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

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caution</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td></td>
</tr>
<tr>
<td>Recommendation</td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
</tr>
</tbody>
</table>
## Contents

**Desktop Office Integration (BC-CI)** ................................................................. 10

- 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

**Error Handling after Calling a Method** ......................................................... 15

- Error Messages and Their Meanings ......................................................... 17
- raise_message ................................................................................................. 18
- get_message ................................................................................................. 19

**Central Instance for Managing SAP DOI** .................................................... 19

- 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

**Documents** .................................................................................................. 31

- Working With Documents ............................................................................. 31
  - Methods of the Interface i_oi_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
    - openactivex_document ........................................................................... 42
    - play_document ........................................................................................ 42
    - play_document_from_table ..................................................................... 43
    - save_document_to_url .......................................................................... 44
Exchanging Data Between R/3 and the Desktop Application

The Link Server

Working With the Link Server

Processing Office Application Events

Event Handling Using Static Methods

Event Handling Using Instance Methods

get_document_cookie

get_script_collection

get_pivot_interface

has_pivot_interface

get_mail_merge_interface

has_mail_merge_interface

get_form_interface

has_form_interface

get_spreadsheet_interface

has_spreadsheet_interface

update_document_links

get_application_property

get_document_handle

get_document_type

get_wordprocessor_interface

has_wordprocessor_interface

get_documentor_interface

get_documentor_type

get_documentor_handle

get_application_property

update_document_links

has_spreadsheet_interface

get_spreadsheet_interface

has_spreadsheet_interface

update_document_links

has_mail_merge_interface

get_mail_merge_interface

has_mail_merge_interface

update_document_links

has_pivot_interface

get_pivot_interface

has_pivot_interface

update_document_links

get_script_collection

get_document_cookie
Desktop Office Integration (BC-CI)

Generic Parameters.................................................................66
start_link_server...................................................................66
stop_link_server...................................................................66
add_string_item.....................................................................67
add_table_item.......................................................................67
add_table_item2.....................................................................68
add_table_item2.....................................................................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
The Word Processor Interface................................................80
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
The Form Interface....................................................................91
Working With The Form Interface.........................................93
Generic Parameters...............................................................94
set_data.................................................................................94
get_data................................................................................95
add_field...............................................................................95
delete_fields.........................................................................96
disconnect_fields...................................................................97
get_form_fields......................................................................97
protect.....................................................................................97
### The Mail Merge Interface

**Using the Mail Merge Interface**

- Generic Parameters
  - set_modus

- The Updating Parameter

### The Script Collection

**Using the Script Collection**

- Generic Parameters
  - set_modus
  - add_script
  - set_data_source
  - get_fields
  - merge_one
  - merge_range
  - print
  - view
  - shutdown

### The Spreadsheet Interface

**Using the Spreadsheet Interface**

- Generic Parameters
  - set_modus
  - get_data_source
  - set_field
  - set_data
  - get_ranges_names
  - get_ranges_data
  - get_active_sheet
  - get_protection

- The Updating Parameter

- Generic Parameters
add_sheet ......................................................................................................................... 123
set_sheet_name .................................................................................................................. 123
delete_sheet........................................................................................................................ 123
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

The Pivot Interface ............................................................................................................... 142
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
delete_sheet ......................................................................................................................... 150
Important Table Structures ................................................................................................. 150

Test Tools for Desktop Office Integration ........................................................................... 152
DOI Installation Test Using a Standalone Executable Program ............................................. 153
DOI Installation Test Within Word ....................................................................................... 154
DOI Installation Test Within Excel ..................................................................................... 155
Test for the TableFactory OCX .......................................................................................... 155

The Document Viewer ....................................................................................................... 155
Using the Document Viewer ............................................................................................... 157
Coding Example ................................................................................................................... 158
Methods of the Document Viewer ......................................................................................... 159
init_viewer .............................................................................................................................................159
view_document_from_url ......................................................................................................................160
view_document_from_table ..................................................................................................................161
close_document ....................................................................................................................................162
destroy_viewer ......................................................................................................................................162
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:
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.]).

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:
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
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_oii_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_oii_errors* as described in process I.

   Example of process II:
   ```
   IF ret_code EQ c_oii_errors=>ret_ok.
   " Document opened successfully
   ELSEIF ret_code EQ c_oii_errors=> ret_document_already_open.
   " Special error handling, e.g. dialog box.
   ELSE.
   CALL METHOD c_oii_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_oi_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:

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message_id</td>
<td></td>
<td>Message class</td>
</tr>
<tr>
<td>message_number</td>
<td></td>
<td>Message number</td>
</tr>
<tr>
<td>param1 to param4</td>
<td></td>
<td>Variable part of message</td>
</tr>
</tbody>
</table>

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 \texttt{i\_oi\_container\_control}:
   \begin{verbatim}
   DATA control TYPE REF TO i\_oi\_container\_control.
   \end{verbatim}

3. Declare an object variable for all of the documents that you want to have open at once.
   \begin{verbatim}
   DATA: document TYPE REF TO i\_oi\_document\_proxy.
   \end{verbatim}

4. Declare an object variable for the data transfer (optional):
   \begin{itemize}
   \item \textbf{If you are working with the Link Server:}
     \begin{verbatim}
     DATA: link\_server TYPE REF TO i\_oi\_link\_server.
     \end{verbatim}
   \item \textbf{If you are working with the Table Collection:}
     \begin{verbatim}
     DATA: table\_coll TYPE REF TO i\_oi\_table\_collection.
     \end{verbatim}
   \end{itemize}

Initialization

The following steps must only occur once in your program.

5. Create the instance \textit{control}.
   \begin{verbatim}
   CALL METHOD c\_oi\_container\_control\_creator=>get\_container\_control
     IMPORTING control = control
     retcode = retcode.
   \end{verbatim}

6. If you want to use Desktop Office Integration in-place, you also need to create a container [Ext.]:
   \begin{verbatim}
   DATA: container TYPE REF TO cl\_gui\_custom\_container.
   CREATE OBJECT container
     EXPORTING container\_name = 'CONTAINER'.
   \end{verbatim}

7. Call the method \textit{init\_control}.
   You have now created the central object for Desktop Office Integration and the connection to the relevant GUI control.
   \begin{verbatim}
   CALL METHOD control->init\_control
     EXPORTING r3\_application\_name = 'Demo Document Container'
      inplace\_enabled = 'X'
      parent = container
     IMPORTING retcode = retcode.
   \end{verbatim}

8. Create an instance document for each document that you want to open:
   \begin{verbatim}
   CALL METHOD control->get\_document\_proxy
     EXPORTING document\_type = document\_type
      document\_format = document\_format
     IMPORTING document\_proxy = document
   \end{verbatim}
retcode = retcode.

9. Create the instance for data transfer (Link Server: method \texttt{get\_link\_server}; Table Collection: method \texttt{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]

\section*{Processing}

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

\section*{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):
   \begin{verbatim}
   CALL METHOD container->free.
   \end{verbatim}

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

\begin{itemize}
\item Remember that you should include error handling [Page 15] after each method call.
\end{itemize}

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 \texttt{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.

\begin{itemize}
\item Any parameters not contained in the individual method descriptions are listed under Generic Parameters [Page 144].
\end{itemize}
Generic Parameters

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

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue ' ': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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
## init_control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td></td>
<td>Name of the container in which you want to start the desktop office application.</td>
</tr>
<tr>
<td>r3_application_name</td>
<td></td>
<td>Heading</td>
</tr>
<tr>
<td>inplace_enabled</td>
<td>X</td>
<td>X: Allows a document to be opened in-place.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' ': No documents may be opened in-place.</td>
</tr>
<tr>
<td>inplace_show_toolbar</td>
<td>X</td>
<td>Show the toolbar of the desktop application (when inplace_enabled = 'X')</td>
</tr>
<tr>
<td>inplace_scroll_documents</td>
<td>X</td>
<td>Show the scrollbars of the desktop application (when inplace_enabled = 'X')</td>
</tr>
<tr>
<td>inplace_resize_documents</td>
<td>X</td>
<td>Resize the document display when the R/3 window is resized (when inplace_enabled = 'X')</td>
</tr>
<tr>
<td>autoalign</td>
<td>X</td>
<td>X: Fill the entire control container with the office application.</td>
</tr>
<tr>
<td>shell_style</td>
<td>X</td>
<td>The display format of the container control in the R/3 screen. Possible values are contained in the include &lt;SAPOLECONTAINERCONTROL&gt;. You do not normally have to change the default value of the parameter.</td>
</tr>
<tr>
<td>register_on_close_event</td>
<td>X</td>
<td>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 &quot;X&quot;.</td>
</tr>
<tr>
<td>register_on_custom_event</td>
<td>X</td>
<td>Registers additional events in the office application (for example, macro calls) Pass the value &quot;X&quot; to react to this event in your ABAP program.</td>
</tr>
<tr>
<td>rep_id</td>
<td>X</td>
<td>Obsolete. 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).</td>
</tr>
<tr>
<td>dynpro_nr</td>
<td>X</td>
<td>Obsolete. 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).</td>
</tr>
<tr>
<td>gui_container</td>
<td>X</td>
<td>Obsolete. Previous use: Name of the container defined in the Screen Painter for which the SAP Document Container Control should be created.</td>
</tr>
<tr>
<td>parent_id</td>
<td>X</td>
<td>Obsolete. 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.</td>
</tr>
</tbody>
</table>
**destroy_control**

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

April 2001
### 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.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rep_id</code></td>
<td>Name of program or module pool containing the screen on which you want to create the SAP Document Container Control.</td>
</tr>
<tr>
<td><code>dynpro_nr</code></td>
<td>Number of the screen on which you want to create the SAP Document Container Control.</td>
</tr>
<tr>
<td><code>gui_container</code></td>
<td>Name of the container defined in the Screen Painter for which the SAP Document Container Control should be created.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>X</td>
<td>Initial: spaces must be given in pixels Screen: spaces correspond to screen lines or screen columns</td>
</tr>
<tr>
<td>left</td>
<td>X</td>
<td>Sets the left margin in the office application window</td>
</tr>
<tr>
<td>top</td>
<td>X</td>
<td>Sets the top margin in the office application window</td>
</tr>
<tr>
<td>width</td>
<td>X</td>
<td>Sets the width of the office application window</td>
</tr>
<tr>
<td>height</td>
<td>X</td>
<td>Sets the height of the office application window</td>
</tr>
<tr>
<td>align</td>
<td>X</td>
<td>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 &lt;CTLDEF&gt;.</td>
</tr>
</tbody>
</table>

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
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

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

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:

<table>
<thead>
<tr>
<th>Word.Document.8:</th>
<th>Word97</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excel.Sheet.8:</td>
<td>Excel97 Worksheet</td>
</tr>
<tr>
<td>Excel.Chart.8:</td>
<td>Excel97 Chart</td>
</tr>
<tr>
<td>Word.Document</td>
<td>The current latest version of Word.</td>
</tr>
</tbody>
</table>
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:

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

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

