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><img src="image" alt="Caution" /></td>
<td>Caution</td>
</tr>
<tr>
<td><img src="image" alt="Example" /></td>
<td>Example</td>
</tr>
<tr>
<td><img src="image" alt="Note" /></td>
<td>Note</td>
</tr>
<tr>
<td><img src="image" alt="Recommendation" /></td>
<td>Recommendation</td>
</tr>
<tr>
<td><img src="image" alt="Syntax" /></td>
<td>Syntax</td>
</tr>
</tbody>
</table>
SET_LOCAL_CONTEXTMENU_MODE..........................................................44
SET_NAVIGATE_ON_DBLCLICK.................................................................45
SET_READONLY_MODE.........................................................................46
SET_SELECTED_TEXT_AS_R3TABLE......................................................46
SET_SELECTED_TEXT_AS_STREAM..........................................................47
SET_SELECTION_INDEXES..................................................................48
SET_SELECTION_POS........................................................................48
SET_SELECTION_POS_IN_LINE.............................................................49
SET_SPACES_ON_INDENT....................................................................50
SET_STATUSBAR_MODE......................................................................50
SET_STATUS_TEXT.............................................................................51
SET_TEXTMODIFIED_STATUS..............................................................51
SET_TEXT_AS_R3TABLE....................................................................52
SET_TEXT_AS_STREAM.......................................................................53
SET_TOOLBAR_MODE.........................................................................53
SET_WORDBREAKPROCEDURE..............................................................54
SET_WORDWRAPBEHAVIOR..................................................................54
UNCOMMENT_LINES..........................................................................55
UNCOMMENT_SELECTION...................................................................56
UNINDENT_LINES.............................................................................57
UNINDENT_SELECTION......................................................................57

Functional Listing...............................................................................58
Creating a Control..............................................................................58
Setting and Getting Text ....................................................................59
Setting and Getting Text Positions.....................................................59
Highlighting and Protecting Text......................................................60
Finding and Replacing Text..............................................................61
Status Bar..........................................................................................61
Toolbar..............................................................................................61
Other Functions................................................................................62

Events..................................................................................................62

Class Constants..................................................................................62

Instance Attributes.............................................................................63

Keyboard and Mouse Control in the Editor Window..............................66

Methods of Class CL_GUI_CFW ..............................................................70
dispatch.............................................................................................70
flush..................................................................................................71
get_living_dynpro_controls.................................................................71
set_new_ok_code.............................................................................72
update_view.....................................................................................72

Methods of Class CL_GUI_OBJECT.........................................................72
is_valid............................................................................................72
free...................................................................................................73

Methods of Class CL_GUI_CONTROL....................................................73
constructor.......................................................................................73
finalize.............................................................................................74
set_registered_events....................................................................74
get_registered_events..................................................................75
is_alive...........................................................................................75
set_alignment ................................................................................................................................. 76
set_position .................................................................................................................................... 76
set_visible ....................................................................................................................................... 77
get_focus ........................................................................................................................................ 77
set_focus ........................................................................................................................................ 77
get_height ....................................................................................................................................... 78
get_width ....................................................................................................................................... 78
Methods of the Class CL_DRAGDROP ...................................................................................... 78
constructor ...................................................................................................................................... 78
add .................................................................................................................................................. 79
clear ................................................................................................................................................ 80
destroy ............................................................................................................................................ 80
get ................................................................................................................................................... 80
get_handle ...................................................................................................................................... 81
modify ............................................................................................................................................. 82
remove ............................................................................................................................................ 83
Methods of the Class CL_DRAGDROPOBJECT ........................................................................ 83
set_flavor ........................................................................................................................................ 83
abort ................................................................................................................................................ 84
**SAP Textedit**

**Use**

You can use the SAP Textedit control to implement an editor for entering and working with text. It can be used as a simple multi-line editor and as an editor for ABAP code. It provides temporary additions that allow you to change the display (for example, to highlight text). Temporary in this sense means that the additions are set and managed at the frontend, but not passed back to the controller. The textedit control does not allow you to set permanent format information or use graphics.

**Features**

The SAP Textedit has three parts:

- An application toolbar containing predefined pushbuttons
- The editor window for displaying text
- The status bar, containing the following five fields:
  - Text message display
  - Details of the text currently selected
  - Current cursor position and total number of lines
  - Change status (‘*’ = changed, ‘ ’ = unchanged)
  - Insert and overwrite modes (‘Ins’ and ‘Ovr’)

The application toolbar and status bar display is optional.
The control provides the following functions:

- Passing text to and from the control using tables
- Display and change modes
- Insert and overwrite mode
- Setting the maximum line length
- Highlighting (also with prefixes at the beginning of the line) and protection against entry for text areas
- Finding out the current cursor position and the position of a selected area
- Finding out and setting the first line displayed in the editor
- Varying line break behavior
  - Code editor: The line break is line-oriented. You can set the maximum line length yourself. The technical implementation uses internal tables.
Using Controls in a WAN

- Text editor
  a) Maximum line length set to width of window
  b) No maximum line length and no line breaks.
- Local context menu containing the same functions as the toolbar
- Optional toolbar for local operations in the control
- A set of keyboard commands for simple navigation within the text. Local operations are simplified by the optional toolbar.
- Cut, copy, and paste for selected text areas, using either the keyboard or the application toolbar.
- Indentation of selected text blocks
- Import or export local files
- Multi-step undo and redo
- Find and replace with the following options:
  - String or part of string
  - Whole word
  - With or without differentiating between upper- and lowercase
  - Within a set of lines
  - Find next
- Events for double-click, F1, F4, drag and drop, and context menus
- Various drag and drop behaviors for files:
  - Display a file without triggering an event
  - Only a single file can be retrieved using drag and drop, and an event is triggered
  - Multiple files can be retrieved using drag and drop, and an event is triggered
- Status indicating whether a text has been changed or not

The control wrapper is implemented in the global class CL_GUI_TEXTEDIT in development class SAPTEXTEDIT.

The development class SAPTEXTEDIT contains example and test programs.

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` 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`. 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.
Special Considerations for the SAP Textedit

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 the SAP Textedit

Modified texts are always transferred synchronously to and from the frontend as a whole text. This means that, in a WAN, transfer times may be very long, regardless of the available bandwidth.

The worst-affected methods are those in which you pass tables as parameters. These are:

- SET_TEXT_AS_STREAM [Page 52]
- SET_SELECTED_TEXT_AS_STREAM [Page 46]
- GET_TEXT_AS_STREAM [Page 24]
- GET_SELECTED_TEXT_AS_STREAM [Page 21]
- SET_TEXT_AS_R3TABLE [Page 51]
- SET_SELECTED_TEXT_AS_R3TABLE [Page 45]
- GET_TEXT_AS_R3TABLE [Page 23]
- GET_SELECTED_TEXT_AS_R3TABLE [Page 20]

The SAP Textedit uses an attribute cache, which stores information about whether the text has been modified, and the current line and column position. Reading these attributes therefore does not necessarily involve round trips to the frontend, since the relevant data is often available from the attribute cache. The call CL_GUI_CFW=>FLUSH [Page 70] is still necessary, however, since you cannot tell as the user when the data from the cache is valid.

Methods

- Arranged Alphabetically [Page 11]
- Arranged by Function [Page 57]

Alphabetical Listing

You requested application help. The following help is available for the current R/3 context:
AUTO_REDRAW

Use
This method uses the administration of the reference counter M_AUTOREDRAW_REFCOUNTER (a read-only instance attribute [Page 62]) to determine when the control is redrawn. The control is only redrawn when the value of the reference counter is zero. This allows you to nest calls.

Features

call method textedit->auto_redraw
   exporting
       enable_redraw = enable_redraw
   exceptions
       error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_redraw</td>
<td>Reference counter</td>
<td>false: Reference counter is increased</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true: Reference counter is decreased</td>
</tr>
</tbody>
</table>

⚠️ You should only use this method when absolutely necessary. If another window is open over the text window and you set the focus to the text window, parts of the other window will appear in the text window. The text will not be displayed properly until the reference counter reaches zero.

COMMENT_LINES

Use
Use this method to designate a set of lines as comment lines.

Features

call method textedit->comment_lines
   exporting
       from_line = from_line
       to_line = to_line
   exporting
       enable_editing_protected_text = enable_editing_protected_text
   exceptions
       error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>First line of block</td>
<td></td>
</tr>
<tr>
<td>to_line</td>
<td>End of block</td>
<td></td>
</tr>
</tbody>
</table>
COMMENT_SELECTION

Use

Use this method to make the selected lines into comment lines. The starting and finishing lines are always converted entirely to comment lines, even if the user did not select all of them.

Features

call method textedit->comment_selection

exporting
   enable_editing_protected_text = enable_editing_protected_text

exceptions
   error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_editing_protected_text</td>
<td>Editing for protected sections of text</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>false Protected sections cannot be edited (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true Protected sections can be edited</td>
</tr>
</tbody>
</table>

CONSTRUCTOR

Use

The constructor creates, initializes, and positions the control.
The constructor method is called automatically when you instantiate the class (create object statement). You generally pass the parameters of the method in the create object statement.

**Features**

create object textedit

- exporting
  - parent = parent
  - lifetime = lifetime
  - max_number_chars = max_number_chars
  - style = style
  - filedrop_mode = filedrop_mode
  - wordwrap_mode = wordwrap_mode
  - wordwrap_position = wordwrap_position
  - wordwrap_to_linebreak_mode = wordwrap_to_linebreak_mode

- exceptions
  - error_cntl_create = 1
  - error_cntl_init = 2
  - error_cntl_link = 3
  - error_dp_create = 4
  - gui_type_not_supported = 5.

