption     = clearoption
startrow       = startrow
varsize        = varsize
no_flush       = no_flush
IMPORTING retcode = retcode
error          = error.
    
```

Parameter	Optional	Description
data_table		The internal table containing the R/3 data you want to place in the table in the document
info_table		An internal table with the type <code>SOI_COLS_TABLE</code> . Each row in this table can contain the index number of a column in the data table. When it displays the data table in the document table, the system uses the columns you specified in the order in which you listed them. To leave a column blank in the table in the document, enter the value 0 in the relevant row of this table.
lowerbound		The index of the first entry of the data table you want to display in the document table. To start at the beginning of the table, use the value 0.
upperbound		The index of the last entry of the data table you want to display in the document table. To display all entries from the entry you specified in <b>lowerbound</b> up to the end of the table, use the value 0.
doctable_number		The number of the table in the document into which you want to insert the data. The numbering starts at 1.
clearoption		Option for clearing existing contents of the table in the document. It can have the following values: <ul style="list-style-type: none"> <li>• 0 - No clearing The method only overwrites fields of the table in which it actually enters data.</li> <li>• 1 - Clear in all rows The method overwrites the contents of the table with the new data from R/3. Data above the specified starting row (<code>startrow</code>) is left untouched. The contents of remaining lines of the old table that are below the insertion are deleted. <b>Note:</b> In rows of the document table in which the method inserts data, the contents of fields that are not filled automatically (value 0 in the corresponding row of <code>info_table</code>) are <b>not</b> deleted.</li> <li>• 2 - Clear only in accessed columns The method deletes the entire contents of the columns of the document table in which it inserts data. The contents of all other columns are left untouched.</li> </ul>
startrow		The first row of the document table in which you want to insert data

**insert\_table2**

varsize		<p>Flag to indicate whether the table in the document should be resized to fit the number of rows passed by the method. Possible values:</p> <ul style="list-style-type: none"> <li>' ': The table is not resized.</li> </ul> <p><b>Caution:</b> If the table has fewer rows than the number of entries you pass to it, not all of the data can be displayed.</p> <ul style="list-style-type: none"> <li>'x': The table is resized.</li> </ul>
wholetable	X	Not yet implemented



While you can use `varsize` to adjust the number of rows in the table automatically, you cannot adjust the number of columns. If you pass more columns from your data table than are available in the document table, the system will not be able to display the additional columns. To find out the number of columns in a particular table in the document, use the method [get table info \[Page 90\]](#).

**insert\_table2**

Use this method to insert the contents of an internal table from your ABAP program into a table in the word processor document. The table in the document **must already exist** before you execute the method.



The difference between this method and `insert_table` is that here you specify the columns of the data table that you want to display by **name**, not by their index number.

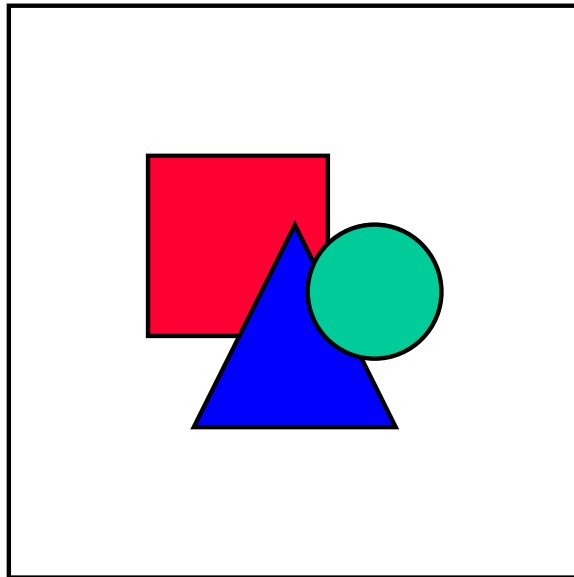
insert\_table2

```
CALL METHOD wp->insert_table2
  EXPORTING data_table      = data_table
            info_col_table = info_col_table
            lowerbound     = lowerbound
            upperbound     = upperbound
            doctable_number = doctable_number
            clearoption    = clearoption
            startrow       = startrow
            varsize        = varsize
            no_flush       = no_flush
  IMPORTING retcode        = retcode
            error          = error.
```

Parameter	Optional	Description
data_table		The internal table containing the R/3 data you want to place in the table in the document
info_col_table		An internal table with the type <code>SOI_COLNAMES_TABLE</code> . Each row in this table can contain the name of a column in the data table. When it displays the data table in the document table, the system uses the columns you specified in the order in which you listed them. To leave a column blank in the table in the document, enter the value ' ' (space) in the relevant row of this table.
lowerbound		The index of the first entry of the data table you want to display in the document table. To start at the beginning of the table, use the value 0.
upperbound		The index of the last entry of the data table you want to display in the document table. To display all entries from the entry you specified in <code>lowerbound</code> up to the end of the table, use the value 0.
doctable_number		The number of the table in the document into which you want to insert the data. The numbering starts at 1.
clearoption		Option for clearing existing contents of the table in the document. It can have the following values: <ul style="list-style-type: none"> <li>• 0 - No clearing The method only overwrites fields of the table in which it actually enters data.</li> <li>• 1 - Clear in all rows The method overwrites the contents of the table with the new data from R/3. Data above the specified starting row (<code>startrow</code>) is left untouched. The contents of the remaining lines of the old table that are below the insertion are deleted. <b>Note:</b> In rows of the document table in which the method inserts data, the contents of fields that are not filled automatically (value 0 in the corresponding row of <code>info_table</code>) are <b>not</b> deleted.</li> <li>• 2 - Clear only in accessed columns The method deletes the entire contents of the columns of the document table in which it inserts data. The contents of all other columns are left untouched.</li> </ul>

**get\_table\_info**

startrow		The first row of the document table in which you want to insert data
varsize		Flag to indicate whether the table in the document should be resized to fit the number of rows passed by the method. Possible values: <ul style="list-style-type: none"> <li>' ': The table is not resized.</li> </ul> <b>Caution:</b> If the table has fewer rows than the number of entries you pass to it, not all of the data can be displayed. <ul style="list-style-type: none"> <li>'x': The table is resized.</li> </ul>
wholetable	X	Not yet implemented



While you can use `varsize` to adjust the number of rows in the table automatically, you cannot adjust the number of columns. If you pass more columns from your data table than are available in the document table, the system will not be able to display the additional columns. To find out the number of columns in a particular table in the document, use the method [get\\_table\\_info \[Page 90\]](#).

**get\_table\_info**

Use this method to find out the number of rows and columns in each table in the word processor document.

```
CALL METHOD wp->get_table_info
    EXPORTING no_flush          = no_flush
    IMPORTING retcode           = retcode
```

```

error = error
CHANGING tableinfo = tableinfo.
    
```

Parameter	Optional	Description
tableinfo		An internal table with the type <code>SOI_TABLEINFO_TABLE</code> . This has two columns - <code>ROWS</code> and <code>COLUMNS</code> , which the method fills with the dimensions of each table in the document. The tables are listed in the internal table in the order in which they appear in the word processor document.

## clear\_table

Use this method to clear the contents of the table in the word processor document. Note that the structure of the table (in particular, the number of rows) remains unchanged.

```

CALL METHOD wp->clear_table
EXPORTING doctable_number = doctable_number
          no_flush         = no_flush
IMPORTING retcode         = retcode
          error            = error.
    
```

Parameter	Optional	Description
doctable_number		The number of the table in the word processor document whose contents you want to delete

## The Form Interface

### Definition

You define this instance with reference to the interface `i_oj_form`.

### Use

This interface is particularly appropriate for using form fields in documents. The data table is passed separately from the [link server \[Page 64\]](#) and [table collection \[Page 73\]](#) using the SAP Data Provider. The effect of this is that the data is transferred first, then placed into the document. User-defined fields are used in the office application.

This has the following advantages:

1. The value replacement is fully controlled by the R/3 System.
2. Values can be saved directly in the document. You can therefore edit the document even without a connection to the R/3 System.
3. This is a very flexible solution, since no office-specific functions are used. Instead, you work directly with the fields that you want to change.

## The Form Interface

However, the inter-process communication does mean that the replacement is slower than if you were to use the built-in mail merge functions in the desktop office application. However, since the main reason for introducing the function was that the data must be modifiable within the document, solutions like [mail merge \[Page 98\]](#) are not suitable.

## Structure

Data is exchanged between the R/3 System and the office application using a table with the data type `soi_form_list`. This table covers the following functions:

- Setting the fields in the document with up to date entries.
- Getting the fields from the fields in the document.
- Setting new values for certain entries (delta management).

The table contains the following fields:

Field name	Description	Possible values
type	Type of field in document	'C': Checkbox 'T': Text field 'L': List box 'P': Property: Field should only be displayed.
name	Name of field in document Data is always passed between pairs of identically-named fields.	
number	This field is only relevant for list boxes. If a list box contains more than one identical entry, this field distinguishes them.	
value	This field contains the current value of the field. For checkboxes, the Boolean representation is simulated using 0 and 1.	
code	You can use this field to control entries in the document directly. This only plays a part when data is transferred from the R/3 System to the office application. In the opposite direction, all entries are set to '0'.	0: No effect 1: Only for list boxes: The entry is selected. 2: Only for list boxes: All entries in the list box are deleted. 3: All types: The field contents cannot be changed.

## Integration

Before creating the *wp* instance, you must create the *document* instance for managing the document.

The interface currently supports Microsoft Word and Excel, and Lotus Word Pro.

## Working With The Form Interface

### Purpose

The form interface allows you to load data from the R/3 System into a form in a desktop office application and return them from the application to the R/3 System after the form has been processed. The desktop office application must support this interface (see the method [has\\_form\\_interface \[Page 57\]](#)).

### Prerequisites

You must create the instance **document** for document management before you can use the interface.

### Process flow

1. Declare an instance for the central object of the Office Integration with reference to the interface *i\_oi\_form*:

```
DATA form TYPE REF TO i_oi_form.
```

2. Use the method [get\\_form\\_interface \[Page 58\]](#) on the document instance for document management to create the instance for the form interface.
3. Use the methods of the form interface:

#### Methods of the form interface

Method	Function
<i>set_data</i>	Passes the data from the R/3 System to the desktop application.
<i>get_data</i>	Transfers the contents of the form in the desktop application to the R/3 System.
<i>add_field</i>	Adds a form field.
<i>delete_fields</i>	Deletes a form field in the desktop application.
<i>disconnect_fields</i>	Terminates the connection between the form field and the R/3 System.
<i>get_form_fields</i>	Reads the form fields from the desktop application.
<i>protect</i>	Controls whether the document can accept input.
<i>set_modus</i>	Changes the display mode.



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## set\_data

Use this method to fill the form in the office application with the up-to-date data from the R/3 System. The fields in the document are altered according to the contents of the table. You can also use this method to change the entries in a list box. To do this, you must create a table containing the new list box entries. These entries are automatically entered in the list box, as long as you do not use the command '2' in the *Code* field, since this deletes all entries in the list box.

```
CALL METHOD form->set_data
  EXPORTING fields = fields
          clear   = clear
          no_flush = no_flush
  IMPORTING retcode = retcode
          error   = error.
```

### Description of parameters

Parameter	Optional	Description
fields		Table with the structure <code>soi_form_list</code> . These entries are used to change the fields in the form.

**get\_data**

clear	X	'X': Before the fields from the <code>fields</code> table are imported, all fields in the form are initialized. The exception to this are list boxes, which are not deleted. ' ': The form fields are updated with the values from the <code>fields</code> table. If the table does not contain a new value for a field, its existing value in the form is not changed.
-------	---	--

Examples for the `fields` table:

**Example 1:**

Type	Name	Number	Value	Code
C	Single	1	1	0
T	Name	1	A. N. Other	0
T	Department	1	ABAP	3

This table would place a value in a checkbox. The second record would place the entry *A. N. Other* into a text field. The final entry would change the *Department* field. The command '3' means that the entry can no longer be changed in the document.

**Example 2:**

Type	Name	Number	Value	Code
L	Colors	1	Red	2
L	Colors	2	Green	1
L	Colors	3	Blue	0

This table fills a list box with entries. The command '2' means that the list box can only contain entries defined in the table. The command '1' selects the second entry.

## get\_data

This method returns a table of all current entries in the form.

```
CALL METHOD form->get_data
  EXPORTING all = all
            no_flush = no_flush
  IMPORTING fields = fields
            retcode = retcode
            error = error.
```

**Description of parameters**

Parameter	Optional	Description
all	X	'X': Transfers all values from the form, including non-selected entries from list boxes.

**add\_field**

fields		Table with the structure <code>soi_form_list</code> . The contents of this table reflect those of the form.
--------	--	---



The `value` column of the `fields` table contains the corresponding field value. The possible values for list boxes are returned in this column. The selected value is indicated by the entry '1' in the `code` column.

**add\_field**

Use this method to add a new form field.