```vba
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

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

You can call this method without previously having used the `init_control` 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>descr_list</td>
<td>X</td>
<td>Possible document types in an internal table with type soi_document_type_descr_list.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_type</td>
<td>X</td>
<td>Not used at present</td>
</tr>
<tr>
<td>link_server</td>
<td></td>
<td>Instance for the link server You define it with reference to the interface i_oi_link_server.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_collection</td>
<td>X</td>
<td>Generated instance for the table collection, created with reference to the interface i_oi_table_collection.</td>
</tr>
</tbody>
</table>

⚠️

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_type</td>
<td></td>
<td>The Program ID of the application as it appears in the Registry (such as 'Word.Document.8')</td>
</tr>
<tr>
<td>extensions</td>
<td></td>
<td>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 soi_colnames_table.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>control_2</td>
<td>X</td>
<td>Reference to the Desktop Office Integration control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data object must be created with reference to the class cl_gui_control.</td>
</tr>
</tbody>
</table>

Documents

Definition

You define this instance with reference to the interface i_o1_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].
Process flow

The following diagram shows the process of editing a document:

Declaration

1. Declare an instance with reference to interface i_o_i_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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>print_document</td>
<td>Prints the document</td>
</tr>
<tr>
<td>clear_selection</td>
<td>Deletes marked text</td>
</tr>
</tbody>
</table>
Working With Documents

### Methods for determining the attributes of a document

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

### Closing

7. 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.

8. 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

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_document [Page 35]</td>
<td></td>
</tr>
<tr>
<td>view_document [Page 36]</td>
<td></td>
</tr>
<tr>
<td>view_document_from_table [Page 37]</td>
<td></td>
</tr>
<tr>
<td>open_document_from_table [Page 39]</td>
<td></td>
</tr>
<tr>
<td>reopen_document [Page 40]</td>
<td></td>
</tr>
<tr>
<td>play_document_from_table [Page 43]</td>
<td></td>
</tr>
<tr>
<td>save_document_to_table [Page 44]</td>
<td></td>
</tr>
<tr>
<td>has_changed [Page 47]</td>
<td></td>
</tr>
<tr>
<td>is_open [Page 47]</td>
<td></td>
</tr>
<tr>
<td>has_standard_interface [Page 48]</td>
<td></td>
</tr>
<tr>
<td>has_wordprocessor_interface [Page 54]</td>
<td></td>
</tr>
<tr>
<td>get_wordprocessor_interface [Page 54]</td>
<td></td>
</tr>
<tr>
<td>get_document_type [Page 55]</td>
<td></td>
</tr>
<tr>
<td>get_document_handle [Page 55]</td>
<td></td>
</tr>
</tbody>
</table>

Methods of the standard interface:

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>save_as [Page 45]</td>
<td></td>
</tr>
<tr>
<td>save_copy_as [Page 45]</td>
<td></td>
</tr>
<tr>
<td>add_menu_bar [Page 49]</td>
<td></td>
</tr>
<tr>
<td>add_menu_popup [Page 49]</td>
<td></td>
</tr>
<tr>
<td>add_menu_item [Page 50]</td>
<td></td>
</tr>
<tr>
<td>delete_menu_item [Page 51]</td>
<td></td>
</tr>
<tr>
<td>select_all [Page 51]</td>
<td></td>
</tr>
<tr>
<td>clear_selection [Page 51]</td>
<td></td>
</tr>
<tr>
<td>cut_selection [Page 51]</td>
<td></td>
</tr>
<tr>
<td>copy_selection [Page 52]</td>
<td></td>
</tr>
<tr>
<td>paste_clipboard [Page 52]</td>
<td></td>
</tr>
<tr>
<td>paste_special [Page 52]</td>
<td></td>
</tr>
<tr>
<td>page_setup [Page 53]</td>
<td></td>
</tr>
<tr>
<td>print_document [Page 53]</td>
<td></td>
</tr>
<tr>
<td>update_document_links [Page 56]</td>
<td></td>
</tr>
<tr>
<td>get_application_property [Page 56]</td>
<td></td>
</tr>
<tr>
<td>execute_macro [Page 53]</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' ': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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.

```plaintext
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>open_inplace</td>
<td>X</td>
<td>'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method <code>get_document_proxy [Page 27]</code>). ' ': Document is opened in a separate window.</td>
</tr>
<tr>
<td>create_view_data</td>
<td>X</td>
<td>Set this option if you want to save presentation data with the document. This is necessary if you want to use the <code>view_document</code> method.</td>
</tr>
</tbody>
</table>
onsave_macro | X | The name of a macro that is called if the document has been changed and one of the following conditions applies:
- The user closes the application (if it is not running in-place)
- You use one of the following methods:
  - `save_document`
  - `reopen_document` with `do_save = 'X'`
  - `close_document` with `do_save = 'X'`

The Macro must be stored in the document.

Note: What constitutes a change to the document differs from application to application:
- 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.
- 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.
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**

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>document_table</code></td>
<td></td>
<td>Internal table containing the document.</td>
</tr>
<tr>
<td><code>document_size</code></td>
<td></td>
<td>File size.</td>
</tr>
<tr>
<td><code>document_title</code></td>
<td>X</td>
<td>Title to be displayed in the title bar of the desktop application.</td>
</tr>
</tbody>
</table>

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://`. 
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 get_document_proxy [Page 27]).
| ' ': 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_title</td>
<td>X</td>
<td>Title to be displayed in the title bar of the desktop application.</td>
</tr>
<tr>
<td>document_url</td>
<td></td>
<td>The URL address of the document.</td>
</tr>
</tbody>
</table>
| 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 get_document_proxy [Page 27]).
|                 |          | ' ': 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 has_changed [Page 47] is always initial). |
onsave_macro | X | The name of a macro that is called if the document has been changed and one of the following conditions applies:
- The user closes the application (if it is not running in-place)
- You use one of the following methods:
  - save_document
  - reopen_document with do_save = 'X'
  - close_document with do_save = 'X'

The Macro must be stored in the document.

**Note:** What constitutes a change to the document differs from application to application:
- 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
- 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.

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```

```java
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_table</td>
<td></td>
<td>Internal table containing the document.</td>
</tr>
<tr>
<td>document_size</td>
<td></td>
<td>File size.</td>
</tr>
<tr>
<td>document_title</td>
<td>X</td>
<td>Title to be displayed in the title bar of the desktop application.</td>
</tr>
<tr>
<td>open_inplace</td>
<td>X</td>
<td>‘X’: Document is opened within the R/3 window (only if the parameter inplace = ‘X’ was set when you called the method get_document_proxy [Page 27]). ‘ ’: Document is opened in a separate window.</td>
</tr>
<tr>
<td>open_readonly</td>
<td>X</td>
<td>Open in display mode (the close_event event is not triggered when the document is closed, and the parameter has_changed [Page 47] is always initial).</td>
</tr>
<tr>
<td>onsave_macro</td>
<td>X</td>
<td>The name of a macro that is called if the document has been changed and one of the following conditions applies:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The user closes the application (if it is not running in-place)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• You use one of the following methods:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- save_document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- reopen_document with do_save = 'X'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- close_document with do_save = 'X'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Macro must be stored in the document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> What constitutes a change to the document differs from application to application:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In MS Word and MS Powerpoint, you must physically change the document in order for the macro to be called</td>
</tr>
<tr>
<td>startup_macro</td>
<td>X</td>
<td>Macro to be run when you open the document.</td>
</tr>
<tr>
<td>protect_document</td>
<td>X</td>
<td>Flags whether to protect document against changes. It is still possible to select within the document.</td>
</tr>
</tbody>
</table>