---

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>Parent of the instance, that is, the contain in which the control is to be displayed</td>
<td></td>
</tr>
<tr>
<td>lifetime</td>
<td>Lifetime management parameter specifying the lifetime of the control</td>
<td>cntl_lifetime_imode: The control remains alive for the lifetime of the internal session (that is, until a statement such as leave program or leave to transaction) cntl_lifetime_dynpro: The control remains alive for the lifetime of the screen (that is, while it remains in the screen stack). It is not destroyed, for example, by a call screen or call transaction statement.</td>
</tr>
<tr>
<td>max_number_chars</td>
<td>Sets the maximum number of characters that can be entered in the control</td>
<td></td>
</tr>
<tr>
<td><strong>style</strong></td>
<td>Controls the appearance and behavior of the control.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Constants from the class CL_GUI_CONTROL that begin with WS_* You can combine styles by adding the constants together. The default value sets a suitable combination of style constants internally.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>filedrop_mode</strong></th>
<th>Parameter controlling drag and drop behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>dropfile_event_off&lt;br&gt;Only a file that replaces the full existing text can be used (default value).&lt;br&gt;dropfile_event_single&lt;br&gt;Only one file can be used. The file is not included in the editor. The file drop event is triggered instead. You can specify the path of the dropped file using the method GET_PATH_OF_DROPPED_FILES [Page 20].&lt;br&gt;dropfile_event_multiple&lt;br&gt;You can use more than one file. The file is not included in the editor. The file drop event is triggered instead. You can specify a list of the paths of the dropped files using the method GET_PATH_OF_DROPPED_FILES [Page 20].</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>wordwrap_mode</strong></th>
<th>Line break behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>wordwrap_off&lt;br&gt;No line break&lt;br&gt;wordwrap_at_windowborder&lt;br&gt;Line break at the edge of the window (default value)&lt;br&gt;wordwrap_at_fixed_position&lt;br&gt;Line break at a fixed position&lt;br&gt;For further information, refer to Class Constants [Page 61]</td>
<td></td>
</tr>
</tbody>
</table>
The advantage of using containers is that you can set certain essential display attributes (position on the screen, behavior when the window is resized, and so on) when you create them. If you do not use a container, you must set all of the attributes by hand, using the method SET_WINDOW_PROPERTY [Ext].

**DELETE_TEXT**

**Use**

This method deletes all of the text in the control.

**Features**

```
call method textedit->delete_text
exceptions
  error_cntl_call_method = 1.
```

**EMPTY_UNDO_BUFFER**

**Use**

Use this method to empty the undo buffer of the control. After the method has been executed, no previous actions can be undone (including the method call itself). The undo buffer is built up again by subsequent actions in the control. The undo buffer is also implicitly deleted when you protect text areas against input using the PROTECT_LINES [Page 30] and PROTECT_SELECTION [Page 31] methods.

**Features**

```
call method textedit->empty_undo_buffer
```
FIND_AND_REPLACE

exceptions
error_cntl_call_method = 1.

FIND_AND_REPLACE

Use
This method allows you to find and replace text.

Features

call method textedit--->find_and_replace
exporting
case_sensitive_mode = case_sensitive_mode
replace_string = replace_string
search_string = search_string
whole_word_mode = whole_word_mode
changing
string_found = string_found

exceptions
invalid_parameter = 1
error_cntl_call_method = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| case_sensitive_mode | Upper- and lowercase                                  | true
Observe
false
Do not observe (default value) |
| replace_string      | Text to replace occurrences of the search string       |                                                     |
| search_string       | Text to be found/replaced                             |                                                     |
| whole_word_mode     | Only find whole words                                 | true
Only find whole words
false
Find whole words and parts of words (default value) |
| string_found        | Return value specifying how often the text was found   |                                                     |

If you do not specify REPLACE_STRING or SEARCH_STRING, the system calls the Find and Replace dialog box.
FIND_AND_SELECT_TEXT

If no text is selected, the search starts at the current cursor position. If text is selected, the search starts at the second character of the selection, but is not restricted to the selected text.

When it reaches the end of the text, the method returns to the beginning of the text. The text may thus be searched more than once.

⚠️ If you want to replace all occurrences of a search string in a text, use the method REPLACE_ALL [Page 37].

FIND_AND_SELECT_TEXT

Use

This method allows you to find and select text.

Features

call method textedit->find_and_select_text
  exporting
    case_sensitive_mode = case_sensitive_mode
    search_string = search_string
    whole_word_mode = whole_word_mode
  changing
    string_found = string_found
  exceptions
    invalid_parameter = 1
    error_cntl_call_method = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| case_sensitive_mode | Upper- and lowercase                             | true
|                   |                                                  | Observe                                              |
|                   |                                                  | false                                               |
|                   |                                                  | Do not observe                                       |
|                   |                                                  | (default value)                                      |
| search_string     | Text to be found/replace                         | true
|                   |                                                  | Only find whole words                                |
|                   |                                                  | false                                               |
|                   |                                                  | Find whole words                                     |
|                   |                                                  | and parts of words                                   |
|                   |                                                  | (default value)                                      |
| whole_word_mode   | Only find whole words                            | true
|                   |                                                  | Only find whole words                                |
|                   |                                                  | false                                               |
|                   |                                                  | Find whole words and parts of words (default value)  |
| string_found      | Return value specifying how often the text was   | true
|                   | found                                           |                                                    |
|                   |                                                  |                                                     |
If you do not specify the SEARCH_STRING parameter, the system calls the *Find* dialog box.

If no text is selected, the search starts at the current cursor position. If text is selected, the search starts at the second character of the selection, but is not restricted to the selected text.

When it reaches the end of the text, the method returns to the beginning of the text. The text may thus be searched more than once.

### GET_FIRST_VISIBLE_LINE

#### Use
This method returns the line number of the topmost visible line in the control.

#### Features

```plaintext
call method textedit->get_first_visible_line
    importing
        line = line
    exceptions
        error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>Line number. The line numbering begins at one.</td>
</tr>
</tbody>
</table>

### GET_LINE_TEXT

#### Use
This method returns the text in line number LINE_NUMBER.

#### Features

```plaintext
call method textedit->get_line_text
    exporting
        line_number = line_number
    importing
        text = text
    exceptions
        invalid_parameter = 1
        error_cntl_call_method = 2.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td></td>
</tr>
</tbody>
</table>
Due to restrictions on communication between the frontend and backend, texts can only be transferred with a length of up to 256 characters.

Workaround for texts longer than 256 characters:
Select a whole line and use the method GET_SELECTED_TEXT_AS_R3TABLE [Page 20] or GET_SELECTED_TEXT_AS_STREAM [Page 21].

GET_PATH_OF_DROPPED_FILES

Use
This method returns a list of the paths of files used in drag and drop. The list is updated in every drag and drop operation. To set various drag and drop attributes, use the method SET_FILEDROP_MODE [Page 41].

Features
call method textedit->get_path_of_dropped_files

exporting
    table = table

exceptions
    error_dp = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>R/3 table containing the paths</td>
<td></td>
</tr>
</tbody>
</table>

GET_SELECTED_TEXT_AS_R3TABLE

Use
This method allows you to save the selected text as an R/3 table (without information about the line breaks). This depends on the behavior of the line breaks that you have defined in the CONSTRUCTOR [Page 13], especially for soft carriage returns.

Features
call method textedit->get_selected_text_as_r3table
GET_SELECTED_TEXT_AS_STREAM

importing
  table = table

exceptions
  error_dp = 1
  potential_data_loss = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>R/3 table</td>
</tr>
</tbody>
</table>

⚠️

If you use a table with shorter lines than those in the SAP Textedit, the text from the control is only passed up to the length of the table line. The remaining text is lost.

If the lines of the table are longer than the lines of the control, the table lines are filled with trailing spaces.

GET_SELECTED_TEXT_AS_STREAM

Use

This method allows you to place the entire text from the editor in an R/3 table as a stream (with information about line breaks).

Features

call method textedit->get_selected_text_as_stream

importing
  selected_text = selected_text

exceptions
  error_dp = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selected_text</td>
<td>R/3 table</td>
</tr>
</tbody>
</table>

For an example of how to work with line break information, refer to the demonstration program SAPTEXTEDIT_TEST_1, include SAPTEXTEDIT_TEST_1F01, subroutine FIND_CR, which replaces the carriage return line feeds.
GET_SELECTION_INDEXES

Use
This method returns information (in characters) about the position of a selected text.

Features

call method textedit->get_selection_indexes
importing
  from_index = from_index
to_index = to_index
exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_index</td>
<td>Character at which the selection starts. The index is relative to the first character in the editor.</td>
</tr>
<tr>
<td>to_index</td>
<td>Character at which the selection ends.</td>
</tr>
</tbody>
</table>

GET_SELECTION_POS

Use
This method returns the position of a selected text.

Features

call method textedit->get_selection_pos
importing
  from_line = from_line
  from_pos = from_pos
to_line = to_line
to_pos = to_pos
exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>Line in which the selection starts</td>
</tr>
<tr>
<td>from_pos</td>
<td>Position in the line at which the selection starts</td>
</tr>
<tr>
<td>to_line</td>
<td>Line in which the selection ends</td>
</tr>
<tr>
<td>to_pos</td>
<td>Position in the line at which the selection ends</td>
</tr>
</tbody>
</table>
GET_TEXTMODIFIED_STATUS

Use

This method returns the change status of the text. You can use the method GET_TEXTMODIFIED_STATUS [Page 50] to set the changed status for a text.

If you get text using the methods GET_TEXT_AS_R3TABLE [Page 23] and GET_TEXT_AS_STREAM [Page 24], the changed status is not reset.