```
CALL METHOD form->add_field
  EXPORTING field = field
           no_flush = no_flush
  IMPORTING retcode = retcode
           error = error.
```

**Description of parameters**

Parameter	Optional	Description
field		Structure with the type <code>soi_form_item</code> . The field is displayed in the form according to the entries in the structure.

**delete\_fields**

Use this method to pass a table containing the fields that you want to delete from the document. Both the link and the displayed text are deleted.

```
CALL METHOD form->delete_fields
  EXPORTING fields = fields
           no_flush = no_flush
  IMPORTING retcode = retcode
           error = error.
```

**Description of parameters**

Parameter	Optional	Description
fields		Table with the structure <code>soi_form_list</code> . The specified fields are deleted from the form.

## disconnect\_fields

Use this method to disconnect the links between the fields named in the table and the R/3 System. The field contents are not deleted from the document; they appear as simple text fields.

```
CALL METHOD form->delete_fields
  EXPORTING fields = fields
           no_flush = no_flush
  IMPORTING retcode = retcode
           error = error.
```

### Description of parameters

Parameter	Optional	Description
fields		Table with the structure <code>soi_form_list</code> . The links between the specified fields in the document and the R/3 System are disconnected.

## get\_form\_fields

This method returns a table containing all of the fields in the form and their formats. Unlike the [get\\_data \[Page 95\]](#) method, no field contents are transferred.

```
CALL METHOD form->get_form_fields
  EXPORTING no_flush = no_flush
  IMPORTING fields = fields
           retcode = retcode
           error = error.
```

### Description of parameters

Parameter	Optional	Description
fields		Table with the structure <code>soi_form_list</code> . It contains all fields from the form.

## protect

Use this method to protect the document. It is then only possible to enter data in the fields of the form. It makes sense to use this option, since it makes it easier for users to navigate around the form.

```
CALL METHOD form->protect
  EXPORTING protect = protect
           no_flush = no_flush
  IMPORTING retcode = retcode
           error = error.
```

**set\_modus****Description of parameters**

Parameter	Optional	Description
protect	X	'X': The form (but not its fields) is protected against changes. ' ': Any part of the form can be changed.

**set\_modus**

Use this method to specify whether the document should be displayed in normal mode or design mode.

```
CALL METHOD form->set_modus
  EXPORTING modus   = modus
           no_flush = no_flush
  IMPORTING retcode = retcode
           error    = error.
```

**Description of parameters**

Parameter	Optional	Description
modus	X	'X': Display in display mode. ' ': Display in design mode.

**The Mail Merge Interface****Definition**

You define this instance with reference to the interface *i\_oi\_mail\_merge*.

**Use**

Use this interface to control the mail merge functions of the desktop application. The interface allows you to transfer data from the R/3 System to the frontend and use it in a mail merge operation.

**Integration**

The interface currently supports Microsoft Word and Lotus Word Pro.

## Using the Mail Merge Interface

### Purpose

The mail merge interface allows you to control the mail merge function of a desktop application from within the R/3 System. The application must support the interface (see also the documentation for [has\\_mail\\_merge\\_interface \[Page 58\]](#)).

### Prerequisites

You must create the instance **document** for document management before you can use the interface.

### Process flow

1. Declare an instance with reference to the interface *i\_oi\_mail\_merge*:
 

```
DATA mail_merge TYPE REF TO i_oi_mail_merge.
```
2. Use the method [get\\_mail\\_merge\\_interface \[Page 58\]](#) on the document instance for document management to create the instance for the mail merge interface.
3. Use the mail merge interface methods:

#### Methods of the Mail Merge Interface

Method	Function
<i>set_data_source</i>	Transfers data to the frontend
<i>get_fields</i>	Mail merge fields
<i>merge_one</i>	Merges a single data record.
<i>merge_range</i>	Merges a set of data records.
<i>print</i>	Prints a mail merge document.
<i>view</i>	Displays the document with merged data.
<i>shutdown</i>	Destroys the interface.
<i>view_field_codes</i>	Displays the field functions in the mail merge document.



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

**set\_data\_source****Description of parameters**

Parameter	Optional	Description
no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

**set\_data\_source**

Use this method to pass data records that you can later merge into the mail merge document using [merge\\_one \[Page 101\]](#) or [merge\\_range \[Page 101\]](#).

. You need a table containing the requisite data. A line of the table contains all of the data required for the fields in a single letter.

```
CALL METHOD mail_merge->set_data_source
  EXPORTING ddic_name = ddic_name
            description = description
            date       = date
            time       = time
            no_flush   = no_flush
  IMPORTING error     = error
            retcode    = retcode
  CHANGING data_table = data_table
            fields_table = fields_table
            properties = properties.
```

**Description of parameters**

Parameter	Optional	Description
ddic_name	X	Name of the table or structure in the ABAP Dictionary.
description	X	Description of the table
date	X	Date of last change
time	X	Time of last change
data_table		Data table

**get\_fields**

fields_table	X	Description of the table structure (if you have not already passed it using the <code>ddic_name</code> parameter). Create the table with reference to the type <code>SOI_FIELDS_TABLE</code> .
properties	X	Additional properties Create the table with reference to the type <code>SOI_PROPERTIES_TABLE</code> .

## get\_fields

This method returns a list of all of the fields that you can address using the mail merge interface.

```
CALL METHOD mail_merge->get_fields
  EXPORTING no_flush = no_flush
  IMPORTING descr_list = descr_list
           retcode = retcode
           error = error.
```

### Description of parameters

Parameter	Optional	Description
descr_list		Table, containing the fields from the mail merge document. Create the table with reference to the type <code>SOI_FIELDS_NAME_TABLE</code> .

## merge\_one

Use this method to import a data record into the mail merge template. The `index` parameter identifies the data record, which you must already have sent to the frontend using the [set\\_data\\_source \[Page 100\]](#) method.

```
CALL METHOD mail_merge->merge_one
  EXPORTING index = index
           no_flush = no_flush
  IMPORTING retcode = retcode
           error = error.
```

### Description of parameters

Parameter	Optional	Description
index		Index of the data record.

**merge\_range****merge\_range**

Use this method to merge a set of data records one after the other into the mail merge template. Use the first and last parameters to delimit the range of table records that you want to use. You must already have passed the table using the method [set\\_data\\_source \[Page 100\]](#).

```
CALL METHOD mail_merge->merge_range
  EXPORTING first    = first
           last     = last
           no_flush = no_flush
  IMPORTING retcode  = retcode
           error    = error.
```

**Description of parameters**

Parameter	Optional	Description
first		The first data record you want to merge
last		The last data record you want to merge

**print**

This method triggers the print function in the office application. Before calling this method, you should use the methods [merge\\_one \[Page 101\]](#) or [merge\\_range \[Page 101\]](#) to incorporate the data in the mail merge template.

```
CALL METHOD mail_merge->print
  EXPORTING no_flush = no_flush
  IMPORTING retcode  = retcode
           error    = error.
```

**view**

Use this method to display the mail merge document with the merged data in the desktop application.

```
CALL METHOD mail_merge->view
  EXPORTING no_flush = no_flush
  IMPORTING error    = error
           retcode   = retcode.
```

**shutdown**

Use this method to close down the mail merge function.

```
CALL METHOD mail_merge->shutdown
  EXPORTING no_flush = no_flush
```

```
IMPORTING error      = error
         retcode    = retcode.
```

## view\_field\_codes

Use this method to specify whether the field functions are displayed.

```
CALL METHOD mail_merge->view_field_codes
  EXPORTING viewcodes = viewcodes
           no_flush   = no_flush
  IMPORTING error      = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
viewcodes		'X': Do not display field functions ' ': Display field functions

## The Script Collection

### Definition

You define this instance with reference to the interface *i\_oj\_script\_collection*.

### Use

This interface allows you to load ASCII files containing scripts into your desktop application. The scripts must be able to be addressed using their name. The scripts can be written in VB Script and Java Script.

The aim of the script collection is to reduce the number of macros required in documents. You can store the macros centrally in a single script, which also makes them easier to maintain.

### Integration

This interface can be used for all applications. The only prerequisite is that the application can access a script engine (script engine of Internet Explorer). This means that the interface does not rely on the relevant macro language being exposed.

## Using the Script Collection

## Using the Script Collection

### Purpose

The script collection allows you to include macros belonging to a certain area in a script file. The script collection downloads the script file to the document, and the macros can be run.

### Prerequisites

You must create the instance **document** for document management before you can use the interface.

### Process flow

1. Declare an instance with reference to interface *i\_oi\_script\_collection*:
 

```
DATA script_coll TYPE REF TO i_oi_script_collection.
```
2. Use the method [get\\_script\\_collection \[Page 60\]](#) on the document instance for document management to create the instance for the mail merge interface.
3. Use the script collection methods:

#### Methods in the Script Collection

Method	Function
<i>add_script</i>	Adds a script to the collection
<i>add_script_from_table</i>	Adds a script to the collection
<i>remove_all_scripts</i>	Remove all scripts
<i>remove_script</i>	Remove a particular script



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

#### Description of parameters

Parameter	Optional	Description
no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.

**add\_script**

retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ''` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## add\_script

Use this method to add a script to the collection. The script is identified by a URL.

```
CALL METHOD script_coll->add_script
  EXPORTING script_name = script_name
            script_type = script_type
            script_url  = script_url
            user_info   = user_info
            no_flush    = no_flush
  IMPORTING retcode    = retcode
            error      = error.
```

### Description of parameters

Parameter	Optional	Description
script_name		Technical name of the script
script_url		The URL address of the script
script_type	<b>X</b>	<b>script_type_unknown:</b> The system tries to determine the type from the MIME type within the URL. If this is not possible, an error occurs. <b>script_type_vbscript:</b> The script is written in VB Script. <b>script_type_jscript:</b> The script is written in Java Script.
user_info	<b>X</b>	Logon and proxy data for accessing an HTTP or FTP server.

## add\_script\_from\_table

Use this method to load a script. The script must be contained in a table in the application program.

```
CALL METHOD script_coll->add_script_from_table
  EXPORTING script_name = script_name
```

**remove\_all\_scripts**

```

    script_type = script_type
    script_table = script_table
    no_flush    = no_flush
    IMPORTING retcode    = retcode
    error      = error.

```

**Description of parameters**

Parameter	Optional	Description
script_name		Technical name of the script
script_type	X	<b>script_type_unknown</b> : The system tries to determine the type from the MIME type within the URL. If this is not possible, an error occurs. <b>script_type_vbscript</b> : The script is written in VB Script. <b>script_type_jscript</b> : The script is written in Java Script.
script_table		Internal table into which the script is to be loaded.

**remove\_all\_scripts**

This method removes all loaded scripts.

```

CALL METHOD script_coll->remove_all_scripts
    EXPORTING no_flush    = no_flush
    IMPORTING error      = error
    retcode    = retcode.

```

**remove\_script**

Use this method to remove a particular script.

```

CALL METHOD script_coll->remove_script
    EXPORTING script_name = script_name
    no_flush    = no_flush
    IMPORTING error      = error
    retcode    = retcode.

```

**Description of parameters**

Parameter	Optional	Description
script_name		Name of the script you want to remove.

## The Spreadsheet Interface

### Definition

You define this instance with reference to the interface *i\_oi\_spreadsheet*.

### Use

You use this interface to communicate with a spreadsheet.

All spreadsheets allow you to define and assign names to ranges. This interface works with named ranges within a spreadsheet.

The individual ABAP methods contain only functions that are available in the spreadsheet packages supported by the interface. That is, each method corresponds directly to a function within the spreadsheet.

### Structure

To exchange data with the spreadsheet, you need tables with special structures. These structure definitions are contained in the [Important Table Structures \[Page 136\]](#) section.

### Integration

The interface currently supports Microsoft Excel and Lotus 1-2-3.

## Using the Spreadsheet Interface

### Purpose

The spreadsheet interface allows you to load data from the R/3 System into a spreadsheet and return it from the application to the R/3 System. The desktop office application must support this interface (see the method [has\\_spreadsheet\\_interface \[Page 57\]](#)).

### Prerequisites

You must create the instance **document** for document management before you can use the interface.