⚠️

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>open_inplace</td>
<td>X</td>
<td>'X': Document is opened within the R/3 window (only if the parameter inplace = 'X' was set when you called the method get_document_proxy [Page 27]). ' ': Document is opened in a separate window.</td>
</tr>
<tr>
<td>open_readonly</td>
<td>X</td>
<td>Open in display mode (the close_event event is not triggered when the document is closed, and the parameter has_changed [Page 47] is always initial).</td>
</tr>
<tr>
<td>do_save</td>
<td>X</td>
<td>Determines whether the document should be saved</td>
</tr>
<tr>
<td>onsave_macro</td>
<td>X</td>
<td>The name of a macro that is called if the document has been changed and one of the following conditions applies:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The user closes the application (if it is not running in-place)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- You use one of the following methods:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- save_document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- reopen_document with do_save = 'X'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- close_document with do_save = 'X'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Macro must be stored in the document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: What constitutes a change to the document differs from application to application:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In MS Word and MS Powerpoint, you must physically change the document in order for the macro to be called</td>
</tr>
<tr>
<td>startup_macro</td>
<td>X</td>
<td>Macro to be run when you open the document.</td>
</tr>
<tr>
<td>has_changed</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

⚠️

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_type</td>
<td>X</td>
<td>Document type ProgId of the Visual Basic application</td>
</tr>
<tr>
<td>document_url</td>
<td>X</td>
<td>The URL address of the document</td>
</tr>
<tr>
<td>user_info</td>
<td>X</td>
<td>Logon and proxy data for accessing an HTTP or FTP server.</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_title</td>
<td>X</td>
<td>Title to be displayed in the title bar of the desktop application.</td>
</tr>
</tbody>
</table>
The URL address of the document

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

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**

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td></td>
<td>The URL address of the document to be saved</td>
</tr>
<tr>
<td>user_info</td>
<td>X</td>
<td>Logon and proxy data for accessing an HTTP or FTP server.</td>
</tr>
<tr>
<td>document_size</td>
<td>X</td>
<td>The length of the saved document is placed in this field.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_table</td>
<td></td>
<td>Internal table in which you want to save the document.</td>
</tr>
<tr>
<td>document_size</td>
<td></td>
<td>The length of the saved document is placed in this field.</td>
</tr>
</tbody>
</table>
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**

```plaintext
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>X</td>
<td>The name of the file to be created</td>
</tr>
<tr>
<td>prompt_user</td>
<td>X</td>
<td>When you execute the method, it starts the relevant dialog box in the office application.</td>
</tr>
</tbody>
</table>

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**

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>X</td>
<td>The name under which the office document is to be saved</td>
</tr>
</tbody>
</table>
close_document

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>do_save</td>
<td>X</td>
<td>Determines whether the document should be saved</td>
</tr>
<tr>
<td>has_changed</td>
<td>X</td>
<td>Flags whether the office application document was changed</td>
</tr>
</tbody>
</table>

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.

```plaintext
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**

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.

```plaintext
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:

```plaintext
call method document->has_changed
    exporting no_flush = no_flush
    importing error = error
    retcode = retcode
    ret_value = ret_value.
```

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ret_value</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

```plaintext
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:

```plaintext
call method document->is_open
    exporting no_flush = no_flush
    importing ret_value = ret_value
    error = error
    retcode = retcode.
```

**Description of parameters**
is_destroyed

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ret_value</td>
<td></td>
<td>Indicates whether the document is open</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ret_value</td>
<td></td>
<td>Flags whether the document has been destroyed</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_available</td>
<td></td>
<td>Flags whether a standard interface is available</td>
</tr>
</tbody>
</table>
add_menu_bar

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

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

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>The name of the new menu bar</td>
</tr>
<tr>
<td>pos</td>
<td>X</td>
<td>Not currently used</td>
</tr>
</tbody>
</table>

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.

```plaintext
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>The name of the new menu</td>
</tr>
</tbody>
</table>
**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.

```plaintext
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**

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

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**

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt_user</td>
<td>X</td>
<td>Flags whether the method should call the corresponding dialog box in the office application when it is executed.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prompt_user</td>
<td>X</td>
<td>Flags whether the method should call the corresponding dialog box in the office application when it is executed</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>macro_string</td>
<td></td>
<td>The name of the macro in the office application</td>
</tr>
<tr>
<td>script_name</td>
<td>X</td>
<td>The macro is part of the document &lt;name&gt;: The macro is placed in the script collection under this name.</td>
</tr>
<tr>
<td>param_count</td>
<td>X</td>
<td>Number of parameters to be passed to the macro</td>
</tr>
<tr>
<td>param1..param9</td>
<td></td>
<td>The parameters to be passed to the macro (not Word 97)</td>
</tr>
<tr>
<td>error_string</td>
<td>X</td>
<td>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.</td>
</tr>
<tr>
<td>retvalue</td>
<td>X</td>
<td>Value generated by the application macro.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_available</td>
<td></td>
<td>Flags whether a standard interface is available</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wp_interface</td>
<td></td>
<td>The instance created for the application-specific interface</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_type</td>
<td></td>
<td>Document type</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
The handle returned by the method points to the document object. The function modules CONTROL_CALL_METHOD, CONTROL_GET_PROPERTY, and CONTROL_SETPROPERTY 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property_name</td>
<td></td>
<td>Name of the property you want to retrieve</td>
</tr>
<tr>
<td>subproperty_name</td>
<td>X</td>
<td>Defines the property further</td>
</tr>
<tr>
<td>retvalue</td>
<td></td>
<td>Value of the property</td>
</tr>
</tbody>
</table>

Possible values:

<table>
<thead>
<tr>
<th>Property_Name</th>
<th>Subproperty_Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'TEMPLATE_PATH'</td>
<td>&quot;</td>
<td>The directory path for user-specific templates in a word processing program</td>
</tr>
<tr>
<td>'INTERNATIONAL'</td>
<td>'DECIMAL_SEPARATOR'</td>
<td>Character for decimal separator</td>
</tr>
<tr>
<td>'INTERNATIONAL'</td>
<td>'THOUSANDS_SEPARATOR'</td>
<td>Character for thousands separator</td>
</tr>
<tr>
<td>'INTERNATIONAL'</td>
<td>'LIST_SEPARATOR'</td>
<td>Separator character for lists</td>
</tr>
<tr>
<td>'INTERNATIONAL'</td>
<td>'DATE_SEPARATOR'</td>
<td>Separator character for dates</td>
</tr>
<tr>
<td>'INTERNATIONAL'</td>
<td>'TIME_SEPARATOR'</td>
<td>Separator character for times</td>
</tr>
</tbody>
</table>
**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 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_available</td>
<td></td>
<td>Flags whether a tables interface is available</td>
</tr>
</tbody>
</table>

**get_spreadsheet_interface**

Creates an instance sheet_interface for managing the tables interface.

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| sheet_interface |          | Name of the generated instance for the tables interface. Object with reference to interface i_oि_spreadsheet.
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_available</td>
<td></td>
<td>Flags whether a form interface is available</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_interface</td>
<td></td>
<td>The instance created for the form interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object with reference to interface i_oi_form</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
**get_mail_merge_interface**

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

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm_interface</td>
<td></td>
<td>Name of the generated instance for the mail merge interface. Object with reference to interface <code>i_oi_mail_merge</code>.</td>
</tr>
</tbody>
</table>

**has_pivot_interface**

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

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_available</td>
<td></td>
<td>Flag to indicate whether the pivot interface is available</td>
</tr>
</tbody>
</table>

**get_pivot_interface**

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

```pli
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

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

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
                   = retcode.

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scripts</td>
<td></td>
<td>Name of the generated instance for the script collection. Object with reference to interface i_oj_script_collection.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document_cookie</td>
<td></td>
<td>Document ID</td>
</tr>
</tbody>
</table>
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](Page 22), 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 (...).
   ENDCCLASS.

2. Implement the static methods:

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

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.
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 (...).
   ENDCCLASS.

2. Implement the methods:

   CLASS event_handler IMPLEMENTATION.
   METHOD close_event.
   <Processing logic>
   ENDMETHOD.

   METHOD custom_event.
   <Processing logic>
   ENDMETHOD.
   ENDCCLASS.

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_oi_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_oilink_server:
   data: link_server type ref to i_oilink_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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue ' ': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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:

```plaintext
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.
```
stop_link_server

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link_server_mode</td>
<td>X</td>
<td>link_server-&gt;link_server_inactive: The link server instance is created but not registered (see note). link_server-&gt;link_server_standardname: The link server is created with the standard identifier link_server-&gt;link_server_customname: The link server is created with the additional identifier server_name_suffix.</td>
</tr>
<tr>
<td>server_name_suffix</td>
<td>X</td>
<td>Suffix to name of link server</td>
</tr>
</tbody>
</table>

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_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

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

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

```plaintext
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:

```plaintext
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.