Features

call method textedit->get_textmodified_status

importing
status = status

exceptions
error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| status     | Changed status| true: Text changed
|            |               | false: Text not changed  |

GET_TEXT_AS_R3TABLE

Use

This method allows you to save the entire text from the editor as an R/3 table (without information about line feeds). This depends on the behavior of the line breaks that you have defined in the CONSTRUCTOR [Page 13], especially for soft carriage returns.

Features

call method textedit->get_text_as_r3table

exporting
only_when_modified = only_when_modified

importing
table = table
is_modified = is_modified

exceptions
error_db = 1
error_cntl_call_method = 2
error_db_create = 3
potential_data_loss = 4.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>R/3 table</td>
<td></td>
</tr>
</tbody>
</table>
| only_when_modified | Attribute when you get text         | true
The text is only written to the R/3 table if the changed status IS:MODIFIED = TRUE. Otherwise, an empty table is passed.
false
The text is retrieved regardless of whether it has been changed (default value) |
| is_modified        | Changed status of the text          | true
Text changed
false
Text not changed |

⚠️

If you use a table with shorter lines than those in the SAP Textedit, the text from the control is only passed up to the length of the table line. The remaining text is lost.

If the lines of the table are longer than the lines of the control, the table lines are filled with trailing spaces.

💡

Setting the parameter ONLY WHEN MODIFIED to TRUE can improve performance, since it minimizes the data exchange between the frontend and backend (and the database).

## GET_TEXT_AS_STREAM

### Use

This method allows you to place the entire text from the editor in an R/3 table as a stream (with information about line breaks).
GET_TEXT_AS_STREAM

Features

call method textedit->get_text_as_r3table

exporting
 only_when_modified = only_when_modified

importing
 text = text
 is_modified = is_modified

exceptions
 error_db = 1
 error_cntl_call_method = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>R/3 table with text</td>
<td></td>
</tr>
</tbody>
</table>
| only_when_modified | Attribute when you get text         | true
 The text is only written to the R/3 table if the changed status IS:MODIFIED = TRUE. Otherwise, an empty table is passed.
 false
 The text is retrieved regardless of whether it has been changed (default value) |
| is_modified        | Changed status of the text          | true
 Text changed
 false
 Text not changed |

Setting the parameter ONLY_WHEN_MODIFIED to TRUE can improve performance, since it minimizes the data exchange between the frontend and backend (and the database).

For an example of how to work with line break information, refer to the demonstration program SAPTEXTEDIT_TEST_1, include SAPTEXTEDIT_TEST_1F01, subroutine FIND_CR, which replaces the carriage return line feeds.
GO_TO_LINE

Use
Use this method to navigate the cursor to the specified line. The cursor position within the line is
the same as it was before the call.

Features

```
call method textedit->go_to_line
  exporting
    line = line
  exceptions
    error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>Line to which you want to navigate</td>
</tr>
</tbody>
</table>

HIGHLIGHT_BREAKPOINT_LINE

Use
Use this method to set a breakpoint.

💡 This method is only used in the ABAP Editor.

Features

```
call method textedit->highlight_breakpoint_line
  exporting
    line = line
    highlight_mode = highlight_mode
  exceptions
    has_no_effect = 1
    error_cntl_call_method = 2
    invalid_parameter = 3.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>Line in which you want to set the breakpoint</td>
<td></td>
</tr>
</tbody>
</table>
HIGHLIGHT_LINES

Use

Use this method to switch the highlighting mode on and off for a range of lines.

Features

call method textedit->highlight_lines

exporting
  from_line = from_line
  to_line = to_line
  highlight_mode = highlight_mode

exceptions
  has_no_effect = 1
  error_cntl_call_method = 2
  invalid_parameter = 3.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>Starting line for the range</td>
<td></td>
</tr>
<tr>
<td>to_line</td>
<td>Final line (inclusive) of the range</td>
<td></td>
</tr>
</tbody>
</table>
| highlight_mode  | Switches highlighting on or off (true/false) | true
                                      |                                      |
|                 |                              | false                               |

If you used the method SET_HIGHLIGHT_COMMENTS_MODE [Page 43] to highlight all comment lines automatically, the method HIGHLIGHT_LINES has no effect.
HIGHLIGHT_SELECTION

Use

This method allows you to switch the highlighting mode for a selection of lines on and off.

Features

call method textedit->highlight_selection

exporting
    highlight_mode = highlight_mode

exceptions
    has_no_effect = 1
    error_cntl_call_method = 2
    invalid_parameter = 3.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>highlight_mode</td>
<td>Switches highlighting on or off (true/false)</td>
<td>true [Switched on (default value)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false [Switched off]</td>
</tr>
</tbody>
</table>

If you used the method `SET_HIGHLIGHT_COMMENTS_MODE [Page 43]` to highlight all comment lines automatically, the method `HIGHLIGHT_SELECTION` has no effect.

INDENT_LINES

Use

Use this method to indent a set of lines by a given number of spaces. Use the parameter M_SPACES_ON_INDENT of the method `SET_SPACES_ON_INDENT [Page 49]` to set the number of spaces.

Features

call method textedit->indent_lines

exporting
    from_line = from_line
    to_line = to_line
    enable_editing_protected_text = enable_editing_protected_text

exceptions
    error_cntl_call_method = 1.
INDENT_SELECTION

Use
Use this method to indent a selected area by a given number of spaces. Use the parameter M_SPACES_ON_INDENT of the method SET_SPACES_ON_INDENT [Page 49] to set the number of spaces.

Only the text that is actually selected is indented (important if you did not select from the beginning of a line or to the end of a line).

Features

```
call method textedit->indent_selection
  exporting
    enable_editing_protected_text = enable_editing_protected_text

exceptions
  error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_editing_protected_text</td>
<td>Editing for protected sections of text</td>
<td>false Protected sections cannot be edited (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true Protected sections can be edited</td>
</tr>
</tbody>
</table>
MAKE_SELECTION_VISIBLE

Use
This method allows you to move a selected area into the visible part of the editor window.

Features

call method textedit->make_selection_visible

exceptions
  error_cntl_call_method = 1.

OPEN_LOCAL_FILE

Use
This method allows you to open a local file.

Features

call method textedit->open_local_file

exporting
  file_name = file_name

exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>Name of the file you want to open. You can specify the path in UNC format.</td>
</tr>
</tbody>
</table>

If you do not set the FILE_NAME parameter, a dialog box appears in which you can enter the name and path of the file.

PROTECT_LINES

Use
You can protect ranges of lines against changes. Use this method to switch the protection on and off.
PROTECT_SELECTION

When a line is protected, its background color changes. If you try to enter text in a protected line, a warning signal sounds.

Features

call method textedit->protect_lines

exporting
    from_line = from_line
    to_line = to_line
    protect_mode = protect_mode
    enable_editing_protected_text = enable_editing_protected_text

exceptions
    error_cntl_call_method = 1
    invalid_parameter = 2.

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>Starting line for the range</td>
<td></td>
</tr>
<tr>
<td>to_line</td>
<td>Final line (inclusive) of the range</td>
<td></td>
</tr>
</tbody>
</table>
| protect_mode        | Switches input protection on or off| true Protection switched on (default value)
                               | false Protection switched off (default value)       |
| enable_editing_protected_text | Editing for protected sections of text | false Protected sections cannot be edited (default value)
                               | true Protected sections can be edited               |

Use

You can protect selections against changes. Use this method to switch the protection on and off.

When a selection is protected, its background color changes. If you try to enter text in a protected line, a warning signal sounds.
**Features**

call method textedit->protect_selection

exporting
  protect_mode = protect_mode

exceptions
  error_cntl_call_method = 1
  invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| protect_mode                      | Switches input protection on or off  | **true**
|                                   | Protection switched on (default value) | **false**
|                                   | Protection switched off               |
| enable_editing_protected_text     | Editing for protected sections of text | **false**
|                                   | Protected sections cannot be edited   |
|                                   | Protected sections can be edited       |

**REGISTER_EVENT_CONTEXT_MENU**

**Use**

Use this method to register context menu event.

To make your coding easier to understand, you should call this method directly after you have created the instance (**CREATE OBJECT**).

**Features**

call method textedit->register_event_context_menu

exporting
  register = register
  appl_event = appl_event
  local_entries = local_entries
**REGISTER_EVENT_DBLCLICK**

exceptions
- error_regist_event = 1
- error_unregist_event = 2
- cntl_error = 3
- event_already_registered = 4
- event_not_registered = 5.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>register</td>
<td>Registering the events</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
</tr>
<tr>
<td>appl_event</td>
<td>Choice between system and application events</td>
<td>space</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'X'</td>
</tr>
<tr>
<td>local_entries</td>
<td>Displays the local entries in the context menu</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
</tr>
</tbody>
</table>

**REGISTER_EVENT_DBLCLICK**

**Use**

Use this method to register the event **DBLCLICK [Ext.]** (double click).

💡

To make your coding easier to understand, you should call this method directly after you have created the instance (**CREATE OBJECT**).

**Features**

call method textedit->register_event_dblclick

exporting
- register = register
- appl_event = appl_event
- navigate_on_dblclick = navigate_on_dblclick
exceptions
   error_regist_event = 1
   error_unregist_event = 2
   cntl_error = 3
   event_already_registered = 4
   event_not_registered = 5.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>register</td>
<td>Registering the events</td>
<td>true (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false Do not register events</td>
</tr>
<tr>
<td>appl_event</td>
<td>Choice between system and application events</td>
<td>space (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'X' Application event</td>
</tr>
<tr>
<td>navigate_on_dblclick</td>
<td>Local navigation</td>
<td>false (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true local navigation</td>
</tr>
</tbody>
</table>

**REGISTER_EVENT_F1**

**Use**

Use this method to register the F1 (function key) event.

💡 To make your coding easier to understand, you should call this method directly after you have created the instance (CREATE OBJECT).

**Features**