### Process flow

1. Declare an instance with reference to the interface *i\_oi\_spreadsheet*.  

```
DATA spreadsheet TYPE REF TO i_oi_spreadsheet.
```
2. Use the method [get\\_spreadsheet\\_interface \[Page 57\]](#) on the document instance for document management to create the instance for the spreadsheet interface.
3. Use the spreadsheet interface methods:

#### Methods of the Spreadsheet Interface

## Using the Spreadsheet Interface

Method	Function
<i>get_active_sheet</i>	Active worksheet
<i>get_dimension</i>	Size of a range
<i>get_protection</i>	Protection of a range
<i>get_sheets</i>	Existing worksheets
<i>get_ranges_data</i>	Data in particular ranges
<i>get_ranges_name</i>	Names of existing ranges
<i>get_cell_format</i>	Format of a particular cell
<i>get_cell_formats_table</i>	Format of a set of cells
<i>get_selection</i>	Selection within a worksheet
<i>get_selected_areas</i>	Multiple selections within a worksheet
<i>select_range</i>	Selects a range
<i>set_selection</i>	Selects any area within a worksheet
<i>select_sheet</i>	Activates a worksheet
<i>set_color</i>	Changes the colors
<i>set_font</i>	Changes the font
<i>set_format</i>	Changes the format
<i>set_format_string</i>	Changes the format (only in Microsoft Excel)
<i>set_frame</i>	Changes the line format
<i>set_ranges_format</i>	Changes the format of a set of ranges
<i>cell_format</i>	Changes the format of a cell
<i>protect</i>	Protects worksheet
<i>protect_range</i>	Protects a range
<i>add_sheet</i>	Adds a worksheet to the current workbook
<i>set_sheet_name</i>	Sets the name of a worksheet
<i>delete_sheet</i>	Deletes a worksheet
<i>insert_range_dim</i>	Adds a range
<i>change_range</i>	Changes a range
<i>insert_range</i>	Inserts a range at the selected point
<i>insert_one_table</i>	Inserts a data table into a range
<i>set_ranges_data</i>	Inserts data into a range
<i>clear_range</i>	Deletes a range
<i>clear_all_ranges</i>	Deletes the contents and formatting of all ranges

Generic Parameters

<i>delete_content_ranges</i>	Deletes the contents of all ranges
<i>delete_ranges</i>	Deletes all of the range definitions
<i>hide_columns</i>	Hides columns
<i>show_columns</i>	Shows columns
<i>hide_rows</i>	Hides rows
<i>show_rows</i>	Shows rows
<i>set_hierarchy</i>	Sets a hierarchy in the worksheet
<i>set_hierarchy_table</i>	Sets multiple hierarchies in the worksheet
<i>clear_hierarchy</i>	Clears the hierarchies in the worksheet
<i>print</i>	Prints a worksheet
<i>set_zoom</i>	Change the zoom
<i>fit_widest</i>	Adjust the column width
<i>screen_update</i>	Switches the screen update on or off
<i>load_lib</i>	Loads an additional DLL that can improve performance
<i>version</i>	Returns the version of the spreadsheet interface



Remember that you should include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

## Generic Parameters

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

### Description of parameters

Parameter	Optional	Description
no_flush	X	'x': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>

## The Updating Parameter



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## The Updating Parameter

The updating parameter is used in several methods of the spreadsheet interface to control whether the screen of the spreadsheet application should be redrawn immediately or not (redrawing the screen in a single step for a bundle of commands is more efficient than having to redraw it several times). It is always optional, and can have the following values:

Value	Meaning
-1 (default value)	The existing update setting remains unchanged
0	The screen update is switched off before the method is called, and remains switched off
1	The screen update is switched on after the method has been executed, and remains switched on

Methods that do not have an updating parameter do not affect the current state of the update mechanism.

## get\_active\_sheet

This method returns the name of the current worksheet.

```
CALL METHOD spreadsheet->get_active_sheet
  EXPORTING no_flush = no_flush
  IMPORTING sheetname = sheetname
  error      = error
  retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
sheetname		Name of current worksheet

## get\_dimension

This method returns the size of the specified range, that is, the top left-hand corner and the number of lines and columns.

```
CALL METHOD spreadsheet->get_dimension
  EXPORTING name      = name
            no_flush  = no_flush
  IMPORTING top       = top
            left      = left
            rows      = rows
            columns   = columns
            error     = error
            retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		Name of the range
top		Top edge of the range
left		Left-hand edge of the range
rows		Number of rows
columns		Number of columns

## get\_protection

The return value is set if the specified worksheet is protected.

```
CALL METHOD spreadsheet->get_protection
  EXPORTING sheetname = sheetname
            no_flush  = no_flush
  IMPORTING protect   = protect
            error     = error
            retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
sheetname		Name of worksheet
protect		'X': Worksheet is protected ' ': Worksheet is not protected

**get\_sheets****get\_sheets**

This method returns a list of names of existing worksheets.

```
CALL METHOD spreadsheet->get_sheets
  EXPORTING no_flush = no_flush
           updating  = updating
  IMPORTING sheets   = sheets
           error     = error
           retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
sheets		List of worksheets. The parameter must be created with reference to the type <code>SOI_SHEETS_TABLE</code> .
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

**get\_ranges\_data**

This method returns the data of given ranges within a worksheet. There are two ways of controlling this, depending on how you set the `all` parameter.

```
CALL METHOD spreadsheet->get_ranges_data
  EXPORTING all      = all
           rangesdef = rangesdef
           updating  = updating
           no_flush  = no_flush
  IMPORTING contents = contents
           error     = error
           retcode   = retcode
  CHANGING ranges    = ranges.
```

**Description of parameters**

Parameter	Optional	Description
all	X	'X': Returns the data in all ranges ' ': Returns only the data from the ranges specified in the <code>ranges</code> table.

**get\_ranges\_names**

rangesdef	X	An internal table with the type <b>SOI_DIMENSION_TABLE</b> with which you can define ranges by their starting position and size instead of their name.  <b>Note:</b> If you use this parameter, the <b>ranges</b> table (see below) must be <b>empty</b> .
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>
contents		Data table. Each row contains the contents of one cell.  The table must be created with reference to the type <b>SOI_GENERIC_TABLE</b> (see the <a href="#">Important Table Structures [Page 136]</a> section).
ranges		Result table for the query. If you use the <b>all</b> or <b>rangesdef</b> parameters, it must be empty. If you do not, you must fill it with the names of the ranges before you call the method.  The table must be created with reference to the type <b>SOI_RANGE_LIST</b> (see the <a href="#">Important Table Structures [Page 136]</a> section).

## get\_ranges\_names

This method returns a table of all ranges with their sizes. The control code has no significance - it is not filled.

```
CALL METHOD spreadsheet->get_ranges_names
  EXPORTING updating = updating
            no_flush = no_flush
  IMPORTING ranges   = ranges
            error    = error
            retcode  = retcode.
```

### Description of parameters

Parameter	Optional	Description
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>
ranges		Table with all ranges  The table must be created with reference to the type <b>SOI_RANGE_LIST</b> (see the <a href="#">Important Table Structures [Page 136]</a> section).

**get\_cell\_format****get\_cell\_format**

Use this method to retrieve the formatting information for a particular cell. Only the color, alignment, and font fields are filled. The currency fields, and formatting such as scientific display, are not filled. This is because the spreadsheet packages supported by the interface offer too many formatting possibilities.

```
CALL METHOD spreadsheet->get_cell_format
  EXPORTING row      = row
           column   = column
           no_flush = no_flush
  IMPORTING format   = format
           error     = error
           retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
row		Line number
column		Column number
format		Format of the cell. Structure with type <code>SOI_FORMAT_ITEM</code> (see also <a href="#">Important Table Structures [Page 136]</a> ).

**get\_cell\_formats\_table**

Use this method to retrieve the formatting information for a group of cells. Only the color, alignment, and font fields are filled. The currency fields, and formatting such as scientific display, are not filled. This is because the spreadsheet packages supported by the interface offer too many formatting possibilities.

```
CALL METHOD spreadsheet->get_cell_formats_table
  EXPORTING no_flush = no_flush
  IMPORTING error     = error
           retcode   = retcode
  CHANGING table     = table.
```

**Description of parameters**

Parameter	Optional	Description
table		Internal table with the data type <code>SOI_CELL_TABLE</code> (see <a href="#">Important Table Structures [Page 136]</a> ). You fill the table with the coordinates of the relevant cells before the method call. The method fills the remaining fields with formatting information.

## get\_selection

This method returns the top left-hand corner and the size of the current selection in the worksheet.

```
CALL METHOD spreadsheet->get_selection
  EXPORTING updating = updating
           no_flush = no_flush
  IMPORTING top      = top
           left     = left
           rows     = rows
           columns  = columns
           error    = error
           retcode  = retcode.
```

### Description of parameters

Parameter	Optional	Description
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>
top		Top edge of the range
left		Left-hand edge of the range
rows		Number of rows
columns		Number of columns

## get\_selected\_areas

Use this method to find out the areas on the current worksheet that are selected. The parameter areas returns an internal table, each row of which describes one selected area of the worksheet.

```
CALL METHOD spreadsheet->get_selected_areas
  EXPORTING no_flush = no_flush
  IMPORTING areas    = areas
           error     = error
           retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
areas		An internal table with the type <code>soi_area_table</code> . For further information, refer to the <a href="#">Important Table Structures [Page 136]</a> section.

**select\_range****select\_range**

Selects an existing named range.

```
CALL METHOD spreadsheet->select_range
  EXPORTING rangename = rangename
         no_flush = no_flush
  IMPORTING error      = error
         retcode  = retcode.
```

**Description of parameters**

Parameter	Optional	Description
rangename		Name of the range you want to select

**set\_selection**

Selects an unnamed range within the current worksheet. You specify the top left-hand corner and the size of the range you want to select.

```
CALL METHOD spreadsheet->set_selection
  EXPORTING top      = top
         left       = left
         rows       = rows
         columns    = columns
         updating   = updating
         no_flush   = no_flush
  IMPORTING error    = error
         retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
top		Top edge of the range
left		Left-hand edge of the range
rows		Number of rows
columns		Number of columns
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

## select\_sheet

The specified worksheet is activated, that is, placed in the foreground. All subsequent [get\\_selection \[Page 114\]](#) and [set\\_selection \[Page 115\]](#) method calls then apply to this worksheet.

```
CALL METHOD spreadsheet->select_sheet
  EXPORTING name      = name
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		Name of worksheet

## set\_color

Sets the font and background colors for a particular area. The values come from the color palette in Excel, not RGB values, even if the spreadsheet in question only accepts RGB values.

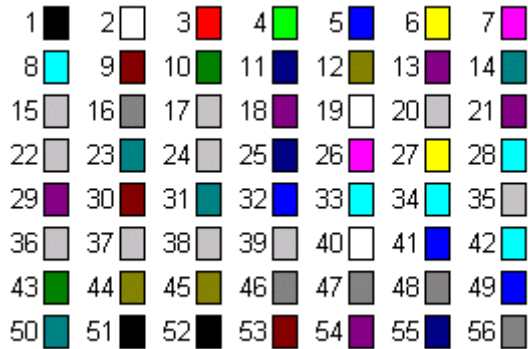
```
CALL METHOD spreadsheet->set_color
  EXPORTING rangename = rangename
           front      = front
           back       = back
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
rangename		Area name
front		Font color (see color palette) '-1': Color remains unchanged
back		Background color (see color palette) '-1': Color remains unchanged

The color palette contains the following colors:

**set\_font**



**set\_font**

This method sets the font for the specified range. The font used is one of the family to which the specified font belongs.

```
CALL METHOD spreadsheet->set_font
EXPORTING rangename = rangename
         family   = family
         size     = size
         bold     = bold
         italic   = italic
         align    = align
         no_flush = no_flush
IMPORTING error    = error
         retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
rangename		Area name
family		Font. The following fonts are currently available: 'Arial' 'Courier New' 'Times New Roman'
size		Font size '-1': Unchanged
bold		'1': Bold '0': Normal '-1': Unchanged

italic		'1': Italic '0': Normal '-1': Unchanged
align		Alignment: <ul style="list-style-type: none"> <li>• '-1': Unchanged</li> <li>• '0': Right-justified</li> <li>• '1': Centered</li> <li>• '2': Left-justified</li> </ul>

## set\_format

You can set the formatting of the cells in a range by specifying a currency using its ISO standard code. This code is also used within the R/3 System, which means that you can normally simply pass the currency code from R/3. You can also specify the currency formatting using a type. The digits in the cell are then displayed according to the specified type.

```
CALL METHOD spreadsheet->set_format
EXPORTING rangename = rangename
        typ      = typ
        currency = currency
        decimals = decimals
        no_flush = no_flush
IMPORTING error   = error
        retcode  = retcode.
```

### Description of parameters

Parameter	Optional	Description
rangename		Area name
typ		Type for currency formatting: <ul style="list-style-type: none"> <li>• 0: Display as text</li> <li>• 1: Display as a simple number</li> <li>• 2: Scientific display</li> <li>• 3: Display as a percentage</li> <li>• -1: The system determines the display type by analyzing the <b>currency</b> parameter.</li> </ul>
currency		ISO standard currency code
decimals	X	The number of decimal places

**set\_format\_string**

Desktop Office Integration uses the lowest common denominator of the various formatting options in the spreadsheet packages that it supports. Consequently, there are fewer formatting options available than in the spreadsheets themselves.