```plaintext
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_name</td>
<td></td>
<td>The name of the internal table in the ABAP program</td>
</tr>
<tr>
<td>item_title</td>
<td>X</td>
<td>Describes the table</td>
</tr>
<tr>
<td>table_url</td>
<td></td>
<td>The location of an internal table when it has been downloaded from</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the presentation server (location created for example using the function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>module ‘DP_CREATE_URL_FROM_TABLE’)</td>
</tr>
<tr>
<td>column_count</td>
<td>X</td>
<td>Number of columns</td>
</tr>
<tr>
<td>line_count</td>
<td>X</td>
<td>Number of lines</td>
</tr>
</tbody>
</table>

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

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

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_name</td>
<td></td>
<td>Technical name of the object to be deleted</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_name</td>
<td></td>
<td>Technical name of the object to which a link is to be placed on the clipboard</td>
</tr>
</tbody>
</table>

💡

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.
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 <GUID>" "<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_o_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:

```
Container
   Tables
      Name
      Type
      Data
      Table
         Name
         Data
         Columns
         Rows
         Ranges
```

**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

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_o_i_table_collection:
   ```
   DATA table_coll TYPE REF TO i_o_i_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
        Next j
    Next i
    Call ThisWorkbook .Container.SendCustomEvent("ON_CHANGE")
End Sub
Generic Parameters

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

### Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>‘X’: Synchronizes the automation queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘ ’: Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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.

```abap
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td></td>
<td>The name of the internal table in the ABAP program</td>
</tr>
</tbody>
</table>
### add_table

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

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:

```plaintext
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:

```plaintext
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td></td>
<td>The name of the internal table in the ABAP program</td>
</tr>
<tr>
<td>table_title</td>
<td>X</td>
<td>Describes the table</td>
</tr>
</tbody>
</table>
| table_type | | Table type:  
- `table_coll->table_type_input`: Passes an empty table to the desktop application. This is used to load data from the application into the R/3 System.  
- `table_coll->table_type_output`: 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**

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`.

```plaintext
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td></td>
<td>The name of the internal table in the ABAP program</td>
</tr>
<tr>
<td>table_url</td>
<td></td>
<td>Memory address of the internal table</td>
</tr>
</tbody>
</table>

---

**get_table_by_url**

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

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td></td>
<td>The name of the internal table in the ABAP program</td>
</tr>
<tr>
<td>description</td>
<td>X</td>
<td>Description of the table</td>
</tr>
<tr>
<td>date</td>
<td></td>
<td>Date of last change</td>
</tr>
<tr>
<td>time</td>
<td></td>
<td>Time of last change</td>
</tr>
<tr>
<td>ddic_name</td>
<td>X</td>
<td>Link to the ABAP Dictionary</td>
</tr>
<tr>
<td>data_table</td>
<td></td>
<td>Table containing data</td>
</tr>
<tr>
<td>fields_table</td>
<td>X</td>
<td>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>.</td>
</tr>
<tr>
<td>properties</td>
<td>X</td>
<td>Additional properties. Create the table with reference to the type <code>SOI_PROPERTIES_TABLE</code>.</td>
</tr>
</tbody>
</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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_name</td>
<td></td>
<td>Name of the internal table in the table collection</td>
</tr>
</tbody>
</table>

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 _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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue ' ': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td></td>
<td>The version of the word processor application</td>
</tr>
</tbody>
</table>

install_template

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td></td>
<td>The version of the word processor application</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template_name</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>attach_styles</td>
<td></td>
<td>'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).</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template_info</td>
<td></td>
<td>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 SOI_TEMPLATE_INFO.</td>
</tr>
</tbody>
</table>

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
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
IMPORTING
  retcode = retcode
  error = error.

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search_string</td>
<td></td>
<td>Searches for a character string</td>
</tr>
<tr>
<td>pos</td>
<td></td>
<td>global: searches the entire text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>up: searches from the current position to the start of the document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>down: searches from the current position to the end of the document</td>
</tr>
<tr>
<td>flag</td>
<td></td>
<td><code>:</code> searches for character string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>word: searched for word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>exact: case-sensitive search</td>
</tr>
</tbody>
</table>

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
IMPORTING
  retcode = retcode
  error = error.
insert_table

IMPORTING retcode = retcode	error = error.

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search_string</td>
<td></td>
<td>Searches for a character string</td>
</tr>
<tr>
<td>replace_string</td>
<td></td>
<td>The character string that replaces the search string</td>
</tr>
<tr>
<td>pos</td>
<td></td>
<td>global: searches the entire text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>up: searches from the current position to the start of the document</td>
</tr>
<tr>
<td></td>
<td></td>
<td>down: searches from the current position to the end of the document</td>
</tr>
<tr>
<td>flag</td>
<td></td>
<td>‘ ‘: searches for character string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>word: searched for word</td>
</tr>
<tr>
<td></td>
<td></td>
<td>exact: case-sensitive search</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_table</td>
<td></td>
<td>The internal table containing the R/3 data you want to place in the table in the document</td>
</tr>
<tr>
<td>info_table</td>
<td></td>
<td>An internal table with the type SOI_COLS_TABLE. 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.</td>
</tr>
<tr>
<td>lowerbound</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>upperbound</td>
<td></td>
<td>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 lowerbound up to the end of the table, use the value 0.</td>
</tr>
<tr>
<td>doctable_number</td>
<td></td>
<td>The number of the table in the document into which you want to insert the data. The numbering starts at 1.</td>
</tr>
<tr>
<td>clearoption</td>
<td></td>
<td>Option for clearing existing contents of the table in the document. It can have the following values:</td>
</tr>
</tbody>
</table>
|                |          | - 0 - No clearing  
  The method only overwrites fields of the table in which it actually enters data.  
|                |          | - 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 (startrow) is left untouched. The contents of remaining lines of the old table that are below the insertion are deleted.  
  **Note:** 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 info_table) are not deleted.  
|                |          | - 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. |
| startrow       |          | The first row of the document table in which you want to insert data |
insert_table2

| varsizes | Flag to indicate whether the table in the document should be resized to fit the number of rows passed by the method. Possible values:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>' '</td>
<td>The table is not resized. <strong>Caution</strong>: If the table has fewer rows than the number of entries you pass to it, not all of the data can be displayed.</td>
</tr>
<tr>
<td>'X'</td>
<td>The table is resized.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>whotetable</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not yet implemented</td>
<td></td>
</tr>
</tbody>
</table>

While you can use `varsizes` 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.
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
  varsizes = varsizes
  no_flush = no_flush
IMPORTING
  retcode = retcode
  error = error.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_table</td>
<td></td>
<td>The internal table containing the R/3 data you want to place in the table in the document.</td>
</tr>
<tr>
<td>info_col_table</td>
<td></td>
<td>An internal table with the type SOI_COLNAMES_TABLE. Each row in this table can contain the name of a column in the data table. 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.</td>
</tr>
<tr>
<td>lowerbound</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>upperbound</td>
<td></td>
<td>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 lowerbound up to the end of the table, use the value 0.</td>
</tr>
<tr>
<td>doctable_number</td>
<td></td>
<td>The number of the table in the document into which you want to insert the data. The numbering starts at 1.</td>
</tr>
<tr>
<td>clearoption</td>
<td></td>
<td>Option for clearing existing contents of the table in the document. It can have the following values:</td>
</tr>
</tbody>
</table>
|                    |          | 0 - No clearing
|                    |          | The method only overwrites fields of the table in which it actually enters data.                                                           |
|                    |          | 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 (startrow) is left untouched. The contents of the remaining lines of the old table that are below the insertion are deleted. **Note:** 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 info_table) are not deleted. |
|                    |          | 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. |
get_table_info

<table>
<thead>
<tr>
<th>startrow</th>
<th>The first row of the document table in which you want to insert data</th>
</tr>
</thead>
<tbody>
<tr>
<td>varsizesize</td>
<td>Flag to indicate whether the table in the document should be resized to fit the number of rows passed by the method. Possible values:</td>
</tr>
<tr>
<td></td>
<td>' ' : The table is not resized.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution</strong>: If the table has fewer rows than the number of entries you pass to it, not all of the data can be displayed.</td>
</tr>
<tr>
<td></td>
<td>'X' : The table is resized.</td>
</tr>
<tr>
<td>wholetable</td>
<td>X</td>
</tr>
</tbody>
</table>

While you can use **varsizesize** 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.

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

The Form Interface

Definition

You define this instance with reference to the interface i_oi_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:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Type of field in document</td>
<td>'C': Checkbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'T': Text field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'L': List box</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'P': Property: Field should only be displayed.</td>
</tr>
<tr>
<td>name</td>
<td>Name of field in document Data is always passed between pairs of identically-named fields.</td>
<td></td>
</tr>
<tr>
<td>number</td>
<td>This field is only relevant for list boxes. If a list box contains more than one identical entry, this field distinguishes them.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>This field contains the current value of the field. For checkboxes, the Boolean representation is simulated using 0 and 1.</td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>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’.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Only for list boxes: The entry is selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: Only for list boxes: All entries in the list box are deleted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3: All types: The field contents cannot be changed.</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue ' ': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td>Error handling [Page 15]</td>
<td></td>
</tr>
<tr>
<td>error</td>
<td>Error handling [Page 15]</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

#### Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td></td>
<td>Table with the structure soi_form_list. These entries are used to change the fields in the form.</td>
</tr>
</tbody>
</table>
Before the fields from the **fields** 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 **fields** 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:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Number</th>
<th>Value</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Single</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>Name</td>
<td>1</td>
<td>A. N. Other</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>Department</td>
<td>1</td>
<td>ABAP</td>
<td>3</td>
</tr>
</tbody>
</table>

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:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Number</th>
<th>Value</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Colors</td>
<td>1</td>
<td>Red</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>Colors</td>
<td>2</td>
<td>Green</td>
<td>1</td>
</tr>
<tr>
<td>L</td>
<td>Colors</td>
<td>3</td>
<td>Blue</td>
<td>0</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>X</td>
<td>'X': Transfers all values from the form, including non-selected entries from list boxes.</td>
</tr>
</tbody>
</table>
add_field

| fields | Table with the structure soi_form_list. The contents of this table reflect those of the form. |

The \texttt{value} column of the \texttt{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 \texttt{code} column.

add_field

Use this method to add a new form field.

\begin{verbatim}
CALL METHOD form->add_field
  EXPORTING field    = field
                    no_flush = no_flush
  IMPORTING retcode  = retcode
                    error    = error.
\end{verbatim}

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td></td>
<td>Structure with the type \texttt{soi_form_item}. The field is displayed in the form according to the entries in the structure.</td>
</tr>
</tbody>
</table>

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.

\begin{verbatim}
CALL METHOD form->delete_fields
  EXPORTING fields   = fields
                    no_flush = no_flush
  IMPORTING retcode  = retcode
                    error    = error.
\end{verbatim}

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td></td>
<td>Table with the structure \texttt{soi_form_list}. The specified fields are deleted from the form.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td></td>
<td>Table with the structure soi_form_list. The links between the specified fields in the document and the R/3 System are disconnected.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td></td>
<td>Table with the structure soi_form_list. It contains all fields from the form.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protect</td>
<td>X</td>
<td>'X': The form (but not its fields) is protected against changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'.' : Any part of the form can be changed.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>modus</td>
<td>X</td>
<td>'X': Display in display mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'.' : Display in design mode.</td>
</tr>
</tbody>
</table>

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_o_i_mail_merge`:
   ```
   DATA mail_merge TYPE REF TO i_o_i_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:

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

   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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue ' ': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddic_name</td>
<td>X</td>
<td>Name of the table or structure in the ABAP Dictionary.</td>
</tr>
<tr>
<td>description</td>
<td>X</td>
<td>Description of the table</td>
</tr>
<tr>
<td>date</td>
<td>X</td>
<td>Date of last change</td>
</tr>
<tr>
<td>time</td>
<td>X</td>
<td>Time of last change</td>
</tr>
<tr>
<td>data_table</td>
<td></td>
<td>Data table</td>
</tr>
</tbody>
</table>
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

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td></td>
<td>Index of the data record.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td></td>
<td>The first data record you want to merge</td>
</tr>
<tr>
<td>last</td>
<td></td>
<td>The last data record you want to merge</td>
</tr>
</tbody>
</table>

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
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewcodes</td>
<td>'X'</td>
<td>Do not display field functions</td>
</tr>
<tr>
<td></td>
<td>' '</td>
<td>Display field functions</td>
</tr>
</tbody>
</table>

The Script Collection

Definition

You define this instance with reference to the interface i_oi_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

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_o_i_script_collection:
   ```
   DATA script_coll TYPE REF TO i_o_i_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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_script</td>
<td>Adds a script to the collection</td>
</tr>
<tr>
<td>add_script_from_table</td>
<td>Adds a script to the collection</td>
</tr>
<tr>
<td>remove_all_scripts</td>
<td>Remove all scripts</td>
</tr>
<tr>
<td>remove_script</td>
<td>Remove a particular script</td>
</tr>
</tbody>
</table>

   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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>‘X’: Synchronizes the automation queue '&lt; Automation queue [Ext.] &gt; is synchronized before leaving the method.</td>
</tr>
</tbody>
</table>
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 `add_script` method is used to add a script to the collection. The script is identified by a URL.

```plaintext
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

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

The `add_script_from_table` method is used to load a script. The script must be contained in a table in the application program.

```plaintext
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

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

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_flux     = no_flux
   IMPORTING error        = error
               retcode     = retcode.

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script_name</td>
<td></td>
<td>Name of the script you want to remove.</td>
</tr>
</tbody>
</table>
The Spreadsheet Interface

Definition
You define this instance with reference to the interface i_oil_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_oil_spreadsheet.
   
   DATA spreadsheet TYPE REF TO i_oil_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

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

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no_flush</td>
<td>X</td>
<td>'X': Synchronizes the automation queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>': Automation queue [Ext.] is synchronized before leaving the method.</td>
</tr>
<tr>
<td>retcode</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>Error handling [Page 15]</td>
</tr>
</tbody>
</table>

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 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>(default value) The existing update setting remains unchanged</td>
</tr>
<tr>
<td>0</td>
<td>The screen update is switched off before the method is called, and remains switched off</td>
</tr>
<tr>
<td>1</td>
<td>The screen update is switched on after the method has been executed, and remains switched on</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheetname</td>
<td></td>
<td>Name of current worksheet</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of the range</td>
</tr>
<tr>
<td>top</td>
<td></td>
<td>Top edge of the range</td>
</tr>
<tr>
<td>left</td>
<td></td>
<td>Left-hand edge of the range</td>
</tr>
<tr>
<td>rows</td>
<td></td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td></td>
<td>Number of columns</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheetname</td>
<td></td>
<td>Name of worksheet</td>
</tr>
<tr>
<td>protect</td>
<td></td>
<td>‘X’: Worksheet is protected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘ ’: Worksheet is not protected</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheets</td>
<td></td>
<td>List of worksheets. The parameter must be created with reference to the type SOI_SHEETS_TABLE.</td>
</tr>
<tr>
<td>updating</td>
<td>X</td>
<td>See The Updating Parameter [Page 109]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>X</td>
<td>'X': Returns the data in all ranges ' ': Returns only the data from the ranges specified in the ranges table.</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updating</td>
<td>X</td>
<td>See The Updating Parameter [Page 109]</td>
</tr>
</tbody>
</table>
| ranges    |          | Table with all ranges
          |          | The table must be created with reference to the type
          |          | SOI_RANGE_LIST (see the Important Table Structures [Page 136] section). |
**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.

```java
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td></td>
<td>Line number</td>
</tr>
<tr>
<td>column</td>
<td></td>
<td>Column number</td>
</tr>
<tr>
<td>format</td>
<td></td>
<td>Format of the cell. Structure with type SOI_FORMAT_ITEM (see also Important Table Structures [Page 136]).</td>
</tr>
</tbody>
</table>

**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.

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td></td>
<td>Internal table with the data type SOI_CELL_TABLE (see Important Table Structures [Page 136]). You fill the table with the coordinates of the relevant cells before the method call. The method fills the remaining fields with formatting information.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updating</td>
<td>X</td>
<td>See The Updating Parameter [Page 109]</td>
</tr>
<tr>
<td>top</td>
<td></td>
<td>Top edge of the range</td>
</tr>
<tr>
<td>left</td>
<td></td>
<td>Left-hand edge of the range</td>
</tr>
<tr>
<td>rows</td>
<td></td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td></td>
<td>Number of columns</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>areas</td>
<td></td>
<td>An internal table with the type soi_area_table. For further information, refer to the Important Table Structures [Page 136] section.</td>
</tr>
</tbody>
</table>
select_range

**select_range**

Selects an existing named range.