```
call method textedit->register_event_f1
   exporting
      register = register
      appl_event = appl_event
```
REGISTER_EVENT_F4

exceptions
  error_regist_event = 1
  error_unregist_event = 2
  cntl_error = 3
  event_already_registered = 4
  event_not_registered = 5.

Parameters | Description | Possible values
---|---|---
register | Registering the events | `true` Register events (default value), `false` Do not register events
appl_event | Choice between system and application events | `space` System event (default value), `'X'` Application event

REGISTER_EVENT_F4

Use

Use this method to register the F4 (function key) event.

💡

To make your coding easier to understand, you should call this method directly after you have created the instance (CREATE OBJECT).

Features

call method textedit->register_event_f4

exporting
  register = register
  appl_event = appl_event

exceptions
  error_regist_event = 1
  error_unregist_event = 2
  cntl_error = 3
  event_already_registered = 4
  event_not_registered = 5.
**REGISTER_EVENT_FILEDROP**

**Use**

Use this method to register the filedrop event.

To make your coding easier to understand, you should call this method directly after you have created the instance (CREATE OBJECT).

**Features**

call method textedit→register_event_filedrop

- **exporting**
  - register = register
  - appl_event = appl_event

- **exceptions**
  - error_regist_event = 1
  - error_unregist_event = 2
  - cntl_error = 3
  - event_already_registered = 4
  - event_not_registered = 5.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| register     | Registering the events    | true
  - Register events (default value)
  - false
  - Do not register events |
| appl_event   | Choice between system and application events | space
  - System event (default value)
  - 'X'
  - Application event |
REPLACE_ALL

Use
This method allows you to replace found text.

Features

```
call method textedit->replace_all
  exporting
    case_sensitive_mode = case_sensitive_mode
    replace_string = replace_string
    search_string = search_string
    whole_word_mode = whole_word_mode

  changing
    counter = counter

  exceptions
    error_cntl_call_method = 1
    invalid_parameter = 2.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>case_sensitive_mode</td>
<td>Upper-/lowercase</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Do not observe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Observe</td>
</tr>
<tr>
<td>replace_string</td>
<td>Text to replace the occurrences of SEARCH_STRING</td>
<td></td>
</tr>
<tr>
<td>search_string</td>
<td>Text to be replaced</td>
<td></td>
</tr>
</tbody>
</table>
whole_word_mode | Only replace whole words | false
| Find whole words and parts of words (default value)
true | Only find whole words |

counter | Return value specifying how many times the search string was replaced |

If you do not set the parameters SEARCH_STRING and REPLACE_STRING, the Find and Replace dialog box appears.

The method starts at the cursor position and searches to the end of the text. You can use the methods SET_SELECTION_INDEXES [Page 47], SET_SELECTION_POS [Page 47], or SET_SELECTION_POS_IN_LINE [Page 48] to set the cursor position. If you want to replace all occurrences within a text, you should set the selection to the first character in the text.

Text replacement is not restricted to the selected area.

SAVE_AS_LOCAL_FILE

Use
Use this method to save text in a local file.

Features

```c
call method textedit->save_as_local_file
exporting
    file_name = file_name
exceptions
    error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>Name of the file you want to save. You can specify the path in UNC format.</td>
</tr>
</tbody>
</table>

If you do not set the FILE_NAME parameter, a dialog box appears in which you can enter the name and path of the file.
SELECT_LINES

Use
Use this method to select a range of lines. The range is automatically placed within the visible section of the control.

Features

```
call method textedit->select_lines
  exporting
    from_line = from_line
    to_line = to_line
  exceptions
    error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>Starting line for the range</td>
</tr>
<tr>
<td>to_line</td>
<td>Final line (inclusive) of the range</td>
</tr>
</tbody>
</table>

You can make the selection invisible to the user as follows:

1. Get the current cursor position
2. Switch off the redraw function by calling the method AUTO_REDRAW [Page 12] and setting the ENABLE_REDRAW parameter to false.
3. Set the required selection. Other actions may follow at this point.
4. If required, set the new position.
5. Reenable the automatic redraw by calling the method AUTO_REDRAW [Page 12] and setting the ENABLE_REDRAW parameter to true.

SET_AUTOINDENT_MODE

Use
Use this method to switch the automatic indentation mode on or off.

For further information about indenting lines, refer to INDENT_LINES [Page 28], UNINDENT_LINES [Page 55], INDENT_SELECTION [Page 29], and UNINDENT_SELECTION [Page 56].

Features

```
call method textedit->set_autoindent_mode
  exporting
    auto_indent = auto_indent
  exceptions
    error_cntl_call_method = 1.
```
**SET_COMMENTS_STRING**

**Use**

Use this method to comment out lines beginning with a particular string.

💡

In the ABAP Editor, the comment character is the asterisk (*).

To switch highlighting on or off for comment lines, use the method **SET_HIGHLIGHT_COMMENTS_MODE [Page 43]**.

**Features**

```plaintext
call method textedit->set_comments_string
  exporting
    comments_string = comments_string
  exceptions
    error_cntl_call_method = 1.
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| auto_indent     | Switches the automatic mode for line indentation on or off | true  
|                 |                                                  | false  

**SET_DRAGDROP**

**Use**

Use this method to set the drag and drop behavior.

**Features**

```plaintext
call method textedit->set_dragdrop
  exporting
    dragdrop = dragdrop.
```
### SET_FILEDROP_MODE

#### Use

Use this method to set the mode used in drag and drop operations with files.

💡 You can also set the mode in the CONSTRUCTOR [Page 13] method when you create an instance.

#### Features

call method textedit->set_filedrop_mode

exporting
   filedrop_mode = filedrop_mode

exceptions
   error_cntl_call_method = 1
   invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| filedrop_mode   | Parameter controlling drag and drop behavior | dropfile_event_off
|                 |                                      | Only a file that replaces the full existing text can be used (default value)     |
|                 |                                      | dropfile_event_single
|                 |                                      | Only one file can be used. The file is not included in the editor. The file drop event is triggered instead. You can specify the path of the dropped file using the method GET_PATH_OF_DROPPED_FILES [Page 20]. |
|                 |                                      | dropfile_event_multiple
|                 |                                      | You can use more than one file. The file is not included in the editor. The file drop event is triggered instead. You can specify a list of the paths of the dropped files using the method GET_PATH_OF_DROPPED_FILES [Page 20]. |
**SET_FIRST_VISIBLE_LINE**

**Use**
Use this method to set the specified line as the first visible line in the editor window.

**Features**
call method textedit->set_first_visible_line
exporting
   line = line
exceptions
   error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>Line number</td>
</tr>
</tbody>
</table>

**SET_HIGHLIGHT_BREAKPOINTS_MODE**

**Use**
This method allows you to switch the highlighting mode for breakpoints on and off.

💡
This method is only used in the ABAP Editor.

**Features**
call method textedit->set_highlight_breakpoints_mode
exporting
   highlight_breakpoints_mode = highlight_breakpoints_mode
exceptions
   error_cntl_call_method = 1
   invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| highlight_breakpoints_mode      | Switches breakpoint highlighting on or off       | true
|                                 |                                                 | Switch on (default value)     |
|                                 |                                                 | false                         |
|                                 |                                                 | Switch off                    |
SET_HIGHLIGHT_COMMENTS_MODE

Use
Use this method to switch automatic highlighting for comment lines on or off. The method SET_COMMENTS_STRING [Page 40] determines the lines that are recognized as comments.

Features

call method textedit->set_highlight_comments_mode

exporting
  highlight_comments_mode = highlight_comments_mode

exceptions
  error_ctl_call_method = 1
  invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>highlight_comments_mode</td>
<td>Switches automatic highlighting on or off</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switch on (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switch off</td>
</tr>
</tbody>
</table>

If this mode is switched on, the methods HIGHLIGHT_SELECTION [Page 27] and HIGHLIGHT_LINES [Page 27] for highlighting areas of text have no effect.

SET_LOCAL_CONTEXTMENU_MODE

Use
Use this method to make the local context menu visible or invisible.
**Features**

call method textedit->set_local_contextmenu_mode

exporting
  visible = visible

exceptions
  error_cntl_call_method = 1
  invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| visible        | Visibility of local context menu | false
|                |                         | Context menu invisible (default) |
|                |                         | true
|                |                         | Context menu visible         |

**SET_NAVIGATE_ON_DBLCLICK**

**Use**

Use this method to set the reaction to a double-click in the control.

**Features**

call method textedit->set_navigate_on_dblclick

exporting
  navigate_on_dblclick_mode = navigate_on_dblclick_mode

exceptions
  error_cntl_call_method = 1
  invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| navigate_on_dblclick_mode    | Mode for reacting to a double-click | true
|                              |                                  | Forward navigation switched on (default value) |
|                              |                                  | Forward navigation switched off               |
**SET_READONLY_MODE**

**Use**

Use this method to switch between read mode and input mode in the textedit control.

You can tell the difference between the two modes from the background colors. The colors are taken from the general SAPgui settings.

**Features**

```plaintext
call method textedit->set_readonly_mode
  exporting
    read_only = read_only
  exceptions
    error_cntl_call_method = 1
    invalid_parameter = 2.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_only</td>
<td>Input mode for the SAP Textedit</td>
<td>true: Input not possible (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>false: Input possible</td>
</tr>
</tbody>
</table>

**SET_SELECTED_TEXT_AS_R3TABLE**

**Use**

This method allows you to insert text from an R/3 table as a stream. The text is inserted at the cursor position. Any selected text is overwritten. The text is passed without information about line breaks.