**set\_format\_string**

If you are using Microsoft Excel, you can format a cell with more options than allowed by the [set\\_format](#) [Page 118] method. This method sends a format string to Excel.



You should avoid using this method, since it cannot be converted by Lotus 1-2-3 or other spreadsheets.

```
CALL METHOD spreadsheet->set_format_string
  EXPORTING rangename = rangename
           formatstring = formatstring
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

**Description of parameters**

Parameter	Optional	Description
rangename		Area name
formatstring		Format string to be passed to Microsoft Excel

**set\_frame**

You can change the appearance of the lines in a range using a control byte and a color index.

```
CALL METHOD spreadsheet->set_frame
  EXPORTING rangename = rangename
           typ = typ
           color = color
           no_flush = no_flush
  IMPORTING error = error
           retcode = retcode.
```

**Description of parameters**

Parameter	Optional	Description
rangename		Area name

**set\_ranges\_format**

typ		Control byte for setting the frame
color		Frame color (see color palette)

The control byte **type** contains the following bits. If a bit is set, its corresponding line is drawn. You can set the thickness of the line to one of four levels using bits 6 and 7.

Bit	Description
0	Sets the left margin
1	Sets the top margin
2	Sets the bottom margin
3	Sets the right margin
4	Horizontal line
5	Sets the left margin
6	Thickness
7	Thickness

## set\_ranges\_format

Use this method to change the format of a set of ranges.

```
CALL METHOD spreadsheet->set_ranges_format
EXPORTING formattable = formattable
          updating     = updating
          no_flush     = no_flush
IMPORTING error       = error
          retcode      = retcode.
```

### Description of parameters

Parameter	Optional	Description
formattable		Control table with type <code>SOI_FORMAT_TABLE</code> (see also <a href="#">Important Table Structures [Page 136]</a> ).
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

## cell\_format

You use this method to set the formatting of a cell or group of cells. It is similar to the [set\\_ranges\\_format \[Page 120\]](#) method, but does not require you to specify a named range in the worksheet. Instead, you specify the cell coordinates of the top left-hand cell, and the number of

**protect**

rows and columns over which the range should extend. If necessary, you can retrieve this information using the method [get\\_selected\\_areas \[Page 114\]](#).

```
CALL METHOD spreadsheet->cell_format
    EXPORTING cells      = cells
             no_flush   = no_flush
    IMPORTING error      = error
             retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
cells		An internal table with the type SOI_CELL_TABLE. For further information, refer to the <a href="#">Important Table Structures [Page 136]</a> section.

**protect**

Use this method to protect the current worksheet so that only certain ranges can still be edited.

```
CALL METHOD spreadsheet->protect
    EXPORTING protect    = protect
             no_flush   = no_flush
    IMPORTING error      = error
             retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
protect		'X': Worksheet is protected ' ': Worksheet is not protected

**protect\_range**

Use this method to change the 'Protected' attribute of a range.

```
CALL METHOD spreadsheet->protect_range
    EXPORTING name       = name
             protect     = protect
             no_flush    = no_flush
    IMPORTING error      = error
             retcode     = retcode.
```

**Description of parameters**

Parameter	Optional	Description
-----------	----------	-------------

name		Name of the range you want to protect
protect		'X': Range is protected ' ': Range is not protected

## add\_sheet

Use this method to create a new worksheet in the current spreadsheet document. The worksheet is always inserted directly before the current sheet.

```
CALL METHOD spreadsheet->add_sheet
  EXPORTING names      = name
           no_flush   = no_flush
  IMPORTING error      = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		A type C field containing the name of the new worksheet.

## set\_sheet\_name

Use this method to rename an existing worksheet in the current spreadsheet document.

```
CALL METHOD spreadsheet->set_sheet_name
  EXPORTING oldname    = oldname
           newname     = newname
           no_flush    = no_flush
  IMPORTING error      = error
           retcode     = retcode.
```

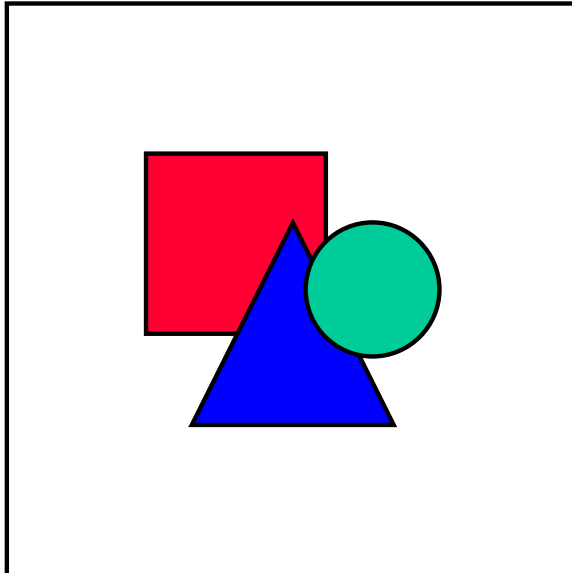
### Description of parameters

Parameter	Optional	Description
oldname		A type C field containing the current name of the worksheet you want to rename
newname		A type C field containing the new name of the worksheet

delete\_sheet

## delete\_sheet

Use this method to delete a worksheet from the current spreadsheet document.



When you use this method, the spreadsheet package reminds the user that the action could cause data loss. You do not have to write your own prompt.

```
CALL METHOD spreadsheet->delete_sheet
  EXPORTING name      = name
           no_flush  = no_flush
  IMPORTING error     = error
           retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		A type C field containing the name of the worksheet you want to delete

## insert\_range\_dim

Inserts a range in the current worksheet with the specified dimensions.

```
CALL METHOD spreadsheet->insert_range_dim
  EXPORTING name      = name
           top       = top
           left      = left
           rows      = rows
           columns   = columns
           updating  = updating
```

```

no_flush = no_flush
IMPORTING error = error
retcode = retcode.
    
```

**Description of parameters**

Parameter	Optional	Description
name		The name of the new range
top		Top edge of the range
left		Left-hand edge of the range
rows		Number of rows
columns		Number of columns
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

## change\_range

Changes the name and size of an existing range.

```

CALL METHOD spreadsheet->change_range
EXPORTING rangename = rangename
newname = newname
rows = rows
columns = columns
updating = updating
no_flush = no_flush
IMPORTING error = error
retcode = retcode.
    
```

**Description of parameters**

Parameter	Optional	Description
rangename		Range name
newname	X	New name of the range
rows	X	Number of rows
columns	X	Number of columns
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

**insert\_range****insert\_range**

Inserts a new range in the current worksheet based on the selection. The top left-hand corner of the current selection becomes the top left-hand corner of the new range. The size of the current selection is irrelevant, since the size of the new range is defined by the corresponding parameters in the CALL METHOD statement.

```
CALL METHOD spreadsheet->insert_range
  EXPORTING name      = name
           rows       = rows
           columns    = columns
           updating   = updating
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
name		The name of the new range
rows		Number of rows
columns		Number of columns
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

**insert\_ranges**

Use this method to insert a set of ranges into the spreadsheet document. Each range is described by one entry in the internal table that you pass in the **ranges** parameter.

```
CALL METHOD spreadsheet->insert_ranges
  EXPORTING ranges      = ranges
           no_flush    = no_flush
  IMPORTING error      = error
           retcode     = retcode.
```

**Description of parameters**

Parameter	Optional	Description
ranges		An internal table with the type <code>SOI_FULL_RANGE_TABLE</code> . Each row contains the description of one range. For further information, refer to the <a href="#">Important Table Structures [Page 136]</a> section.

## insert\_one\_table

Use this method to insert a single table into the worksheet. The format of the table is not important. You pass information about the format by specifying the name of an ABAP Dictionary object or supplying a description of the structure.

```
CALL METHOD spreadsheet->insert_one_table
  EXPORTING ddic_name = ddic_name
            data_table = data_table
            fields_table = fields_table
            rangename = rangename
            wholetable = wholetable
            updating = updating
            no_flush = no_flush
  IMPORTING error = error
            retcode = retcode.
```

### Description of parameters

Parameter	Optional	Description
ddic_name	X	Name of the ABAP Dictionary structure
data_table		Data table
fields_table		Description of the table structure (if you have not already passed it using the <code>ddic_name</code> parameter). Create the table with reference to the type <code>SOI_FIELDS_TABLE</code> .
rangename		Area name
wholetable	X	Inserts the entire table, regardless of the size of the range. (See also <code>SPREADSHEET-&gt;SPREADSHEET_INSERTALL</code> in <a href="#">soi_range_list [Page 136]</a> ).
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>

## set\_ranges\_data

Use this method to set data in ranges. You must insert the data from the table `contents` into the worksheet in the order defined in the `ranges` or `rangesdef` table. You can use control codes to access parts of individual ranges.

```
CALL METHOD spreadsheet->set_ranges_data
  EXPORTING ranges = ranges
            contents = contents
            rangesdef = rangesdef
            updating = updating
            no_flush = no_flush
```

**clear\_range**

IMPORTING error = error  
retcode = retcode.

**Description of parameters**

Parameter	Optional	Description
ranges		Table with all ranges  The table must be created with reference to the type <b>SOI_RANGE_LIST</b> (see the <a href="#">Important Table Structures [Page 136]</a> section).
contents		Data table  The table must be created with reference to the type <b>SOI_GENERIC_TABLE</b> (see the <a href="#">Important Table Structures [Page 136]</a> section).
rangesdef	X	An internal table with the type <b>SOI_DIMENSION_TABLE</b> with which you can define ranges by their starting position and size instead of their name.  <b>Note:</b> If you use this parameter, the <b>ranges</b> table (see above) must be <b>empty</b> .
updating	X	See <a href="#">The Updating Parameter [Page 109]</a>



The data in the contents table may not have leading spaces. You should therefore insert the data into the table using the WRITE statement to format it at the same time. Otherwise, the leading spaces will be considered when the column width is calculated in the [fit\\_widest \[Page 134\]](#) method.

**clear\_range**

Deletes the contents of the specified range. Although the contents are deleted, the range itself is not.

CALL METHOD spreadsheet->clear\_range  
EXPORTING name = name  
no\_flush = no\_flush  
IMPORTING error = error  
retcode = retcode.

**Description of parameters**

Parameter	Optional	Description
name		The range to be deleted.

## clear\_all\_ranges

Use this method to delete the contents and formatting of all of the ranges in the worksheet. The ranges themselves are **not** deleted. To delete a range definition from the worksheet, use the method [delete\\_ranges \[Page 129\]](#).

```
CALL METHOD spreadsheet->clear_all_ranges
    EXPORTING no_flush = no_flush
    IMPORTING error     = error
             retcode   = retcode.
```

## delete\_content\_ranges

Use this method to delete the contents of one or more named ranges in the worksheet.

```
CALL METHOD spreadsheet->delete_content_ranges
    EXPORTING ranges      = ranges
             no_flush    = no_flush
    IMPORTING error       = error
             retcode     = retcode.
```

### Description of parameters

Parameter	Optional	Description
ranges		Table containing the list of ranges.  The table must be created with reference to the type <code>SOI_RANGE_LIST</code> . However, in each entry, you only need to fill the <code>name</code> field.

## delete\_ranges

Use this method to delete the definitions of one or more ranges in the worksheet. The contents of the range are not deleted. To delete the contents of a range, use the method [delete\\_content\\_ranges \[Page 128\]](#).

```
CALL METHOD spreadsheet->delete_ranges
    EXPORTING ranges      = ranges
             no_flush    = no_flush
    IMPORTING error       = error
             retcode     = retcode.
```

### Description of parameters

hide\_columns

Parameter	Optional	Description
ranges		Table containing the names of the ranges.  The table must be created with reference to the type <b>SOI_FULL_RANGE_TABLE</b> . However, in each entry in the table, you only need to fill the <b>name</b> field.