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td></td>
<td>Name of the range you want to select</td>
</tr>
</tbody>
</table>

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.

```plaintext
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td></td>
<td>Top edge of the range</td>
</tr>
<tr>
<td>left</td>
<td></td>
<td>Left-hand edge of the range</td>
</tr>
<tr>
<td>rows</td>
<td></td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td></td>
<td>Number of columns</td>
</tr>
<tr>
<td>updating</td>
<td>X</td>
<td>See [The Updating Parameter][Page 109]</td>
</tr>
</tbody>
</table>

[Page 109]
**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.

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

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of worksheet</td>
</tr>
</tbody>
</table>

**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.

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

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td></td>
<td>Area name</td>
</tr>
<tr>
<td>front</td>
<td></td>
<td>Font color (see color palette) ' -1 ' : Color remains unchanged</td>
</tr>
<tr>
<td>back</td>
<td></td>
<td>Background color (see color palette) ' -1 ' : Color remains unchanged</td>
</tr>
</tbody>
</table>

The color palette contains the following colors:
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td></td>
<td>Area name</td>
</tr>
<tr>
<td>family</td>
<td></td>
<td>Font. The following fonts are currently available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Arial'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Courier New'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Times New Roman'</td>
</tr>
<tr>
<td>size</td>
<td></td>
<td>Font size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td>bold</td>
<td></td>
<td>'1': Bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'0': Normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td>Area name</td>
<td></td>
</tr>
<tr>
<td>typ</td>
<td></td>
<td>Type for currency formatting:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0: Display as text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1: Display as a simple number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 2: Scientific display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 3: Display as a percentage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- -1: The system determines the display type by analyzing the currency parameter.</td>
</tr>
<tr>
<td>currency</td>
<td></td>
<td>ISO standard currency code</td>
</tr>
<tr>
<td>decimals</td>
<td>X</td>
<td>The number of decimal places</td>
</tr>
</tbody>
</table>
Desktop Office Integration (BC-CI)

**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` 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.

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td></td>
<td>Area name</td>
</tr>
<tr>
<td>formatstring</td>
<td></td>
<td>Format string to be passed to Microsoft Excel</td>
</tr>
</tbody>
</table>

**set_frame**

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

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td></td>
<td>Area name</td>
</tr>
</tbody>
</table>
**set_ranges_format**

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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sets the left margin</td>
</tr>
<tr>
<td>1</td>
<td>Sets the top margin</td>
</tr>
<tr>
<td>2</td>
<td>Sets the bottom margin</td>
</tr>
<tr>
<td>3</td>
<td>Sets the right margin</td>
</tr>
<tr>
<td>4</td>
<td>Horizontal line</td>
</tr>
<tr>
<td>5</td>
<td>Sets the left margin</td>
</tr>
<tr>
<td>6</td>
<td>Thickness</td>
</tr>
<tr>
<td>7</td>
<td>Thickness</td>
</tr>
</tbody>
</table>

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

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formattable</td>
<td></td>
<td>Control table with type <code>SOI_FORMAT_TABLE</code> (see also [Important Table Structures](Page 136)).</td>
</tr>
<tr>
<td>updating</td>
<td>X</td>
<td>See [The Updating Parameter](Page 109)</td>
</tr>
</tbody>
</table>

**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` 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cells</td>
<td></td>
<td>An internal table with the type SOI_CELL_TABLE. For further information, refer to the Important Table Structures [Page 136] section.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protect</td>
<td></td>
<td>‘X’: Worksheet is protected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘ ’: Worksheet is not protected</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>A type C field containing the name of the new worksheet.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oldname</td>
<td></td>
<td>A type C field containing the current name of the worksheet you want to rename</td>
</tr>
<tr>
<td>newname</td>
<td></td>
<td>A type C field containing the new name of the worksheet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>A type C field containing the range you want to protect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protect</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'X'</td>
<td>Range is protected</td>
</tr>
<tr>
<td>' '</td>
<td>Range is not protected</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>A type C field containing the name of the worksheet you want to delete</td>
</tr>
</tbody>
</table>

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
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rangename</td>
<td></td>
<td>Range name</td>
</tr>
<tr>
<td>newname</td>
<td>X</td>
<td>New name of the range</td>
</tr>
<tr>
<td>rows</td>
<td>X</td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td>X</td>
<td>Number of columns</td>
</tr>
<tr>
<td>updating</td>
<td>X</td>
<td>See The Updating Parameter [Page 109]</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>The name of the new range</td>
</tr>
<tr>
<td>rows</td>
<td></td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td></td>
<td>Number of columns</td>
</tr>
<tr>
<td>updating</td>
<td>X</td>
<td>See The Updating Parameter [Page 109]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ranges</td>
<td></td>
<td>An internal table with the type SOI_FULL_RANGE_TABLE. Each row contains the description of one range. For further information, refer to the Important Table Structures [Page 136] section.</td>
</tr>
</tbody>
</table>
**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.

```abap
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddic_name</td>
<td>✗</td>
<td>Name of the ABAP Dictionary structure</td>
</tr>
<tr>
<td>data_table</td>
<td></td>
<td>Data table</td>
</tr>
<tr>
<td>fields_table</td>
<td></td>
<td>Description of the table structure (if you have not already passed it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>using the ddic_name parameter).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Create the table with reference to the type SOI_FIELDS_TABLE.</td>
</tr>
<tr>
<td>rangename</td>
<td></td>
<td>Area name</td>
</tr>
<tr>
<td>wholetable</td>
<td>✗</td>
<td>Inserts the entire table, regardless of the size of the range. (See also</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPREADSHEET-&gt;SPREADSHEET_INSERTALL in soi_range_list [Page 136]).</td>
</tr>
<tr>
<td>updating</td>
<td>✗</td>
<td>See The Updating Parameter [Page 109]</td>
</tr>
</tbody>
</table>

**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.

```abap
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| ranges      |          | Table with all ranges
  The table must be created with reference to the type S0I_RANGE_LIST (see the Important Table Structures [Page 136] section). |
| contents    |          | Data table
  The table must be created with reference to the type S0I_GENERIC_TABLE (see the Important Table Structures [Page 136] section). |
| rangesdef   | X        | An internal table with the type S0I_DIMENSION_TABLE with which you can define ranges by their starting position and size instead of their name.
  Note: If you use this parameter, the ranges table (see above) must be empty. |
| updating    | X        | See The Updating Parameter [Page 109] |

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>The range to be deleted.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ranges</td>
<td></td>
<td>Table containing the list of ranges. The table must be created with reference to the type SOI_RANGE_LIST. However, in each entry, you only need to fill the name field.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of the range</td>
</tr>
<tr>
<td>first</td>
<td>X</td>
<td>Number of the first column in the range you want to hide</td>
</tr>
<tr>
<td>last</td>
<td>X</td>
<td>Number of the last column in the range you want to hide</td>
</tr>
</tbody>
</table>

*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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of the range</td>
</tr>
</tbody>
</table>

### 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Range to be hidden</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Range to be displayed</td>
</tr>
</tbody>
</table>

**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].`

```plaintext
CALL METHOD spreadsheet->set_hierarchy
  EXPORTING firstline = firstline
  length    = length
  IMPORTING error = error
              retcode = retcode.
```

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstline</td>
<td></td>
<td>The first line of the hierarchy area (type I)</td>
</tr>
<tr>
<td>length</td>
<td></td>
<td>The length of the hierarchy area in lines (type I)</td>
</tr>
</tbody>
</table>
**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.

```plaintext
CALL METHOD spreadsheet->set_hierarchy_table
  EXPORTING table    = table
  no_flush = no_flush
  IMPORTING error    = error
  retcode  = retcode.
```

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td></td>
<td>An internal table with the type soi_hierarchy_table. For further information, refer to the Important Table Structures [Page 136] section.</td>
</tr>
</tbody>
</table>

**clear_hierarchy**

Use this method to clear all of the hierarchy levels you have set in the worksheet.

```plaintext
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.

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of the worksheet you want to print</td>
</tr>
</tbody>
</table>
**set_zoom**

Use the `set_zoom` method to change the zoom of the current worksheet.

```callmethod
CALL METHOD spreadsheet->set_zoom
   EXPORTING zoom     = zoom
   no_flush = no_flush
   IMPORTING error    = error
   retcode  = retcode.
```

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoom</td>
<td></td>
<td>Zoom factor for the current worksheet. The values are interpreted as percentages.</td>
</tr>
</tbody>
</table>

**fit_widest**

This method adjusts the width the cells in an area to the width of the widest cell contents.

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

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of the range you want to change</td>
</tr>
</tbody>
</table>

If you do not specify a range name, the function applies to the entire worksheet.

If you called the method `set_ranges_data` 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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updating</td>
<td></td>
<td>Flag to specify whether the screen display should be updated. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' ' : Do not update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'X' : Update</td>
</tr>
</tbody>
</table>

**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.

```plaintext
CALL METHOD spreadsheet->load_lib
   EXPORTING no_flush = no_flush
   IMPORTING error    = errorversion
                    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.