**Features**

```plaintext
call method textedit->set_selected_text_as_r3table
  exporting
    table = table
    enable_editing_protected_text = enable_editing_protected_text
  exceptions
    error_dp = 1
    error_db_create = 2.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>R/3 table with text</td>
<td></td>
</tr>
</tbody>
</table>
If you use a table with shorter lines than those in the SAP Textedit, the text from the control is only passed up to the length of the table line. The remaining text is lost.

If the lines of the table are longer than the lines of the control, the table lines are filled with trailing spaces.

**SET_SELECTED_TEXT_AS_STREAM**

**Use**
This method allows you to insert text from an R/3 table as a stream. The text is inserted at the cursor position. Any selected text is overwritten. The text is passed with information about line breaks.

**Features**

```plaintext
call method textedit->set_selected_text_as_stream
exporting
  selected_text = selected_text
  enable_editing_protected_text = enable_editing_protected_text

exceptions
  error_dp = 1
  error_db_create = 2.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>selected_text</td>
<td>R/3 table with text</td>
<td></td>
</tr>
<tr>
<td>enable_editing_protected_text</td>
<td>Editing for protected sections of text</td>
<td>false Protected sections cannot be edited (default value) true Protected sections can be edited</td>
</tr>
</tbody>
</table>
SET_SELECTION_INDEXES

Use

Use this method to set a selection for a text area based on character index. The position is automatically placed within the visible section of the control.

Features

call method textedit->set_selection_indexes
exporting
   from_index = from_index
   to_index = to_index

exceptions
   error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_index</td>
<td>Character index at which the selection begins. The index is relative to the first character in the control.</td>
</tr>
<tr>
<td>to_index</td>
<td>Character index at which the selection ends. The index is relative to the first character in the control.</td>
</tr>
</tbody>
</table>

If the “from” and “to” character positions are the same, no text is selected, but the cursor is placed at the corresponding position.

You can make the selection invisible to the user as follows:

1. Get the current cursor position
2. Switch off the redraw function by calling the method AUTO_REDRAW [Page 12] and setting the ENABLE_REDRAW parameter to false.
3. Set the required selection. Other actions may follow at this point.
4. If required, set the old position.
5. Reenable the automatic redraw by calling the method AUTO_REDRAW [Page 12] and setting the ENABLE_REDRAW parameter to true.

SET_SELECTION_POS

Use

Use this method to set a selection for a text area. The position is automatically placed within the visible section of the control.
Features

call method textedit->set_selection_pos

exporting
  from_line = from_line
  from_pos = from_pos
  to_line = to_line
  to_pos = to_pos

exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>Line number at which the selection should begin</td>
</tr>
<tr>
<td>from_pos</td>
<td>Position within the line at which the selection should begin</td>
</tr>
<tr>
<td>to_line</td>
<td>Line number at which the selection should end</td>
</tr>
<tr>
<td>to_pos</td>
<td>Position within the line at which the selection should end</td>
</tr>
</tbody>
</table>

If the “from” and “to” line and position numbers are the same, no text is selected, but the cursor is placed at the corresponding position.

You can make the selection invisible to the user as follows:

6. Get the current cursor position
7. Switch off the redraw function by calling the method AUTO_REDRAW [Ext.] and setting the ENABLE_REDRAW parameter to false.
8. Set the required selection. Other actions may follow at this point.
9. If required, set the old position.
10. Reenable the automatic redraw by calling the method AUTO_REDRAW [Ext.] and setting the ENABLE_REDRAW parameter to true.

SET_SELECTION_POS_IN_LINE

Use

Use this method to position the cursor within a line.

Features

call method textedit->set_selection_pos_in_line

exporting
  line = line
  pos = pos
**SET_SPACES_ON_INDENT**

exceptions
   error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>Line number at which the selection should begin</td>
</tr>
<tr>
<td>pos</td>
<td>Position within the line</td>
</tr>
</tbody>
</table>

**SET_SPACES_ON_INDENT**

**Use**

Use this method to set the number of spaces by which lines or a selected section are indented. For further information, refer to [INDENT_LINES](Page 28), [UNINDENT_LINES](Page 55), [INDENT_SELECTION](Page 29), and [UNINDENT_SELECTION](Page 56). You specify the number of spaces in the class constant M_SPACES_ON_INDENT (for further information, refer to [Class Constants](Page 61)).

**Features**

call method textedit->set_spaces_on_indent
   exporting
      number_of_spaces = number_of_spaces
   exceptions
      error_cntl_call_method = 1
      invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_of_spaces</td>
<td>Number of spaces</td>
</tr>
</tbody>
</table>

**SET_STATUSBAR_MODE**

**Use**

Use this method to show or hide the status bar of the control.

**Features**

call method textedit->set_statusbar_mode
   exporting
      statusbar_mode = statusbar_mode
exceptions
erro cntl_call_method = 1
invalid parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| statusbar_mode       | Status bar mode  | false: Status bar invisible (default value)  
|                      |                  | true: Status bar visible              |

## SET_STATUS_TEXT

### Use
Use this method to display a text in the status bar of the control. To show or hide the status bar, use the method SET_STATUSBAR_MODE [Page 49].

### Features

```
call method textedit->set_status_text
    exporting
    status_text = status_text
excepti ons
    erro cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status_text</td>
<td>Text to be displayed in the status bar</td>
</tr>
</tbody>
</table>

## SET_TEXTMODIFIED_STATUS

### Use
Use this method to set the changed status of the text. The method GET_TEXTMODIFIED_STATUS [Page 23] returns the current changed status of a text.
SET_TEXT_AS_R3TABLE

If you get text using the methods GET_TEXT_AS_R3TABLE [Page 51] and GET_TEXT_AS_STREAM [Page 52], the changed status is not reset.

This method is useful if you want to ignore changes to a text. For example, if you receive text from the control and want to store it permanently in the database, you can set it to unchanged without having to specify any parameters.

**Features**

call method textedit->set_textmodified_status

exporting
status = status

exceptions
error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| status     | Changed status| false
Text not changed (default value)
true
Text changed

SET_TEXT_AS_R3TABLE

**Use**

This method allows you to place text from the control into an R/3 table. Existing text is overwritten. The text is passed without information about line breaks.

**Features**

call method textedit->set_text_as_r3table

exporting
  table = table

exceptions
  error_dp =1
  error_dp_create = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>R/3 table with text</td>
</tr>
</tbody>
</table>

⚠️ If you use a table with shorter lines than those in the SAP Textedit, the text from the control is only passed up to the length of the table line. The remaining text is lost.
SET_TEXT_AS_STREAM

Use
This method allows you to set text from an R/3 table as a stream. Existing text is overwritten. The text is passed with information about line breaks.

Features

call method textedit->set_text_as_stream

exporting
text = text

exceptions
error_dp =1
error_dp_create = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>R/3 table with text</td>
</tr>
</tbody>
</table>

SET_TOOLBAR_MODE

Use
Use this method to show or hide the toolbar.

Features

call method textedit->set_toolbar_mode

exporting
toolbar_mode = toolbar_mode

exceptions
error_cntl_call_method = 1
invalid_parameter = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| toolbar_mode    | Toolbar mode    | false
|                 |                 | Toolbar invisible (default value) |
|                 |                 | true
|                 |                 | Toolbar visible               |
SET_WORDBREAK_PROCEDURE

Use
Use this method to control line break behavior for normal text and ABAP coding.

Features

call method textedit->set_wordbreak_procedure
exporting
text_type = text_type
exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| text_type  | Line break behavior    | text_standard
|            |                        | Text                    |
|            |                        | text_abap               |
|            |                        | ABAP coding             |

SET_WORDWRAP_BEHAVIOR

Use
Use this method to set the behavior of the line break.

Features

call method textedit->set_wordwrap_behavior
exporting
  wordwrap_mode = wordwrap_mode
  wordwrap_position = wordwrap_position
  wordwrap_to_linebreak_mode = wordwrap_to_linebreak_mode
exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### wordwrap_mode

<table>
<thead>
<tr>
<th>Line break behavior</th>
<th>possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>wordwrap_off</td>
<td>No line break</td>
</tr>
<tr>
<td>wordwrap_at_windowborder</td>
<td>Line break at the edge of the window (default value)</td>
</tr>
<tr>
<td>wordwrap_at_fixed_position</td>
<td>Line break at a fixed position</td>
</tr>
</tbody>
</table>

For further information, refer to [Class Constants][Page 61] and the [CONSTRUCTOR][Page 13] method.

### wordwrap_position

If `wordwrap_mode = textedit->wordwrap_at_fixed_position`, position for the automatic line break in a line

Default value: -1

### wordwrap_to_linebreak_mode

Converts soft line breaks to hard ones when you save in the R/3 System

<table>
<thead>
<tr>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
</tr>
<tr>
<td>Soft line breaks are ignored when you save</td>
</tr>
<tr>
<td>true</td>
</tr>
<tr>
<td>Soft line breaks are converted to hard breaks when you save</td>
</tr>
</tbody>
</table>

---

**UNCOMMENT_LINES**

**Use**

Use this method to convert a set of comment lines back into normal lines.

**Features**

```plaintext
call method textedit->uncomment_lines
exporting
from_line = from_line
to_line = to_line
enable_editing_protected_text = enable_editing_protected_text

exceptions
error_cntl_call_method = 1.
```
UNCOMMENT_SELECTION

<table>
<thead>
<tr>
<th>from_line</th>
<th>First line of block</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_line</td>
<td>End of block</td>
</tr>
<tr>
<td>enable_editing_protected_text</td>
<td>Editing for protected sections of text</td>
</tr>
</tbody>
</table>

**Use**

Use this method to convert the selected comment lines into normal lines.

**Features**

```java
call method textedit->uncomment_selection
  exporting
    enable_editing_protected_text = enable_editing_protected_text
  exceptions
    error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| enable_editing_protected_text | Editing for protected sections of text | false  
Protected sections cannot be edited (default value)  
true  
Protected sections can be edited |
UNINDENT_LINES

Use
Use this method to reduce the indentation of a set of lines by a certain number of spaces. Use the parameter M_SPACES_ON_INDENT of the method SET_SPACES_ON_INDENT [Page 49] to set the number of spaces. The indentation is only reduced if there are spaces at the beginning of the lines in question.