## hide\_columns

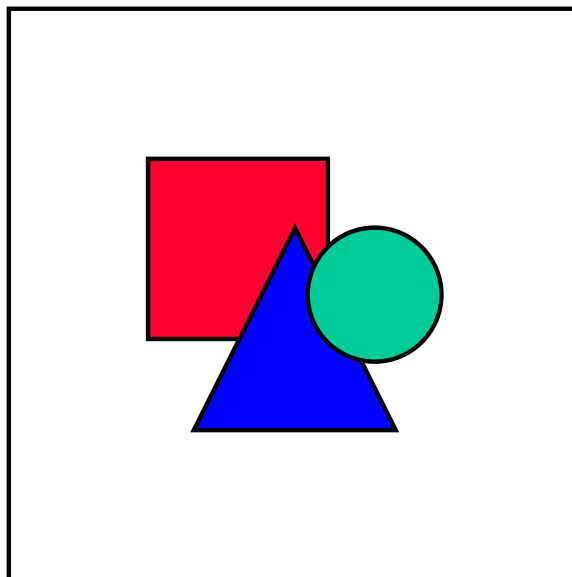
Hides the columns in the specified range.

CALL METHOD spreadsheet->hide\_columns

```
EXPORTING name      = name
          first     = first
          last      = last
          no_flush  = no_flush
IMPORTING error     = error
          retcode   = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		Name of the range
first	X	Number of the first column in the range you want to hide
last	X	Number of the last column in the range you want to hide



**first** and **last** only have an effect when you use both parameters. To hide a single column within the range, use the same value for both **first** and **last**.

## show\_columns

Shows the columns in the specified range (see also [hide\\_columns \[Page 129\]](#)).

```
CALL METHOD spreadsheet->show_columns
  EXPORTING name      = name
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		Name of the range

## hide\_rows

Hides the rows in the specified range. However, you can still access the contents using the [get\\_ranges\\_data \[Page 111\]](#) method.

```
CALL METHOD spreadsheet->hide_rows
  EXPORTING name      = name
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		Range to be hidden

## show\_rows

Makes the lines in the specified range visible.

```
CALL METHOD spreadsheet->show_rows
  EXPORTING name      = name
           no_flush   = no_flush
  IMPORTING error     = error
           retcode    = retcode.
```

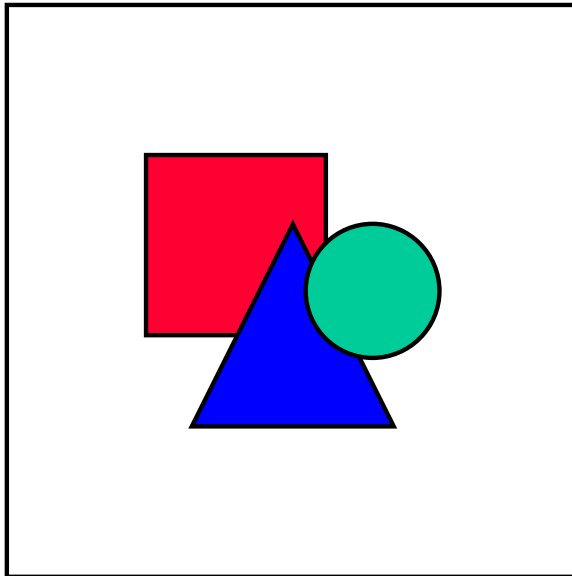
### Description of parameters

**set\_hierarchy**

Parameter	Optional	Description
name		Range to be displayed

**set\_hierarchy**

You use this method to set a hierarchy level in the worksheet. You must determine an area by specifying its starting row and number of rows. The user can then show or hide this area in the spreadsheet program.



To set more than one hierarchy using a single method call, use the method [set\\_hierarchy table \[Page 132\]](#).

```
CALL METHOD spreadsheet->set_hierarchy
  EXPORTING firstline = firstline
           length     = length
  IMPORTING error     = error
           retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
firstline		The first line of the hierarchy area (type I)
length		The length of the hierarchy area in lines (type I)

## set\_hierarchy\_table

Use this method to set a range of hierarchy levels in a worksheet. For each level, you specify its starting line and length in lines. The user can then show or hide the areas in the spreadsheet program.

```
CALL METHOD spreadsheet->set_hierarchy_table
    EXPORTING table      = table
             no_flush   = no_flush
    IMPORTING error      = error
             retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
table		An internal table with the type soi_hierarchy_table. For further information, refer to the <a href="#">Important Table Structures [Page 136]</a> section.

## clear\_hierarchy

Use this method to clear all of the hierarchy levels you have set in the worksheet.

```
CALL METHOD spreadsheet->clear_hierarchy
    EXPORTING no_flush = no_flush
    IMPORTING error    = error
             retcode   = retcode.
```

## print

Use this method to print the specified worksheet from the office application. You cannot specify any formatting (such as paper format) - this is adopted directly from the settings in your spreadsheet package.

```
CALL METHOD spreadsheet->print
    EXPORTING name      = name
             no_flush   = no_flush
    IMPORTING error     = error
             retcode    = retcode.
```

### Description of parameters

Parameter	Optional	Description
name		Name of the worksheet you want to print

**set\_zoom**

## set\_zoom

Use the `set_zoom` method to change the zoom of the current worksheet.

```
CALL METHOD spreadsheet->set_zoom
  EXPORTING zoom    = zoom
           no_flush = no_flush
  IMPORTING error   = error
           retcode  = retcode.
```

**Description of parameters**

Parameter	Optional	Description
zoom		Zoom factor for the current worksheet. The values are interpreted as percentages.

## fit\_widest

This method adjusts the width the cells in an area to the width of the widest cell contents.

```
CALL METHOD spreadsheet->fit_widest
  EXPORTING name    = name
           no_flush = no_flush
  IMPORTING error   = error
           retcode  = retcode.
```

**Description of parameters**

Parameter	Optional	Description
name		Name of the range you want to change

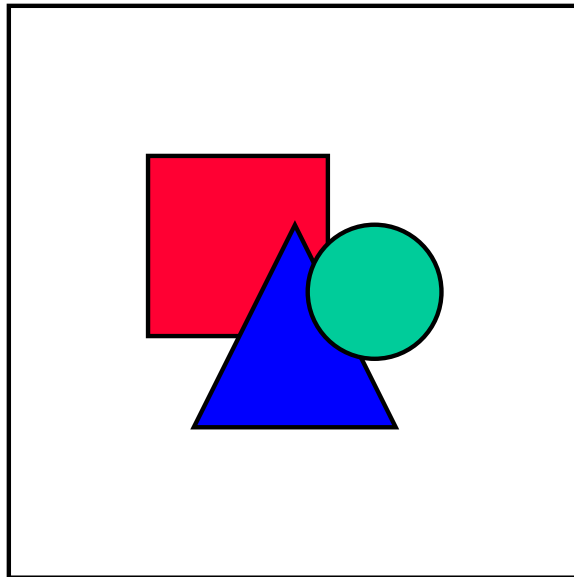


If you do not specify a range name, the function applies to the entire worksheet.

If you called the method [set\\_ranges\\_data \[Page 127\]](#) and the entries had leading spaces, the results may be unsatisfactory.

## screen\_update

Use this method to determine whether the spreadsheet display should be updated after each step. By switching off the screen update when you execute a whole set of method calls, you can improve the performance of your application.



Some methods have their own **UPDATING** parameter, which also controls whether the display is updated. This parameter will override the global setting that you make using **SCREEN\_UPDATE**.

```
CALL METHOD spreadsheet->screen_update
EXPORTING updating = updating
          no_flush = no_flush
IMPORTING error      = error
          retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
updating		Flag to specify whether the screen display should be updated. Possible values: ' ': Do not update 'x': Update

**load\_lib**

Use this method to load an additional DLL that can improve the performance of the following methods of the spreadsheet application:

- set\_ranges\_format
- cell\_format
- insert\_ranges

**version**

- `get_cell_formats`
- `get_dimension`



Loading this DLL constitutes a change to the document, so the user will be prompted to save, even if he or she has not changed it themselves.

```
CALL METHOD spreadsheet->load_lib
EXPORTING no_flush = no_flush
IMPORTING error      = errorversi
          retcode    = retcode.
```