```plaintext
CALL METHOD spreadsheet->version
   EXPORTING no_flush = no_flush
   IMPORTING version  = version
                    error    = errorversion
                    retcode  = retcode.
```

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td></td>
<td>A type C field containing the version of the spreadsheet interface</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
### Important Table Structures

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>C(1)</td>
<td>Name of the range</td>
</tr>
<tr>
<td>rows</td>
<td>C(1)</td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td>C(1)</td>
<td>Number of columns</td>
</tr>
<tr>
<td>code</td>
<td>C(1)</td>
<td>Function in the range:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_CLEAR: Deletes range</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_COLUMNSHIDE: Hides columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_ROWSHIDE: Hides rows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_PROTECT: Range is protected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_UNPROTECT: Range is not protected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_COLUMNSSHOW: Columns are displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_ROWSSHOW: Rows are displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_INSERTALL: The entire table is inserted, regardless of the size of the area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SPREADSHEET-&gt;SPREADSHEET_NEWRANGE: Creates a new range</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>C(4)</td>
<td>Row</td>
</tr>
<tr>
<td>column</td>
<td>C(4)</td>
<td>Column</td>
</tr>
<tr>
<td>value</td>
<td>C(256)</td>
<td>Value</td>
</tr>
</tbody>
</table>

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`

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>C(256)</td>
<td>Name of the range</td>
</tr>
<tr>
<td>front</td>
<td>I</td>
<td>Font color (see color palette)</td>
</tr>
<tr>
<td>back</td>
<td>I</td>
<td>Background color (see color palette)</td>
</tr>
<tr>
<td>font</td>
<td>C(256)</td>
<td>Name of the font family. The following values are permitted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Arial'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Courier New'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Times New Roman'</td>
</tr>
<tr>
<td>size</td>
<td>I</td>
<td>Font size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td>bold</td>
<td>I</td>
<td>'1': Bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'0': Normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td>italic</td>
<td>I</td>
<td>'1': Italic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'0': Normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td>align</td>
<td>I</td>
<td>Alignment:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'0': Right-justified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'1': Centered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'2': Left-justified</td>
</tr>
<tr>
<td>frametype</td>
<td>I</td>
<td>Control byte for setting the frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td>framecolor</td>
<td>I</td>
<td>Frame color (see color palette)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'-1': Unchanged</td>
</tr>
<tr>
<td>currency</td>
<td>C(3)</td>
<td>ISO standard currency code</td>
</tr>
</tbody>
</table>
number | 1 | Specifies the format of a cell in a range.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• 1: Display as a simple number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: Scientific display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3: Display as a percentage</td>
</tr>
</tbody>
</table>

The following colors are permitted:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sets the left margin</td>
</tr>
<tr>
<td>1</td>
<td>Sets the top margin</td>
</tr>
<tr>
<td>2</td>
<td>Sets the bottom margin</td>
</tr>
<tr>
<td>3</td>
<td>Sets the right margin</td>
</tr>
<tr>
<td>4</td>
<td>Horizontal line</td>
</tr>
<tr>
<td>5</td>
<td>Sets the left margin</td>
</tr>
<tr>
<td>6</td>
<td>Thickness</td>
</tr>
<tr>
<td>7</td>
<td>Thickness</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>C(128)</td>
<td>Name of the range</td>
</tr>
</tbody>
</table>
Important Table Structures

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>I</td>
<td>Top row of the range</td>
</tr>
<tr>
<td>left</td>
<td>I</td>
<td>Leftmost column of the range</td>
</tr>
<tr>
<td>rows</td>
<td>I</td>
<td>Number of rows in the range</td>
</tr>
<tr>
<td>columns</td>
<td>I</td>
<td>Number of columns in the range</td>
</tr>
<tr>
<td>sheets</td>
<td>C(128)</td>
<td>Worksheet on which the range is defined</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>I</td>
<td>Top row of the range</td>
</tr>
<tr>
<td>left</td>
<td>I</td>
<td>Leftmost column of the range</td>
</tr>
<tr>
<td>rows</td>
<td>I</td>
<td>Number of rows in the range</td>
</tr>
<tr>
<td>columns</td>
<td>I</td>
<td>Number of columns in the range</td>
</tr>
<tr>
<td>front</td>
<td>I</td>
<td>Font color (see color palette)</td>
</tr>
<tr>
<td>back</td>
<td>I</td>
<td>Background color (see color palette)</td>
</tr>
<tr>
<td>font</td>
<td>C(256)</td>
<td>Font. The following are permitted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Arial</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Courier New</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Times Roman</td>
</tr>
<tr>
<td>size</td>
<td>I</td>
<td>Font size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use −1 if the font size is to remain unchanged.</td>
</tr>
<tr>
<td>bold</td>
<td>I</td>
<td>• ‘1’: Bold</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ‘0’: Normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ‘−1’: Unchanged</td>
</tr>
<tr>
<td>italic</td>
<td>I</td>
<td>• ‘1’: Italic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ‘0’: Normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ‘−1’: Unchanged</td>
</tr>
</tbody>
</table>
### Important Table Structures

<table>
<thead>
<tr>
<th>align</th>
<th>I</th>
<th>Alignment:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• ' -1 ': Unchanged</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ' 0 ': Right-justified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ' 1 ': Centered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ' 2 ': Left-justified</td>
</tr>
</tbody>
</table>

| 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. |
|         |     | • 1: Display as a simple number |
|         |     | • 2: Scientific display |
|         |     | • 3: Display as a percentage |

| decimals | I | Number of decimal places |
| input | I | • ' 0 ': Input off |
|        |   | • ' 1 ': Input on |

The following colors are permitted:

---

April 2001 141
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

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>I</td>
<td>Topmost row of the range</td>
</tr>
<tr>
<td>left</td>
<td>I</td>
<td>Leftmost column of the range</td>
</tr>
<tr>
<td>rows</td>
<td>I</td>
<td>Number of rows</td>
</tr>
<tr>
<td>columns</td>
<td>I</td>
<td>Number of columns</td>
</tr>
</tbody>
</table>

The Pivot Interface

Definition

You define this instance with reference to the interface i_oi_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

<table>
<thead>
<tr>
<th>Method</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_source_table</td>
<td>Inserts the contents of an R/3 internal table into a pivot table</td>
</tr>
<tr>
<td>set_fieldtype</td>
<td>Reassigns the type of an existing field in the pivot table</td>
</tr>
<tr>
<td>get_allfields</td>
<td>Returns a list of fields in a specified pivot table</td>
</tr>
<tr>
<td>get_allpivottables</td>
<td>Returns a list of the names of all pivot tables in a workbook</td>
</tr>
<tr>
<td>drill_down</td>
<td>Expands the data display for a particular value to display more detail</td>
</tr>
<tr>
<td>drill_up</td>
<td>Collapses the data display for a particular value to display only a summary</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
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.

```vba
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddic_name</td>
<td>X</td>
<td>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. <strong>Note:</strong> If your internal table does not have the structure of an ABAP Dictionary table, you must use the <code>fields_table</code> parameter instead.</td>
</tr>
<tr>
<td>data_table</td>
<td></td>
<td>The name of the internal table in your ABAP program containing the data that you want to display in the pivot table</td>
</tr>
</tbody>
</table>
### set_source_table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>X</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields_table</td>
<td>X</td>
<td>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.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>The name to be given to the pivot table</td>
</tr>
<tr>
<td>items</td>
<td>X</td>
<td>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_NAMETYPE_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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a row in the pivot table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a column in the pivot table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot_datafield</code>: The values of the field appear in the body of the table itself. <strong>Note</strong>: If you want to specify the fields by their index number and not by name, use the <code>itemsnr</code> parameter instead.</td>
</tr>
<tr>
<td>itemsnr</td>
<td>X</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a row in the pivot table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a column in the pivot table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;datafield</code>: The values of the field appear in the body of the table itself. <strong>Note</strong>: 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.</td>
</tr>
</tbody>
</table>
set_fieldtype

If you used the `ddic_name` 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:

- `'S'`: Short field labels
- `'M'`: Medium field labels
- `'L'`: Long field labels.

**Note**: If you specify a value that is not valid (or no value at all) the system cannot display the pivot table correctly.

You cannot use this parameter if you used `fields_table` to describe the structure of your data table.

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].`

```abap
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td>Name of the field in the pivot table</td>
</tr>
<tr>
<td>type</td>
<td></td>
<td>New type of the field. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;rowfield</code>: Each value of the field is a row of the pivot table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;columnfield</code>: Each value of the field is a column of the pivot table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>i_oi_pivot=&gt;datafield</code>: The field is part of the body of the table</td>
</tr>
<tr>
<td>pivot</td>
<td></td>
<td>The name of the pivot table in the workbook</td>
</tr>
<tr>
<td>consolidation</td>
<td></td>
<td>The summarization type of the field. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>ADD</code>: Sum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>CNT</code>: Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>MIN</code>: Minimum value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>MAX</code>: Maximum value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>AVG</code>: Mean value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>NOP</code>: No special summarization option</td>
</tr>
</tbody>
</table>

**get_allfields**

Use this method to return a list of the fields in the pivot table.