Features

```r
call method textedit->unindent_lines
exporting
  from_line = from_line
  to_line = to_line
  enable_editing_protected_text = enable_editing_protected_text
```

```r
exceptions
  error_cntl_call_method = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>from_line</td>
<td>Beginning of the section you want to indent</td>
<td></td>
</tr>
<tr>
<td>to_line</td>
<td>End of the section you want to indent</td>
<td></td>
</tr>
<tr>
<td>enable_editing_protected_text</td>
<td>Editing for protected sections of text</td>
<td>false Protected sections cannot be edited (default value) true Protected sections can be edited</td>
</tr>
</tbody>
</table>

UNINDENT_SELECTION

Use
Use this method to reduce the indentation of the selection by a certain number of spaces. Use the parameter M_SPACES_ON_INDENT of the method SET_SPACES_ON_INDENT [Page 49] to set the number of spaces. The indentation is only reduced if there are spaces at the beginning of the lines in the selection.

Features

```r
call method textedit->unindent_selection
```
Functional Listing

exporting
  enable_editing_protected_text = enable_editing_protected_text

exceptions
  error_cntl_call_method = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Possible values</th>
</tr>
</thead>
</table>
| enable_editing_protected_text | Editing for protected sections of text     | false  
Protected sections cannot be edited (default value) 
true  
Protected sections can be edited |

Functional Listing

You requested application help. The following help is available for the current R/3 context:

This functional listing allows you to find the right method of the SAP Textedit Control for the task you are trying to implement. The functions are divided up by themes. Within each thematic group, the relevant methods are arranged alphabetically.

Areas

Creating and Destroying a Control [Ext.]  
Setting and Getting Text [Ext.]  
Setting and Getting Text Positions [Ext.]  
Highlighting and Protecting Text [Ext.]  
Finding and Replacing Text [Ext.]  
Status Bar [Ext.]  
Toolbar [Ext.]  
Other Functions [Ext.]

Creating a Control

CONSTRUCTOR [Page 13]
Setting and Getting Text

There are two different ways of exchanging data with tables. One way includes information about line breaks, the other does not. The sets of methods have the suffix AS_STREAM (containing line break information) or AS_R3TABLE (without line break information).

### Setting Text

- SET_SELECTED_TEXT_AS_R3TABLE [Page 45]
- SET_SELECTED_TEXT_AS_STREAM [Page 46]
- SET_TEXT_AS_R3TABLE [Page 51]
- SET_TEXT_AS_STREAM [Page 52]

### Getting Text

- GET_LINE_TEXT [Page 19]
- GET_SELECTED_TEXT_AS_R3TABLE [Page 20]
- GET_SELECTED_TEXT_AS_STREAM [Page 21]
- GET_TEXT_AS_R3TABLE [Page 23]
- GET_TEXT_AS_STREAM [Page 24]
- OPEN_LOCAL_FILE [Page 30]
- SAVE_AS_LOCAL_FILE [Page 38]

### Deleting Text

- DELETE_TEXT [Page 16]

---

⚠️ **Warning:** If you use GET methods with the type AS_R3TABLE and use a table with shorter lines than those in the SAP Textedit, the text from the control is only passed up to the length of the table line. The remaining text is lost.

If the lines of the table are longer than the lines of the control, the table lines are filled with trailing spaces.

Setting and Getting Text Positions

There are two ways of accessing text positions. You can either use a line and position within the line, or the absolute character index. Character index means that all of the characters in the text are sequentially numbered.

The main difference is in the way the data passed to the control (as a table) is visualized. If you are using automatic line breaks in the textedit control, the line numbers in the table may not be the same as the line numbers in the control. The same applies, of course, to positions within a line. You define the behavior of the line break in the constructor (for further information, refer to CONSTRUCTOR [Page 13]).

The difference is particularly apparent when you set the line break to WORDWRAP_AT_WINDOWBORDER (automatic line break when the edge of the window is reached). If the user changes the size of the SAPgui and the control container can also be resized, the size of the textedit control will change. If the user makes the SAPgui window smaller, the textedit control will become narrower. Consequently, lines containing words that collide with the window border are automatically split. So even without the user entering any data, the lines and positions within a line are no longer the same as those stored in the internal table at the backend.
It is possible to create a similar example for the line break behavior using \texttt{WORDWRAP\_AT\_FIXED\_POSITION}.

If you need a direct correspondence between a position in the internal table and a position in the textedit control, you should, because of the behavior described above, use the character index.

**Access Using Line and Position Numbers**

Line and position numbers in a line begin at 1, so the first character in a text is at line 1, position 1.

Lines and position numbers outside the text area have a logical value. If a number is less than 1, it is set to 1. If it is greater than the number of lines or positions, it is set to the number of the greatest existing line or position.

<table>
<thead>
<tr>
<th>Getting Text Positions</th>
<th>Setting Text Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{GET_FIRST_VISIBLE_LINE} [Page 19]</td>
<td>\texttt{SET_FIRST_VISIBLE_LINE} [Page 42]</td>
</tr>
<tr>
<td>\texttt{GET_SELECTION_POS} [Page 22]</td>
<td>\texttt{SET_SELECTION_POS} [Page 47]</td>
</tr>
<tr>
<td>\texttt{GO_TO_LINE} [Page 26]</td>
<td>\texttt{SELECT_LINES} [Page 39]</td>
</tr>
<tr>
<td>\texttt{SET_SELECTION_POS_IN_LINE} [Page 48]</td>
<td></td>
</tr>
</tbody>
</table>

**Access Using the Character Index**

The character index starts at 0. Position zero is the position before the first character. Information about the text position is independent of the line break behavior (see above).

<table>
<thead>
<tr>
<th>Getting Text Positions</th>
<th>Setting Text Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{GET_SELECTION_INDEXES} [Page 21]</td>
<td>\texttt{SET_SELECTION_INDEXES} [Page 47]</td>
</tr>
</tbody>
</table>

**Highlighting and Protecting Text**

- **Redrawing the Control**
  - \texttt{AUTO\_REDRAW} [Page 12]

- **Switching Highlighting on and off**
  - \texttt{HIGHLIGHT\_LINES} [Page 27]
  - \texttt{HIGHLIGHT\_SELECTION} [Page 27]

- **Switching Input Protection on and off**
  - \texttt{PROTECT\_LINES} [Page 30]
### Finding and Replacing Text

**Finding and Replacing Text**

- **EMPTY_UNDO_BUFFER [Page 16]**
- **FIND_AND_REPLACE [Page 17]**
- **FIND_AND_SELECT_TEXT [Page 18]**
- **REPLACE_ALL [Page 37]**

### Status Bar

**Setting the Status Bar**

<table>
<thead>
<tr>
<th>Command</th>
<th>Read-Only Instance Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SET_STATUSBAR_MODE [Page 49]</strong></td>
<td>Instance Attributes [Page 62]</td>
</tr>
<tr>
<td><strong>SET_STATUS_TEXT [Page 50]</strong></td>
<td></td>
</tr>
</tbody>
</table>

### Toolbar

**Setting the Toolbar**

<table>
<thead>
<tr>
<th>Command</th>
<th>Read-Only Instance Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SET_TOOLBAR_MODE [Page 52]</strong></td>
<td>Instance Attributes [Page 62]</td>
</tr>
</tbody>
</table>

---

**Comment Lines**

- **SET_COMMENTS_STRING [Page 40]**
- **SET_HIGHLIGHT_COMMENTS_MODE [Page 43]**

**Switching Between Display and Change Mode**

- **SET_READONLY_MODE [Page 45]**

**Moving a Selection to the Visible Part of the Editor Window**

- **MAKE_SELECTION_VISIBLE [Page 30]**
Other Functions

Setting the Line Break

| SET_WORDWRAP_BEHAVIOR [Page 53] |

Events

To react to an event in your ABAP program, you must have registered it. To register events, use the methods

| REGISTER_EVENT_CONTEXT_MENU [Page 32], REGISTER_EVENT_DBLCLICK [Page 33], REGISTER_EVENT_F1 [Page 34], REGISTER_EVENT_F4 [Page 35], and REGISTER_EVENT_FILEDROP [Page 36]. Events that are triggered but for which you are not registered are filtered by the presentation server, and not passed to the application server. For further information, refer to Event Handling [Ext]. |

<table>
<thead>
<tr>
<th>Event</th>
<th>Description / Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTEXT_MENU</td>
<td>Calls the context menu using the right-hand mouse button. For further information, refer to Context Menu [Ext].</td>
</tr>
<tr>
<td>CONTEXT_MENU_SELECTED</td>
<td>Item selection in the context menu. For further information, refer to Context Menu [Ext].</td>
</tr>
<tr>
<td>DBLCLICK</td>
<td>Double click</td>
</tr>
<tr>
<td>F1</td>
<td>Function key F1 pressed</td>
</tr>
<tr>
<td>F4</td>
<td>Function key F4 pressed</td>
</tr>
<tr>
<td>FILEDROP</td>
<td>Drag and drop events. For further information, refer to Drag and Drop [Ext]</td>
</tr>
<tr>
<td>ON_DRAG</td>
<td>Drag and drop events.</td>
</tr>
<tr>
<td>ON_DROP</td>
<td>Drag and drop events.</td>
</tr>
<tr>
<td>ON_DROP_COMPLETE</td>
<td>Drag and drop events.</td>
</tr>
<tr>
<td>ON_GET_FLAVOR</td>
<td>Drag and drop events.</td>
</tr>
</tbody>
</table>

For an example of event handling, refer to the demonstration program SAPTEXTEDIT_TEST_EVENTS in development class SAPTEXTEDIT.

Class Constants

| Class Constant | Type / Value | Description / Use |

April 2001
Instance Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP_COMMENTLINE_IDENTIFIER</td>
<td>TYPE C VALUE *</td>
<td>Used to highlight comment lines when you use the control as a program editor</td>
</tr>
<tr>
<td>BOOL_INITIAL FALSE TRUE</td>
<td>TYPE I VALUE -1, 0, 1</td>
<td>Simulation of type BOOL</td>
</tr>
</tbody>
</table>
| STRING_LENGTH                                 | TYPE I VALUE 256 | Type for strings. When you pass parameters to the control, this definition corresponds to the maximum number of characters that can be transferred between the frontend and application server.  

⚠️ There is no warning if you try to pass a longer string. |
| WORDWRAP_OFF                                  | TYPE I VALUE 0 | No automatic word-wrap in control. You only need this definition when you create an object in the constructor. |
| WORDWRAP_AT_WINDOWBORDER                      | TYPE I VALUE 1 | Automatic word-wrap at the edge of the control window. You only need this definition when you create an object in the constructor. |
| WORDWRAP_AT_FIXEDPOSITION                     | TYPE I VALUE 2 | Automatic word-wrap at a fixed position in the control. You only need this definition when you create an object in the constructor. |

ABAP does not contain any constants with type BOOL. Instead, this type is simulated using the integer constants `false` and `true`. You can address them using `clGuiTextEdit=>true` and `clGuiTextEdit=>false`.

Instance Attributes

The following instance attributes describe the state of the control. They are all flagged as Read Only, so you can use, but not change them.

The frontend attributes are not set from within the control. This means that you can only change them using SET methods from the ABAP proxy. This allows the values of the frontend properties to be reflected in attributes on the ABAP side. To set an attribute, you need to call the corresponding set method, which implicitly updates the corresponding instance attribute. This
Instance Attributes

means that you do not need to call the get methods, thus avoiding flushes, which affect performance.

The instance attributes are initialized in the constructor [Page 13], and are updated when you call the corresponding set methods.

<table>
<thead>
<tr>
<th>Instance Attributes</th>
<th>Type / Value</th>
<th>Description / Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_AUTOREDRAW_REFCOUNTER</td>
<td>TYPE I VALUE 0 READ-ONLY</td>
<td>Reference counter: Determines when the control is redrawn</td>
</tr>
<tr>
<td>M_AUTO_INDENT</td>
<td>TYPE I READ-ONLY</td>
<td>Value for automatic line indentation</td>
</tr>
<tr>
<td>M_COMMENTS_STRING(STRING_LENGTH)</td>
<td>TYPE C READ-ONLY</td>
<td>Character which, when placed at the beginning of a line, signals that the entire line is a comment. To set the value, use the method SET_COMMENTS_STRING [Page 40]. To set the highlighting, use the method SET_HIGHLIGHT_COMMENTS_MODE [Page 43].</td>
</tr>
<tr>
<td>M_HIGHLIGHT_BREAKPOINTS_MODE</td>
<td>TYPE I READ-ONLY</td>