**version**

This method returns the version of the spreadsheet interface. You can use this in your program to work out whether the current version of the interface supports all of the methods you have used.

```
CALL METHOD spreadsheet->version
EXPORTING no_flush = no_flush
IMPORTING version  = version
          error     = errorversi
          retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
version		A type C field containing the version of the spreadsheet interface

**Important Table Structures**

**Description of Data Type `soi_range_list`**

The methods for setting and getting data from a named range use the separation between the description of the range and the data itself. Note that the sequence must be observed both in the range description (structure `soi_range_list`) and in the data (structure `soi_generic_table`). This means that you must list all data from the first range before you can insert data into the second range.

**Structure `soi_range_list`**

Field	Type	Description
	e	

Important Table Structures

name	C	Name of the range
rows	C	Number of rows
columns	C	Number of columns
code	C	Function in the range: <ul style="list-style-type: none"> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_CLEAR</b>: Deletes range</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_COLUMNSHIDE</b>: Hides columns</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_ROWSHIDE</b>: Hides rows</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_PROTECT</b>: Range is protected</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_UNPROTECT</b>: Range is not protected</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_COLUMNSSHOW</b>: Columns are displayed.</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_ROWSSHOW</b>: Rows are displayed.</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_INSERTALL</b>: The entire table is inserted, regardless of the size of the area</li> <li>• <b>SPREADSHEET-&gt;SPREADSHEET_NEWRANGE</b>: Creates a new range</li> </ul>

The name identifies the range in the worksheet. This is, in effect, the key with which you always access the range. The size of the range is always given in columns and rows.

Some functions allow you to access a specific area in a worksheet. You can see from the table which functions are implemented.

### Description of Data Type `soi_generic_table`

In this table, you can save data from the range and use the [Data Provider \[Ext.\]](#) to transfer it to or retrieve it from the frontend. The data is transferred directly as a string with no type information.

#### Structure `soi_generic_table`

Field	Type	Description
row	C(4)	Row
column	C(4)	Column
value	C(256)	Value



The sequence of the data must correspond to the sequence of the range description, for example, `range1` before `range2`. The data table must then contain the data for the ranges in the sequence `range1 range2`.

### Description of Data Type `soi_format_table`

Use this table to specify the format of a range. The format consists of various attributes, all of which can be set in a single line. Each variable attribute corresponds to a column of the structure.

### Important Table Structures

To create a work area for this table, use the structure `soi_format_item` as a reference.



The entry “-1” always indicates that the existing attribute value for the range should not be changed.

#### Structure `soi_format_table`

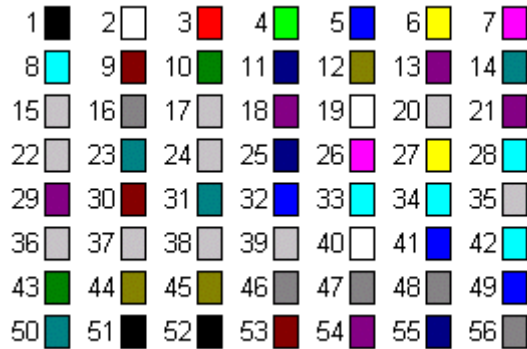
Field	Type	Description
name	C(256)	Name of the range
front	I	Font color (see color palette)
back	I	Background color (see color palette)
font	C(256)	Name of the font family. The following values are permitted: 'Arial' 'Courier New' 'Times New Roman'
size	I	Font size '-1': Unchanged
bold	I	'1': Bold '0': Normal '-1': Unchanged
italic	I	'1': Italic '0': Normal '-1': Unchanged
align	I	Alignment: <ul style="list-style-type: none"> <li>• '-1': Unchanged</li> <li>• '0': Right-justified</li> <li>• '1': Centered</li> <li>• '2': Left-justified</li> </ul>
frametype	I	Control byte for setting the frame '-1': Unchanged
framecolor	I	Frame color (see color palette) '-1': Unchanged
currency	C(3)	ISO standard currency code

Important Table Structures

number	I	<p>Specifies the format of a cell in a range.</p> <ul style="list-style-type: none"> <li>• 1: Display as a simple number</li> <li>• 2: Scientific display</li> <li>• 3: Display as a percentage</li> </ul>
--------	---	--



The following colors are permitted:



The control byte **type** contains the following bits. If a bit is set, its corresponding line is drawn. You can set the thickness of the line to one of four levels using bits 6 and 7.

Bit	Description
0	Sets the left margin
1	Sets the top margin
2	Sets the bottom margin
3	Sets the right margin
4	Horizontal line
5	Sets the left margin
6	Thickness
7	Thickness

### Description of Data Type `soi_full_range_table`

Each line of a table with the type `soi_full_range_table` specifies the full definition of a range. The individual lines have the data type `soi_full_range_item`.

#### Structure `soi_full_range_table`

Field	Type	Description
name	C(128)	Name of the range

**Important Table Structures**

top	I	Top row of the range
left	I	Leftmost column of the range
rows	I	Number of rows in the range
columns	I	Number of columns in the range
sheets	C(128)	Worksheet on which the range is defined

**Description of Data Type `soi_cell_table`**

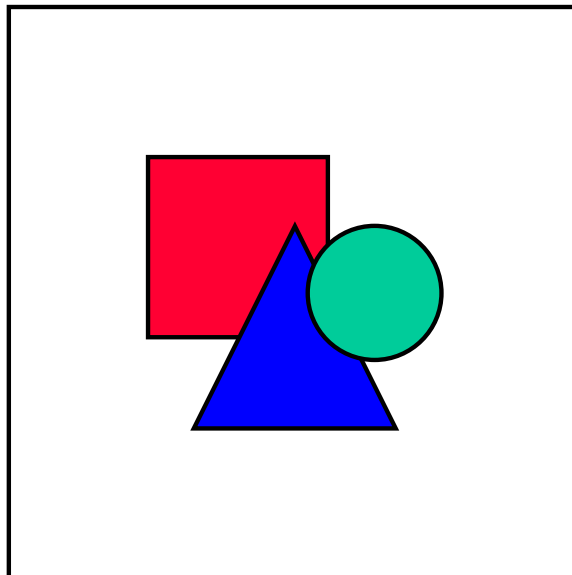
Each line of a table with the type `soi_cell_table` specifies the attributes of a range of cells. However, no range name is used. Instead, the cell area is defined by its starting position and the number of rows and columns it contains. The individual lines have the data type `soi_cell_item`.

**Structure `soi_cell_table`**

Field	Type	Description
top	I	Top row of the range
left	I	Leftmost column of the range
rows	I	Number of rows in the range
columns	I	Number of columns in the range
front	I	Font color (see color palette)
back	I	Background color (see color palette)
font	C(256)	Font. The following are permitted: <ul style="list-style-type: none"> <li>• Arial</li> <li>• Courier New</li> <li>• Times Roman</li> </ul>
size	I	Font size Use -1 if the font size is to remain unchanged.
bold	I	<ul style="list-style-type: none"> <li>• '1': Bold</li> <li>• '0': Normal</li> <li>• '-1': Unchanged</li> </ul>
italic	I	<ul style="list-style-type: none"> <li>• '1': Italic</li> <li>• '0': Normal</li> <li>• '-1': Unchanged</li> </ul>

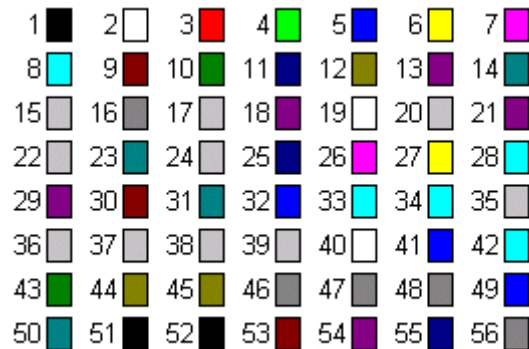
Important Table Structures

align	I	Alignment: <ul style="list-style-type: none"> <li>'-1': Unchanged</li> <li>'0': Right-justified</li> <li>'1': Centered</li> <li>'2': Left-justified</li> </ul>
frametype	I	Control byte for setting the frame '-1': Unchanged
framecolor	I	Frame color (see color palette) '-1': Unchanged
currency	C(3)	ISO standard currency code
number	I	Specifies the format of a cell in a range. <ul style="list-style-type: none"> <li>1: Display as a simple number</li> <li>2: Scientific display</li> <li>3: Display as a percentage</li> </ul>
decimals	I	Number of decimal places
input	I	<ul style="list-style-type: none"> <li>'0': Input off</li> <li>'1': Input on</li> </ul>



The following colors are permitted:

## The Pivot Interface



## Description of Data Type `soi_dimension_table`

You can use an internal table with this type to identify a range by specifying the coordinates of its top left-hand corner, its length, and its width. The lines of `soi_dimension_table` have the line type `soi_dimension_item`.

### Structure `soi_dimension_item`

Field	Type	Decription
top	I	Topmost row of the range
left	I	Leftmost column of the range
rows	I	Number of rows
columns	I	Number of columns

## The Pivot Interface

### Definition

You define this instance with reference to the interface `i_oj_pivot`.

### Use

The pivot interface allows you to insert the contents of an internal table directly into a pivot table in a spreadsheet program. It also contains methods that allow you to change the formatting of the pivot table from within your R/3 application.

### Integration

This interface currently supports Microsoft Excel 97 or 2000. It is not compatible with Lotus 1-2-3.

## Using the Pivot Interface

### Purpose

You use an instance of the pivot interface to communicate directly with a pivot table in Microsoft Excel.

### Prerequisites

Before you can use the pivot interface, you must create the central document management instance **document**. Microsoft Excel 97 or 2000 must be installed on the frontend. To find out whether the document type of the **document** instance supports the pivot interface, use the method [has\\_pivot\\_interface \[Page 59\]](#).

### Process Flow

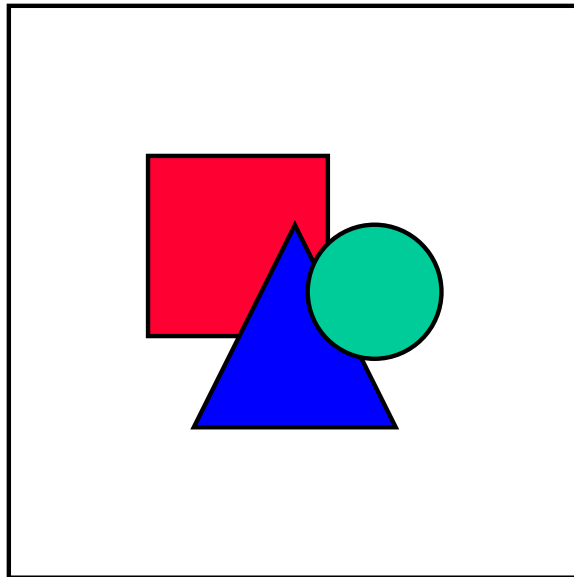
1. In the data declaration part of your program, declare a data object with reference to the interface *i\_oi\_pivot*:
 

```
DATA pivot TYPE REF TO i_oi_pivot.
```
2. Use the method [get\\_pivot\\_interface \[Page 59\]](#) on the document management instance to create the instance for the pivot interface.
3. Use the instance from step 2 and the following methods to work with the pivot table:

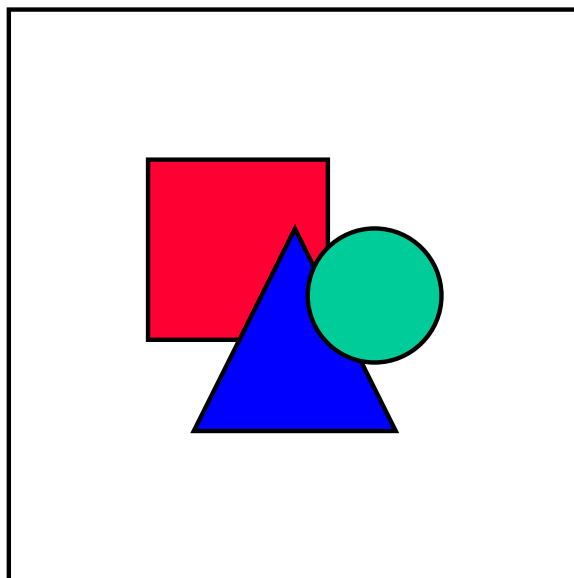
#### Methods of the pivot interface *i\_oi\_pivot*

Method	Function
<i>set_source_table</i>	Inserts the contents of an R/3 internal table into a pivot table
<i>set_fieldtype</i>	Reassigns the type of an existing field in the pivot table
<i>get_allfields</i>	Returns a list of fields in a specified pivot table
<i>get_allpivottables</i>	Returns a list of the names of all pivot tables in a workbook
<i>drill_down</i>	Expands the data display for a particular value to display more detail
<i>drill_up</i>	Collapses the data display for a particular value to display only a summary

**Generic Parameters**



Remember to include [error handling \[Page 15\]](#) after each method call.



Any parameters not contained in the individual method descriptions are listed under [Generic Parameters \[Page 144\]](#).

**Generic Parameters**

The following table summarizes the parameters that are used in several different methods within Desktop Office Integration.

**Description of parameters**

Parameter	Optional	Description
-----------	----------	-------------

**set\_source\_table**

no_flush	X	'X': Synchronizes the automation queue ' ': <a href="#">Automation queue [Ext.]</a> is synchronized before leaving the method.
retcode		<a href="#">Error handling [Page 15]</a>
error		<a href="#">Error handling [Page 15]</a>



If you call methods with the parameter `no_flush = 'X'`, the method is not sent to the presentation server directly, and is therefore not executed immediately.

Instead, the methods are buffered until the next method call with the parameter `no_flush = ' '` occurs or you explicitly call the method [cl\\_gui\\_cfw=>flush \[Ext.\]](#). If you use this technique correctly it can improve performance. However, if you use it incorrectly, inconsistencies may arise in your data. For further information, refer to the [Automation Queue \[Ext.\]](#) section.

## set\_source\_table

Use this method to create a new pivot table in the current workbook using the contents of an internal table. The system creates the pivot table on a new worksheet.

The method allows you to select which fields of the internal table you want to appear in the pivot table. You must also decide whether they should be row fields, column fields, or data fields.

```
CALL METHOD pivot->set_source_table
EXPORTING ddic_name      = ddic_name
          data_table     = data_table
          fields_table   = fields_table
          name           = name
          items          = items
          itemsnr        = itemsnr
          length         = length
IMPORTING error          = error
          retcode        = retcode.
```

### Description of parameters

Parameter	Optional	Description
ddic_name	X	The name of a table or structure in the ABAP Dictionary with the same structure as the internal table you want to insert in the workbook.  <b>Note:</b> If your internal table does not have the structure of an ABAP Dictionary table, you must use the <code>fields_table</code> parameter instead.
data_table		The name of the internal table in your ABAP program containing the data that you want to display in the pivot table

**set\_source\_table**

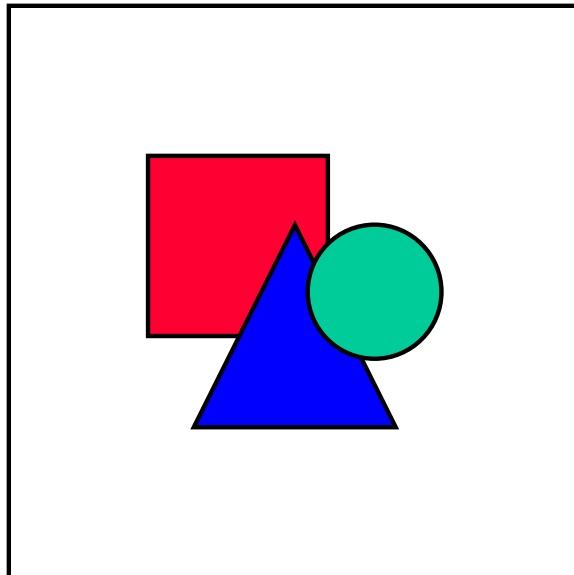
fields_table	X	You use this parameter when you cannot use the <code>ddic_name</code> parameter to describe the structure of the data table. It is an internal table with the type <code>SOI_FIELDS_TABLE</code> , which you must fill with the names of the fields in the data table.
name		The name to be given to the pivot table
items	X	<p>If you described your data table using the <code>fields_table</code> parameter, you can use this parameter to specify the fields of the table that should appear in the pivot table, and whether they should be row fields, column fields, or data fields. It is an internal table with the type <code>SOI_NAME_TYPE_TABLE</code>. This has two fields: <code>name</code>, containing the name of the field, and <code>type</code>, which describes the type of the field. <code>type</code> can have the following values:</p> <p><code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a row in the pivot table</p> <p><code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a column in the pivot table</p> <p><code>i_oi_pivot_datafield</code>: The values of the field appear in the body of the table itself.</p> <p><b>Note:</b> If you want to specify the fields by their index number and not by name, use the <code>itemsnr</code> parameter instead.</p>
itemsnr	X	<p>Use this parameter to specify the fields of the table that should appear in the pivot table, and whether they should be row fields, column fields, or data fields. It is an internal table with the type <code>SOI_COLTYPE_TABLE</code>. This has two fields: <code>col</code>, containing the index number of the field from the internal table, and <code>type</code>, which describes the type of the field. <code>type</code> can have the following values:</p> <p><code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a row in the pivot table</p> <p><code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a column in the pivot table</p> <p><code>i_oi_pivot=&gt;datafield</code>: The values of the field appear in the body of the table itself.</p> <p><b>Note:</b> If you used the <code>ddic_name</code> parameter, you must use this parameter to define how the fields of the data table should be arranged in the pivot table. If you used the <code>fields_name</code> parameter, you may use the <code>items</code> parameter instead, which allows you to specify the fields by name instead of by index number.</p>

## set\_fieldtype

length	X	<p>If you used the <code>ddic_name</code> parameter to specify the structure of your data table, you can use this parameter to specify the length of the field labels that the system should use in the pivot table. These are taken from the definitions of the underlying data elements in the ABAP Dictionary. Possible values:</p> <ul style="list-style-type: none"> <li>• 'S': Short field labels</li> <li>• 'M': Medium field labels</li> <li>• 'L': Long field labels.</li> </ul> <p><b>Note:</b> If you specify a value that is not valid (or no value at all) the system cannot display the pivot table correctly.</p> <p>You cannot use this parameter if you used <code>fields_table</code> to describe the structure of your data table.</p>
--------	---	---

## set\_fieldtype

Use this method to set or change the field type of a field in the pivot table. You can, for example, change the type of a field (row, column, or data field), or its summarization type (sum field, maximum, minimum, count, or average).



You can only use this method with fields that are already in the pivot table. To add or remove fields from the table, use the method [set\\_source\\_table \[Page 145\]](#).