```plaintext
CALL METHOD pivot->get_allfields
    EXPORTING pivottable = pivottable
        no_flush = no_flush
    IMPORTING allfields = allfields
        error = error
        retcode = retcode.
```

**Description of parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pivottable</td>
<td></td>
<td>The name of the pivot table for which you want to find out the field names</td>
</tr>
<tr>
<td>allfields</td>
<td></td>
<td>An internal table (with the type <code>soi_nametype_table</code>) containing the names of the fields in the pivot table.</td>
</tr>
</tbody>
</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
  IMPORTING allpivottables = allpivottables
                         error       = error
                         retcode        = retcode.

Description of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allpivottables</td>
<td></td>
<td>An internal table with the type soi_fields_name_table containing the names of the pivot tables in the worksheet in the order in which they occur.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td></td>
<td>The name of the field in the pivot table by which the data is grouped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> The name is the field label as it appears in the header of the pivot table, <strong>not</strong> the field name from the ABAP Dictionary.</td>
</tr>
<tr>
<td>value</td>
<td></td>
<td>The value of the field that you specified in row which you want to expand the entries in the pivot table</td>
</tr>
<tr>
<td>pivottable</td>
<td></td>
<td>The name of the pivot table in the workbook</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| row       |          | The name of the field in the pivot table by which the data is grouped.  
           |          | **Note**: The name is the field label as it appears in the header of the pivot table, not the field name from the ABAP Dictionary. |
| value     |          | The value of the field that you specified in row 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**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>C(256)</td>
<td>The name of a field in the data table</td>
</tr>
</tbody>
</table>
The type that the field should have in the pivot table. The following values are possible:

- \texttt{i\_oi\_pivot=>rowfield}: Each value of the field is a new row in the pivot table
- \texttt{i\_oi\_pivot=>columnfield}: Each value of the field is a new column in the pivot table
- \texttt{i\_oi\_pivot=>datafield}: The values of the field appear in the body of the table itself

This data type also has a corresponding internal table type called \texttt{SOI\_NAMETYPE\_TABLE}.

### Data Type \texttt{SOI\_COLTYPE\_ITEM}

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>col</td>
<td>I</td>
<td>The position of a field in the data table</td>
</tr>
</tbody>
</table>
| type  | I    | The type that the field should have in the pivot table. The following values are possible:  
  - \texttt{i\_oi\_pivot=>rowfield}: Each value of the field is a new row in the pivot table  
  - \texttt{i\_oi\_pivot=>columnfield}: Each value of the field is a new column in the pivot table  
  - \texttt{i\_oi\_pivot=>datafield}: The values of the field appear in the body of the table itself |
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:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>check_doi.exe</td>
<td>Checks the paths, installation, and Registry keys for desktop office</td>
</tr>
<tr>
<td></td>
<td>applications and the SAP Desktop Office Integration OCX files. Runs</td>
</tr>
<tr>
<td></td>
<td>as a standalone program.</td>
</tr>
</tbody>
</table>
DOI Installation Test Using a Standalone Executable Program

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doi_word_check.doc [Page 154]</td>
<td>Checks the paths, installation, and Registry keys for desktop office</td>
</tr>
<tr>
<td></td>
<td>applications and the SAP Desktop Office Integration OCX files. Runs as a</td>
</tr>
<tr>
<td></td>
<td>macro in a Word document.</td>
</tr>
<tr>
<td>doi_xls_check.xls [Page 154]</td>
<td>Checks the paths, installation, and Registry keys for desktop office</td>
</tr>
<tr>
<td></td>
<td>applications and the SAP Desktop Office Integration OCX files. Runs as a</td>
</tr>
<tr>
<td></td>
<td>macro in an Excel workbook.</td>
</tr>
<tr>
<td>TableFactoryTest.doc [Page 155]</td>
<td>Checks the SAP TableFactory OCX installation.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Document Type</th>
<th>Document Extension</th>
<th>ProgID in Registry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Word document</td>
<td>.DOC</td>
<td>Word.Document</td>
</tr>
<tr>
<td>Microsoft Excel workbook</td>
<td>.XLS</td>
<td>Excel.Sheet</td>
</tr>
<tr>
<td>Microsoft Powerpoint presentation</td>
<td>.PPT</td>
<td>PowerPoint.Show</td>
</tr>
<tr>
<td>Rich Text Format</td>
<td>.RTF</td>
<td>The application you have associated with files of this type using the Open With dialog box in Windows</td>
</tr>
<tr>
<td>Portable Document Format</td>
<td>.PDF</td>
<td>The application you have associated with files of this type using the Open With dialog box in Windows</td>
</tr>
<tr>
<td>Quicktime Movie</td>
<td>.MOV</td>
<td>Quicktime.Mov</td>
</tr>
<tr>
<td>Lotus 123 Workbook</td>
<td>123</td>
<td>Lotus123.Workbook</td>
</tr>
<tr>
<td>Visio Drawing</td>
<td>VSD</td>
<td>Visio.Drawing</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>HKEY_CLASSES_ROOT.*</th>
<th>Content type &amp; ProgID</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEY_CLASSES_ROOT&lt;ProgID&gt;\shell\open\command</td>
<td>Location of the application</td>
</tr>
<tr>
<td>HKEY_CLASSES_ROOT\MIME\database\content Type</td>
<td>Extension</td>
</tr>
</tbody>
</table>

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
   \[ \text{C\_OI\_CONTAINER\_CONTROL\_CONTAINER} => \text{GET\_DOCUMENT\_VIEWER}. \]
   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 \text{INIT\_VIEWER} [Page 159].

4. You load the document into the Document Viewer using either
   \text{VIEW\_DOCUMENT\_FROM\_URL} [Page 160] or \text{VIEW\_DOCUMENT\_FROM\_TABLE} [Page 161].

5. You close the document using the \text{CLOSE\_DOCUMENT} [Page 162] method.

6. Once you are finished with the Document Viewer instance, you destroy it using the \text{DESTROY\_VIEWER} [Page 162] method.

Coding Example

This extract represents only the steps listed in \text{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'.

\[ \text{For precise details of the methods, refer to the individual method documentation. For details of how to use SAP Container controls, refer to the \text{SAP Container [Ext.]} documentation.} \]

Data Declarations

You must declare the following data objects in your program:

\[ \text{DATA: custom_container TYPE REF TO cl\_gui\_custom\_container, document_viewer TYPE REF TO i\_oi\_document\_viewer.} \]

Instantiating the Container and Document Viewer

This is best done in a PBO module. \text{It must only happen once}. You can ensure this by checking whether one of the controls (for example, \text{custom\_container}) is still initial, and only executing the following statements if it is.

\[ \text{IF custom\_container IS INITIAL.} \]
\[ \text{CREATE OBJECT custom\_containerEXPORTING container\_name = 'CONTAINER100'.} \]
CALL METHOD C_OI_CONTAINER_CONTROL_CREATOR=>GET_DOCUMENT_VIEWER
IMPORTING viewer = document_viewer.

CALL METHOD document_viewer->init_viewer
EXPORTING parent = custom_container.

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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>init_viewer</td>
<td>Initializes the Document Viewer instance</td>
</tr>
<tr>
<td>destroy_viewer</td>
<td>Destroys the Document Viewer instance</td>
</tr>
<tr>
<td>view_document_from_url</td>
<td>Displays a document referenced by a particular URL (either a physical</td>
</tr>
<tr>
<td></td>
<td>location, such as HTTP://..., FTP://..., or FILE://..., or a generated</td>
</tr>
<tr>
<td></td>
<td>SAPR3:// address from the BDS)</td>
</tr>
<tr>
<td>view_document_from_table</td>
<td>Displays a document stored in an internal table</td>
</tr>
<tr>
<td>close_document</td>
<td>Closes the last document to have been opened</td>
</tr>
</tbody>
</table>

**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` or `view_document_from_table`.

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

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.
```

<table>
<thead>
<tr>
<th>Parameter and Type</th>
<th>Optional</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent TYPE REF TO CL_GUI_CONTAINER</td>
<td></td>
<td>The reference variable pointing to the SAP Container instance in which you want the Document Viewer to appear</td>
</tr>
<tr>
<td>document_url TYPE C</td>
<td></td>
<td>The URL of the document you want to display</td>
</tr>
<tr>
<td>show_inplace TYPE C</td>
<td></td>
<td>Indicates whether the document should be displayed in the R/3 window or a separate browser. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'X': Display in R/3 window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ' ': Display in separate browser</td>
</tr>
</tbody>
</table>
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.

```abap
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_initalized       = 4
                   invalid_parameter    = 5.
```

### Parameter and Type

<table>
<thead>
<tr>
<th>Parameter and Type</th>
<th>Optional</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>show_inplace</td>
<td>TYPE C</td>
<td>Indicates whether the document should be displayed in the R/3 window or a separate browser. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 'X': Display in R/3 window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ' ': Display in separate browser</td>
</tr>
<tr>
<td>type</td>
<td>TYPE C</td>
<td>The MIME type of the document.</td>
</tr>
<tr>
<td>subtype</td>
<td>TYPE C</td>
<td>The MIME subtype of the document.</td>
</tr>
<tr>
<td>size</td>
<td>TYPE I</td>
<td>The size of the document in bytes.</td>
</tr>
<tr>
<td>document_table</td>
<td>TYPE STANDARD TABLE</td>
<td>The internal table containing the document</td>
</tr>
</tbody>
</table>

### 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`.

April 2001 161
close_document

However, the mapping between the parameters is not one-to-one. `get_with_table` returns the following relevant parameters:

<table>
<thead>
<tr>
<th>Parameter and Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>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>.</td>
</tr>
<tr>
<td>TYPE SBDST_CONTENT</td>
<td></td>
</tr>
<tr>
<td>components</td>
<td>A components table containing various administrative information about the document, including the MIME type and the size.</td>
</tr>
<tr>
<td>TYPE SBDST_COMPONENTS</td>
<td></td>
</tr>
</tbody>
</table>

**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.
```