<td>Breakpoint highlighting on or off (true or false). You set this using the method SET_HIGHLIGHT_BREAKPOINTS_MODE [Page 42].</td>
</tr>
<tr>
<td>M_HIGHLIGHT_COMMENTS_MODE</td>
<td>TYPE I READ-ONLY</td>
<td>Comment line highlighting on or off (true or false). To set the attribute, use the method SET_HIGHLIGHT_COMMENTS_MODE [Page 43].</td>
</tr>
<tr>
<td>M_READONLY_MODE</td>
<td>TYPE I READ-ONLY</td>
<td>Text displayed in protected or input mode (true or false) To set the attribute, use the method SET_READONLY_MODE [Page 45].</td>
</tr>
<tr>
<td>Instance Attributes</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><strong>M_SPACES_ON_INDENT</strong></td>
<td><strong>TYPE I VALUE 2 READ-ONLY</strong></td>
<td>Number of spaces that you want to insert or delete for indentations. To set the attribute, use the method <strong>SET_SPACES_ON_INDENT [Page 49]</strong>.</td>
</tr>
<tr>
<td><strong>M_STATUS_TEXT(STRING_LENGTH)</strong></td>
<td><strong>TYPE C READ-ONLY</strong></td>
<td>Text in first element of the status line. To set it, use the method <strong>SET_STATUS_TEXT [Page 50]</strong>.</td>
</tr>
<tr>
<td><strong>M_STATUSBAR_MODE</strong></td>
<td><strong>TYPE I READ-ONLY</strong></td>
<td>Display or suppress the status bar (false or true). To set it, use the method <strong>SET_STATUSBAR_MODE [Page 49]</strong>.</td>
</tr>
<tr>
<td><strong>M_TOOLBAR_MODE</strong></td>
<td><strong>TYPE I READ-ONLY</strong></td>
<td>Display or suppress the toolbar (false or true). To set it, use the method <strong>SET_TOOLBAR_MODE [Page 52]</strong>.</td>
</tr>
<tr>
<td><strong>M_WORDBREAK_PROCEDURE</strong></td>
<td><strong>TYPE I READ-ONLY</strong></td>
<td>Line break behavior for normal text or ABAP code (TEXT_STANDARD or TEXT_ABAP)</td>
</tr>
<tr>
<td><strong>M_WORDWRAP_MODE</strong></td>
<td><strong>TYPE I READ-ONLY</strong></td>
<td>Line break behavior:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No line break (WORDWRAP_OFF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Line break at edge of window (WORDWRAP_AT_WINDOW)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Line break at a fixed position (WORDWRAP_AT_FIXED_POSITION)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You set these values when you create the instance (CONSTRUCTOR [Page 13] method) or explicitly using the method <strong>SET_WORDWRAP_BEHAVIOR [Page 53]</strong>.</td>
</tr>
</tbody>
</table>
## Keyboard and Mouse Control in the Editor Window

### M_WORDWRAP_POSITION

**TYPE I READ-ONLY**

Position in a line at which the line break automatically occurs. You set this value when you create the SAP Textedit instance (CONSTRUCTOR [Page 13] method).

This value is only used if you have set M_WORDWRAP_MODE to 2 or WORDWRAP_AT_FIXED_POSITION.

### M_WORDWRAP_TO_LINEBREAK_MODE

**TYPE I READ-ONLY**

Converts soft line breaks to hard ones when you save in the R/3 System:

- Soft line breaks are ignored when you save. Line ends at the next hard line break (false).
- Soft line breaks are converted to hard breaks on saving (true)

You set this value when you create the SAP Textedit instance (CONSTRUCTOR [Page 13] method).

### M_NAVIGATE_ON_DBLCLICK

**TYPE I READ-ONLY**

Mode for reacting to a double-click. To set the value, use the method SET_NAVIGATE_ON_DBLCLICK [Page 44].

---

### Keyboard and Mouse Control in the Editor Window

**Insert and overwrite mode**

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ins</td>
<td>Switches between insert and overwrite mode</td>
</tr>
</tbody>
</table>
# Keyboard and Mouse Control in the Editor Window

## Clipboard Control

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + Ins</td>
<td>Copies the selected text to the clipboard</td>
</tr>
<tr>
<td>Shift + Ctrl + Ins</td>
<td></td>
</tr>
<tr>
<td>Ctrl + C</td>
<td></td>
</tr>
<tr>
<td>Shift + Ctrl + C</td>
<td></td>
</tr>
<tr>
<td>Ctrl + Del</td>
<td>Cuts the selected text to the clipboard</td>
</tr>
<tr>
<td>Shift + Ctrl + Del</td>
<td></td>
</tr>
<tr>
<td>Ctrl + X</td>
<td></td>
</tr>
<tr>
<td>Shift + Ctrl + X</td>
<td></td>
</tr>
<tr>
<td>Ctrl + Ins</td>
<td>Pastes the contents of the clipboard at the cursor position</td>
</tr>
<tr>
<td>Shift + Ctrl + Ins</td>
<td></td>
</tr>
<tr>
<td>Ctrl + V</td>
<td></td>
</tr>
<tr>
<td>Shift + Ctrl + V</td>
<td></td>
</tr>
</tbody>
</table>

## Undo / Redo

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + Z</td>
<td>Undo</td>
</tr>
<tr>
<td>Alt + Backspace</td>
<td></td>
</tr>
<tr>
<td>Ctrl + Y</td>
<td>Redo</td>
</tr>
</tbody>
</table>

## Navigating in the Text

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Cursor moves one line higher</td>
</tr>
<tr>
<td>{}↓</td>
<td>Cursor moves one line lower</td>
</tr>
<tr>
<td>{}←</td>
<td>Cursor moves one character to the left</td>
</tr>
<tr>
<td>{}→</td>
<td>Cursor moves one character to the right</td>
</tr>
<tr>
<td>PgUp</td>
<td>Cursor moves up one page</td>
</tr>
<tr>
<td>PgDn</td>
<td>Cursor moves down one page</td>
</tr>
<tr>
<td>Home</td>
<td>Cursor moves to beginning of line</td>
</tr>
<tr>
<td>End</td>
<td>Cursor moves to end of line</td>
</tr>
<tr>
<td>Ctrl + \↑</td>
<td>Cursor moves to beginning of the current paragraph</td>
</tr>
</tbody>
</table>
Keyboard and Mouse Control in the Editor Window

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + ↓</td>
<td>Cursor moves to beginning of the next paragraph</td>
</tr>
<tr>
<td>Ctrl + ←</td>
<td>Cursor moves to beginning of next word to the left</td>
</tr>
<tr>
<td>Ctrl + →</td>
<td>Cursor moves to beginning of next word to the right</td>
</tr>
<tr>
<td>Ctrl + PgUp</td>
<td>Cursor moves to the first fully-visible character in the editor window</td>
</tr>
<tr>
<td>Ctrl + PgDn</td>
<td>Cursor moves to the last fully-visible character in the editor window</td>
</tr>
<tr>
<td>Ctrl + Home</td>
<td>Cursor moves to the first character of the text</td>
</tr>
<tr>
<td>Ctrl + End</td>
<td>Cursor moves to the last character of the text</td>
</tr>
</tbody>
</table>

Selecting Text

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + A</td>
<td>Selects entire text</td>
</tr>
<tr>
<td>Ctrl + Shift + A</td>
<td>Selects the text from the current cursor position one whole line upwards</td>
</tr>
<tr>
<td>Shift + ↑</td>
<td>Selects the text from the current cursor position one whole line downwards</td>
</tr>
<tr>
<td>Shift + ←</td>
<td>Selects the text from the current cursor position one character to the left</td>
</tr>
<tr>
<td>Shift + →</td>
<td>Selects the text from the current cursor position one character to the right</td>
</tr>
<tr>
<td>Ctrl + PgUp</td>
<td>Selects one whole editor page upwards from the current cursor position</td>
</tr>
<tr>
<td>Ctrl + PgDn</td>
<td>Selects one whole editor page downwards from the current cursor position</td>
</tr>
<tr>
<td>Ctrl + Home</td>
<td>Selects the text from the current cursor position to the beginning of the line</td>
</tr>
<tr>
<td>Ctrl + End</td>
<td>Selects the text from the current cursor position to the end of the line</td>
</tr>
<tr>
<td>Shift + Ctrl + ↑</td>
<td>Selects the text from the current cursor position to the beginning of the current paragraph</td>
</tr>
<tr>
<td>Shift + Ctrl + ↓</td>
<td>Selects the text from the current cursor position to the end of the current paragraph</td>
</tr>
<tr>
<td>Shift + Ctrl + ←</td>
<td>Selects the text from the current cursor position to the beginning of the next word to the left</td>
</tr>
<tr>
<td>Shift + Ctrl + →</td>
<td>Selects the text from the current cursor position to the beginning of the next word to the right</td>
</tr>
<tr>
<td>Shift + Ctrl + PgUp</td>
<td>Selects the text from the current cursor position to the first fully visible line in the editor window</td>
</tr>
</tbody>
</table>
## Keyboard and Mouse Control in the Editor Window

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift + Ctrl + PgDn</td>
<td>Selects the text from the current cursor position to the last fully visible line in the editor window</td>
</tr>
<tr>
<td>Shift + Ctrl + Home</td>
<td>Selects the text from the current cursor position to the beginning of the text</td>
</tr>
<tr>
<td>Shift + Ctrl + End</td>
<td>Selects the text from the current cursor position to the end of the text</td>
</tr>
<tr>
<td>Press and hold left mouse button and drag mouse</td>
<td>Selects the text block</td>
</tr>
<tr>
<td>Single click the left-hand edge of a line with the left mouse button</td>
<td>Selects the entire line</td>
</tr>
<tr>
<td>Click the left-hand edge of a line with the left mouse button and drag up or down</td>
<td>Selects a successive set of lines</td>
</tr>
<tr>
<td>Ctrl + Single click on the left-hand edge of a line with the left mouse button</td>
<td>Selects entire text</td>
</tr>
</tbody>
</table>

### Find and Replace

If text is selected, the actions affect the selected area. However, this only applies to actions triggered at the frontend. The selection does not affect find and replace functions started from the ABAP program.

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + F</td>
<td>Displays the find and replace dialog</td>
</tr>
<tr>
<td>Shift + Ctrl + F</td>
<td>Displays the find and replace dialog</td>
</tr>
<tr>
<td>Alt + Ctrl + F</td>
<td>Displays the find and replace dialog</td>
</tr>
<tr>
<td>Ctrl + G</td>
<td>Finds the next occurrence of the search string</td>
</tr>
<tr>
<td>Shift + Ctrl + G</td>
<td>Finds the next occurrence of the search string</td>
</tr>
<tr>
<td>Alt + Ctrl + G</td>
<td>Finds the next occurrence of the search string</td>
</tr>
</tbody>
</table>

### Drag & Drop Control

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left mouse button on selected text</td>
<td>Drag and drop: move</td>
</tr>
<tr>
<td>Ctrl + Left mouse button on selected text</td>
<td>Drag and drop: Copy</td>
</tr>
</tbody>
</table>

### Context Menu

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl + F10</td>
<td>Context menu</td>
</tr>
<tr>
<td>Right mouse button</td>
<td>Context menu</td>
</tr>
</tbody>
</table>

### Indentation

<table>
<thead>
<tr>
<th>Keyboard command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>Indents the current line or selected text</td>
</tr>
</tbody>
</table>
Methods of Class CL_GUI_CFW

The class CL_GUI_CFW contains static methods that apply to all instantiated custom controls when you call them.

**dispatch**

Use this method to dispatch application events (see Event Handling [Ext.]) to the event handlers registered for the events. If you do not call the method within the PAI event of your application program, it is called automatically by the system after the PAI has been processed. The method returns a return code from which you can tell if the call was successful.

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>return_code</td>
<td>cl_gui_cfw=&gt;rc_found: The event was successfully directed to a handler method.</td>
</tr>
<tr>
<td></td>
<td>cl_gui_cfw=&gt;rc_unknown: The event was not registered in the event list.</td>
</tr>
<tr>
<td></td>
<td>cl_gui_cfw=&gt;rc_noevent: No event was triggered in a control. The function code was therefore a normal one (for example, from a menu entry).</td>
</tr>
<tr>
<td></td>
<td>cl_gui_cfw=&gt;rc_nodispatch: No handler method could be assigned to the event.</td>
</tr>
</tbody>
</table>
An event can only be dispatched once. After that, it is "spent". Consequently, attempting to dispatch the events a second time does not trigger the handler events again.

### flush

Use this method to synchronize the [automation queue][Ext]. The buffered operations are sent to the frontend using GUI RFC. At the frontend, the automation queue is processed in the sequence in which you filled it.

If an error occurs, an exception is triggered. You must catch and handle this error. Since it is not possible to identify the cause of the error from the exception itself, there are tools available in the Debugger and the SAPgui to enable you to do so.

**Debugger**: Select the option **Automation Controller: Always process requests synchronously**. The system then automatically calls the method `cl_gui_cfw=>flush` after each method called by the Automation Controller.

**SAPGUI**: In the SAPgui settings, under **Trace**, select **Automation**. The communication between the application server and the Automation Controller is then logged in a trace file that you can analyze at a later date.