```
CALL METHOD pivot->set_fieldtype
  EXPORTING name      = name
           type       = type
           pivot      = pivot
           consolidation = consolidation
           no_flush   = no_flush
```

**get\_allfields**

```
IMPORTING error          = error
          retcode       = retcode.
```

**Description of parameters**

Parameter	Optional	Description
name		Name of the field in the pivot table
type		New type of the field. Possible values: <ul style="list-style-type: none"> <li>• <code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a row of the pivot table</li> <li>• <code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a column of the pivot table</li> <li>• <code>i_oi_pivot=&gt;datafield</code>: The field is part of the body of the table</li> </ul>
pivot		The name of the pivot table in the workbook
consolidation		The summarization type of the field. Possible values: <ul style="list-style-type: none"> <li>• <b>ADD</b>: Sum</li> <li>• <b>CNT</b>: Count</li> <li>• <b>MIN</b>: Minimum value</li> <li>• <b>MAX</b>: Maximum value</li> <li>• <b>AVG</b>: Mean value</li> <li>• <b>NOF</b>: No special summarization option</li> </ul>

**get\_allfields**

Use this method to return a list of the fields in the pivot table.

```
CALL METHOD pivot->get_allfields
EXPORTING pivottable = pivottable
          no_flush   = no_flush
IMPORTING allfields  = allfields
          error      = error
          retcode    = retcode.
```

**Description of parameters**

Parameter	Optional	Description
pivottable		The name of the pivot table for which you want to find out the field names
allfields		An internal table (with the type <code>soi_nametype_table</code> ) containing the names of the fields in the pivot table.

## get\_allpivottables

Use this method to return a list of all of the pivot tables in the current workbook.

```
CALL METHOD pivot->get_allpivottables
    EXPORTING no_flush      = no_flush
             allpivottables = allpivottables
             error          = error
             retcode        = retcode.
```

### Description of parameters

Parameter	Optional	Description
allpivottables		An internal table with the type <code>soi_fields_name_table</code> containing the names of the pivot tables in the worksheet in the order in which they occur.

## drill\_down

Use this method to expand a summarized row of a pivot table so that its constituent rows are displayed.

```
CALL METHOD pivot->drill_down
    EXPORTING row          = row
             value         = value
             pivottable    = pivottable
             no_flush      = no_flush
    IMPORTING error        = error
             retcode       = retcode.
```

### Description of parameters

Parameter	Optional	Description
row		The name of the field in the pivot table by which the data is grouped. <b>Note:</b> The name is the <b>field label</b> as it appears in the header of the pivot table, <b>not</b> the field name from the ABAP Dictionary.
value		The value of the field that you specified in <code>row</code> which you want to expand the entries in the pivot table
pivottable		The name of the pivot table in the workbook

**drill\_up****drill\_up**

Use this method to collapse a set of rows in the pivot table so that only a summary line is displayed.

```
CALL METHOD pivot->drill_up
  EXPORTING row      = row
           value     = value
           pivottable = pivottable
           no_flush  = no_flush
  IMPORTING error    = error
           retcode   = retcode.
```

**Description of parameters**

Parameter	Optional	Description
row		The name of the field in the pivot table by which the data is grouped. <b>Note:</b> The name is the <b>field label</b> as it appears in the header of the pivot table, <b>not</b> the field name from the ABAP Dictionary.
value		The value of the field that you specified in <b>row</b> which you want to hide the entries in the pivot table
pivottable		The name of the pivot table in the workbook

**Important Table Structures**

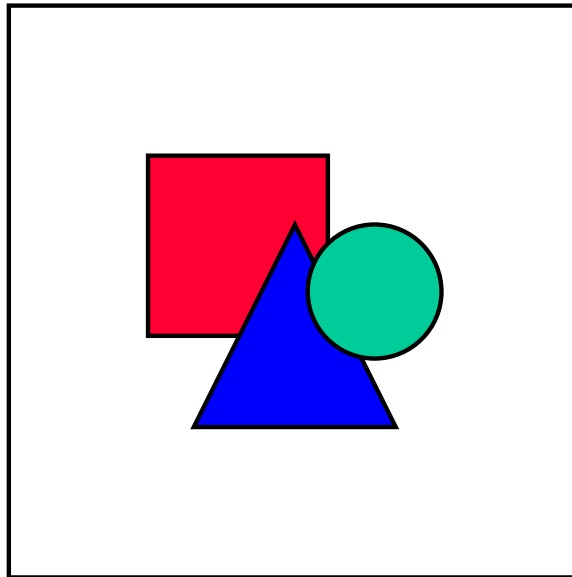
The pivot interface uses the following data types, which are defined in type group SOI (development class SOFFICEINTEGRATION):

**Data Type SOI\_NAMETYPE\_ITEM**

Field	Type	Description
name	C(256)	The name of a field in the data table

Important Table Structures

type	I	<p>The type that the field should have in the pivot table. The following values are possible:</p> <ul style="list-style-type: none"> <li>• <code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a new row in the pivot table</li> <li>• <code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a new column in the pivot table</li> <li>• <code>i_oi_pivot=&gt;datafield</code>: The values of the field appear in the body of the table itself</li> </ul>
------	---	---

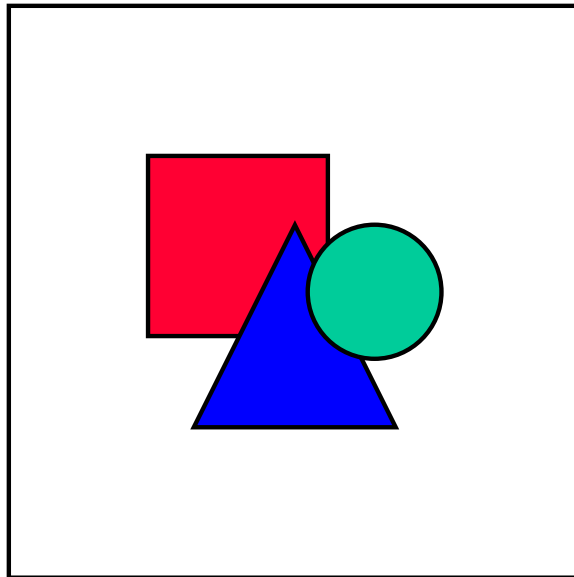


This data type also has a corresponding internal table type called `SOI_NAMETYPE_TABLE`.

### Data Type `SOI_COLTYPE_ITEM`

Field	Type	Description
col	I	The position of a field in the data table
type	I	<p>The type that the field should have in the pivot table. The following values are possible:</p> <ul style="list-style-type: none"> <li>• <code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a new row in the pivot table</li> <li>• <code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a new column in the pivot table</li> <li>• <code>i_oi_pivot=&gt;datafield</code>: The values of the field appear in the body of the table itself</li> </ul>

## Test Tools for Desktop Office Integration



This data type also has a corresponding internal table type called `SOI_COLTYPE_TABLE`.

## Test Tools for Desktop Office Integration

### Purpose

The test tools for Desktop Office Integration are intended as a diagnostic tool for finding errors in the installation and registration of the desktop components necessary for SAP Desktop Office Integration. Should you ever need to report problems with your office integration applications, the results that these tests generate will help SAP to trace the cause of the error.

### Implementation Considerations

The necessary files for the tests are installed automatically on your frontend computer when you install SAPgui version 4.6C.

### Integration

The test tools only examine the Desktop Office Integration components that are installed on the frontend. You can therefore run them without being logged onto an R/3 System.

### Features

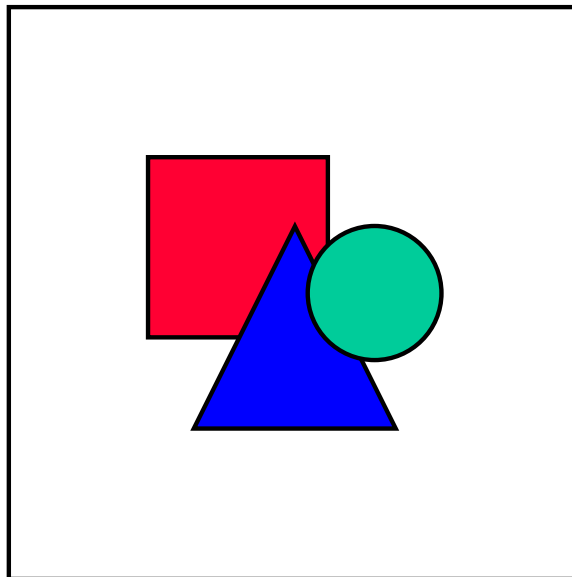
There are four tools in all, all of which are installed in the *TestTools* directory of your SAPgui installation. The tools have the following functions:

Tool	Function
<a href="#">check_doi.exe</a> [Page 153]	Checks the paths, installation, and Registry keys for desktop office applications and the SAP Desktop Office Integration OCX files. Runs as a standalone program.

**DOI Installation Test Using a Standalone Executable Program**

<a href="#">doi_word_check.doc</a> <a href="#">[Page 154]</a>	Checks the paths, installation, and Registry keys for desktop office applications and the SAP Desktop Office Integration OCX files. Runs as a macro in a Word document.
<a href="#">doi_xls_check.xls</a> <a href="#">[Page 154]</a>	Checks the paths, installation, and Registry keys for desktop office applications and the SAP Desktop Office Integration OCX files. Runs as a macro in an Excel workbook.
<a href="#">TableFactoryTest.doc</a> <a href="#">[Page 155]</a>	Checks the SAP TableFactory OCX installation.

The first three of these tools create a list of results that you can save and make available to SAP support if you have problems with your Desktop Office Integration applications.



The lists created by the tools contain large amounts of technical information. SAP does not expect you to analyze this information yourself, merely to send it to support staff if requested.

## DOI Installation Test Using a Standalone Executable Program

### Use

This program checks the paths, installation, and Registry keys of the desktop office applications and OCX files used in SAP Desktop Office Integration. Its results are useful to SAP if you report problems using Desktop Office Integration.

### Prerequisites

You must be using SAPgui version 4.6C or higher.

## DOI Installation Test Within Word

### Activities

The program `check_doi.exe` is installed under `... \TestTools\vbprojects` in the directory on your PC in which the SAPgui is installed.

1. Run the program from this location.
2. In the program window, select the Microsoft Office version that you are using, then choose *Go*.
3. The program generates the results list. When it has finished, choose *Save output as*. A dialog box appears.
4. Enter a file name for the results, and choose *Save*.

You can send this list to SAP if requested.

## DOI Installation Test Within Word

### Use

The Word document `doi_word_check.doc` contains a macro that checks the paths, installation, and Registry keys of the desktop office applications and OCX files used in SAP Desktop Office Integration. Its results are useful to SAP if you report problems using Desktop Office Integration.

### Prerequisites

You must be using SAPgui version 4.6C or higher and Microsoft Office 97 or Office 2000.

### Activities

The document `doi_word_check.doc` is installed under `... \TestTools\macros` in the directory on your PC in which the SAPgui is installed.

1. Open the document from this location.
2. Choose *Tools* → *Macros* → *Macros*.
3. Select *CheckSettings* and choose *Run*. The system checks the Office Integration environment and creates the result list.
4. Choose *Save*, enter a name and choose *Save* again to save the list. You can send it to SAP if requested.

## DOI Installation Test Within Excel

### Use

The Excel workbook `doi_xls_check.xls` contains a macro that checks the paths, installation, and Registry keys of the desktop office applications and OCX files used in SAP Desktop Office Integration. Its results are useful to SAP if you report problems using Desktop Office Integration.

### Prerequisites

You must be using SAPgui version 4.6C or higher and Microsoft Office 97 or Office 2000.

## Activities

The Excel workbook `doi_xls_check.xls` is installed under `... \TestTools\macros` in the directory on your PC in which the SAPgui is installed.

1. Open the workbook from this location.
2. Choose *Tools* → *Macro* → *Macros*.
3. Select *CheckSettings* and choose *Run*. The system checks the Office Integration environment and creates the result list.
4. Choose *Save*, enter a name and choose *Save* again to save the list. You can send it to SAP if requested.

## Test for the TableFactory OCX

### Use

The TableFactory OCX is an SAP frontend component that is used extensively by Desktop Office Integration. The Word document `TableFactoryTest.doc` allows you to check whether it is functioning correctly on your PC.

### Prerequisites

You must be using SAPgui version 4.6C or higher and Microsoft Office 97 or Office 2000.

### Features

The Word document `TableFactoryTest.doc` is installed under `... \TestTools\macros` in the directory on your PC in which the SAPgui is installed. It has two macros:

- **Test**, which tests the SAP TableFactory OCX, and
- **Clear**, which clears the test results from the document.

### Activities

1. Open the document and choose *Tools* → *Macro* → *Macros*.
2. Select *Test* and choose *Run*. The system tests the TableFactory OCX by inserting text into the first table in the document. This should be exactly the same as the text in the second table.

You can clear the text from the first table by running the clear macro.

## The Document Viewer

### Purpose

The Document Viewer is a new feature of Office Integration in Release 4.6C. It allows you to display documents either in a browser within the R/3 window or in a separate browser window.

## The Document Viewer

### Implementation Considerations

You can use the Document Viewer whenever you want the user to be able to display documents but not change them. Unlike conventional Office Integration techniques, it is supported by the SAPGUI for HTML environment.

### Integration

You must be using a version of the SAPgui with Release 4.6C or higher. The Document Viewer is compatible with both SAPGUI for Windows (Windows 95/98/NT) and SAPGUI for HTML.

The following must also be installed on your PC:

- Microsoft Internet Explorer version 4 or higher
- The desktop application relevant to each document type you want to display (for example, Word if you want to display DOC files)

You can use the SAP Business Document Service to store documents. The URLs or internal tables returned by the BDS can be used directly in the methods of the Document Viewer interface.

### Features

The Document Viewer allows you to display documents either in a browser within the SAPGUI (either for Windows or HTML) or in a separate browser window.

Unlike conventional Office Integration, it is compatible with the SAPGUI for HTML.

The ABAP programming interface for the Document Viewer is provided by the global interface `i_oi_document_viewer`.

The Document Viewer supports HTML plus the following additional document types:

Document Type	Document Extension	ProgID in Registry
Microsoft Word document	.DOC	Word.Document
Microsoft Excel workbook	.XLS	Excel.Sheet
Microsoft Powerpoint presentation	.PPT	PowerPoint.Show
Rich Text Format	.RTF	The application you have associated with files of this type using the <i>Open With</i> dialog box in Windows
Portable Document Format	.PDF	The application you have associated with files of this type using the <i>Open With</i> dialog box in Windows
Quicktime Movie	.MOV	Quicktime.Mov
Lotus 123 Workbook	123	Lotus123.Workbook
Lotus Word Pro Document	LWP	Wordpro.Document
Visio Drawing	VSD	Visio.Drawing



These are the document types that are definitely supported. It may be possible to display others, depending on the capabilities of the underlying Web browser.

The Document Viewer relies on the plug-in mechanism of the underlying Web browser, which can work out the format of the document and the application it needs to launch in order to display it. To do this, it uses the following information from the Windows Registry:

HKEY_CLASSES_ROOT\.***	Content type & ProgID
HKEY_CLASSES_ROOT\ <progid>\shell\open\command</progid>	Location of the application
HKEY_CLASSES_ROOT\MIME\database\content Type	Extension

These entries can be generated in one of the following ways:

- When you install an application. For example, when you install Word, the file type .DOC is registered to it
- When you first open a file with an unregistered extension. In this case, the *Open With* dialog box appears, in which you have to associate an application with the extension.

## Constraints

It is not currently possible to edit and save documents in the Document Viewer.

## Example

The SAP System contains an example program called `SAPRDEMOVIEWING`.

# Using the Document Viewer

## Purpose

Use the Document Viewer when you want to allow the user to display a document but not change it.

In the SAPGUI for HTML, the Document Viewer is the only available variant of SAP Desktop Office Integration.

## Prerequisites

In order to use the Document Viewer, you should be familiar with [ABAP Objects \[Ext.\]](#) and the concept of [SAP Container Controls \[Ext.\]](#).

## Process Flow



For an extract of the required ABAP coding, see the [Coding Example \[Page 158\]](#).

## Coding Example

1. You declare reference variables for a SAP Container (which will accommodate the Document Viewer) and the Document Viewer instance itself.
2. You instantiate the SAP Container Control in the normal way, and get a reference to the Document Viewer using the method  
`C_OI_CONTAINER_CONTROL_CONTAINER=>GET_DOCUMENT_VIEWER.`  
 This method returns a reference to an instance of the Document Viewer. From now on, you work with the methods of this instance.
3. You start the viewer instance using the instance method [INIT\\_VIEWER \[Page 159\]](#).
4. You load the document into the Document Viewer using either [VIEW\\_DOCUMENT\\_FROM\\_URL \[Page 160\]](#) or [VIEW\\_DOCUMENT\\_FROM\\_TABLE \[Page 161\]](#).
5. You close the document using the [CLOSE\\_DOCUMENT \[Page 162\]](#) method.
6. Once you are finished with the Document Viewer instance, you destroy it using the [DESTROY\\_VIEWER \[Page 162\]](#) method.

## Coding Example

This extract represents only the steps listed in [Using the Document Viewer \[Page 157\]](#), and does not include tasks such as retrieving a document from the BDS. It assumes that you are going to display a document in a SAP Custom Container on screen 100 of a program. You would have to create the Custom Container area in the Screen Painter. We have called the container 'CONTAINER100'.



For precise details of the methods, refer to the individual method documentation. For details of how to use SAP Container controls, refer to the [SAP Container \[Ext.\]](#) documentation.

## Data Declarations

You must declare the following data objects in your program:

```
DATA: custom_container TYPE REF TO cl_gui_custom_container,
      document_viewer  TYPE REF TO i_oi_document_viewer.
```

## Instantiating the Container and Document Viewer

This is best done in a PBO module. **It must only happen once.** You can ensure this by checking whether one of the controls (for example, `custom_container`) is still initial, and only executing the following statements if it is.

```
IF custom_container IS INITIAL.
  CREATE OBJECT custom_container
    EXPORTING container_name = 'CONTAINER100'.
```

Methods of the Document Viewer

```
CALL METHOD C_OI_CONTAINER_CONTROL_CREATOR=>GET_DOCUMENT_VIEWER
      IMPORTING viewer = document_viewer.

CALL METHOD document_viewer->init_viewer
      EXPORTING parent = custom_container.

ENDIF.
```

### Loading the Document

You can do this either in the PBO module or in the PAI event in reaction to a user action.

```
CALL METHOD document_viewer->view_document_from_url
      EXPORTING document_url = <url of the document>
              show_inplace = 'X'.
```

### Destroying the Control

```
CALL METHOD document_viewer->destroy_viewer.

CALL METHOD custom_container->free.

FREE: document_viewer, custom_container.
```

## Methods of the Document Viewer

Once you have instantiated the Document Viewer, you can work with it by calling its methods:

Method	Function
init_viewer	Initializes the Document Viewer instance
destroy_viewer	Destroys the Document Viewer instance
view_document_from_url	Displays a document referenced by a particular URL (either a physical location, such as HTTP://..., FTP://..., or FILE://..., or a generated SAPR3:// address from the BDS)
view_document_from_table	Displays a document stored in an internal table
close_document	Closes the last document to have been opened

### init\_viewer

Use this method to initialize the Document Viewer instance and attach it to the relevant SAP Container. Note that this method does not open a document. Instead, you must call one of the separate methods [view document from url \[Page 160\]](#) or [view document from table \[Page 161\]](#).

```
CALL METHOD document_viewer->init_viewer
      EXPORTING parent = parent
      EXCEPTIONS cntl_error = 1
                 cntl_install_error = 2
                 dp_install_error = 3
                 dp_error = 4
```

view\_document\_from\_url

Parameter and Type	Optional	Meaning
parent TYPE REF TO CL_GUI_C ONTAINER		The reference variable pointing to the SAP Container instance in which you want the Document Viewer to appear

## view\_document\_from\_url

Use this method to display a document in the Document Viewer. You identify the document by its Uniform Resource Locator (URL). The URL can be generated (by the BDS), but can also have one of the forms HTTP://, FTP://, or FILE://. You must already have initiated the Document Viewer ([init\\_viewer \[Page 159\]](#) method).



You should use the Business Document Service (BDS) to manage your documents. Its programming interface contains a method that returns the URL of a document. You can use the URL directly in the `document_url` parameter of this method.

```
CALL METHOD document_viewer->view_document_from_url
    EXPORTING document_url      = document_url
              show_inplace    = show_inplace
    EXCEPTIONS cntl_error      = 1
               not_initialized = 2
               dp_error_general = 3
               invalid_parameter = 4.
```

Parameter and Type	Optional	Meaning
document_url TYPE C		The URL of the document you want to display
show_inplace TYPE C		Indicates whether the document should be displayed in the R/3 window or a separate browser. Possible values: <ul style="list-style-type: none"> <li>• 'X': Display in R/3 window</li> <li>• ' ': Display in separate browser</li> </ul>

## view\_document\_from\_table

Use this method to display a document that is stored in an internal table in your ABAP program. To retrieve a document from the Business Document Service as an internal table, use the method `get_with_table` of class `cl_bds_document_set`.



You are strongly recommended to use the Business Document Service in conjunction with this method. Although it is possible to use it with standalone documents that you store yourself in the database, you then have to make sure that the parameters are correctly filled. If you use the BDS, the parameter values are provided automatically by the `get_with_table` method.

```
CALL METHOD document_viewer->view_document_from_table
  EXPORTING show_inplace = show_inplace
            type          = type
            subtype       = subtype
            size          = size
  CHANGING document_table = document_table
  EXCEPTIONS dp_invalid_parameter = 1
            dp_error_general      = 2
            cntl_error            = 3
            not_initialized       = 4
            invalid_parameter     = 5.
```

Parameter and Type	Optional	Meaning
show_inplace TYPE C		Indicates whether the document should be displayed in the R/3 window or a separate browser. Possible values: <ul style="list-style-type: none"> <li>'X': Display in R/3 window</li> <li>' ': Display in separate browser</li> </ul>
type TYPE C		The MIME type of the document. <b>Note:</b> See <a href="#">Filling the Parameters Correctly</a> below.
subtype TYPE C		The MIME subtype of the document. <b>Note:</b> See <a href="#">Filling the Parameters Correctly</a> below.
size TYPE I		The size of the document in bytes. <b>Note:</b> See <a href="#">Filling the Parameters Correctly</a> below.
document_table TYPE STANDARD TABLE		The internal table containing the document

### Filling the Parameters Correctly

You can retrieve the type, subtype, size and document\_table parameters for this method from the EXPORTING parameters of the `get_with_table` method of `cl_bds_document_set`.

## close\_document

However, the mapping between the parameters is not one-to-one. `get_with_table` returns the following relevant parameters:

Parameter and Type	Meaning
content TYPE SBDST_CONTENT	The internal table containing the document content. This can be passed directly to the <code>document_table</code> parameter of <code>view_document_from_table</code> .
components TYPE SBDST_COMPONENTS	A components table containing various administrative information about the document, including the MIME type and the size.

## Retrieving the MIME Type and Subtype

After you have called the `get_with_table` method of `cl_bds_document_set`, the MIME type and subtype of the document are contained in the field `components-mimetype` in the form `<type>/<subtype>`. To separate them, use the statement `SPLIT ... AT '/' INTO field1 field2`.

## Retrieving the File Size

The size of the document is contained in the field `components-comp_size`. You can pass this value directly to the `size` parameter of `view_document_from_table`.

## close\_document

Use this method to close a document that has been open in the Document Viewer. It always closes the **last** document that you opened.

```
CALL METHOD document_viewer->close_document
      EXCEPTIONS cntl_error      = 1
                 not_initialized = 2
```

## destroy\_viewer

Use this method to destroy the Document Viewer instance. After destroying the instance, free the memory occupied by the reference variable using the `FREE` statement.

```
CALL METHOD document_viewer->destroy_viewer
      EXCEPTIONS not_initialized = 1
                 free_failed    = 2.
```