```plaintext
CALL METHOD cl_gui_cfw=>flush
    EXCEPTIONS CNTL_SYSTEM_ERROR = 1
             CNTL_ERROR = 2.
```

Do not use any more synchronizations in your program than are really necessary. Each synchronization opens a new RFC connection to the SAPgui.

### get_living_dynpro_controls

This method returns a list of reference variables to all active custom controls.

```plaintext
CALL METHOD cl_gui_cfw=>get_living_dynpro_controls
    IMPORTING control_list = control_list.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>control_list</td>
<td>List of reference variables of active custom controls.</td>
</tr>
<tr>
<td></td>
<td>The list has the type <code>CNTO_CONTROL_LIST</code> (defined in class <code>CL_GUI_CFW</code>).</td>
</tr>
</tbody>
</table>
**set_new_ok_code**

You may only use this method in the handler method of a system event. It sets an OK_CODE that triggers PAI processing. This means that data is transferred from the screen to the program, and you can take control of the program in your PAI modules.

CALL METHOD cl_gui_cfw=>set_new_ok_code
   EXPORTING new_code = new_code
   IMPORTING rc = rc.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_code</td>
<td>Function code that you want to place in the OK_CODE field (SY-UCOMM).</td>
</tr>
<tr>
<td>return_code</td>
<td>cl_gui_cfw=&gt;rc_posted: The OK_CODE was set successfully and the automatic field checks and PAI will be triggered after the event handler method has finished. cl_gui_cfw=&gt;rc_wrong_state: The method was not called from the handler method of a system event. cl_gui_cfw=&gt;rc_invalid: The OK_CODE that you set is invalid.</td>
</tr>
</tbody>
</table>

**update_view**

Calling the flush [Page 70] method only updates the automation queue if the queue contains return values.

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. 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.

CALL METHOD cl_gui_cfw=>update_view
   EXCEPTIONS CNTL_SYSTEM_ERROR = 1
   CNTL_ERROR          = 2.

**Methods of Class CL_GUI_OBJECT**

The class CL_GUI_OBJECT contains important methods for custom control wrappers. The only one relevant for application programs is the is_valid [Page 71] method.

**is_valid**

This method informs you whether a custom control for an object reference still exists at the frontend.
CALL METHOD my_control->is_valid
IMPORTING result = result.

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| result     | 0: Custom control is no longer active at the frontend  
1: Custom control is still active |

### free

Use this method to destroy a custom control at the frontend. Once you have called this method, you should also initialize the object reference (FREE my_control).

CALL METHOD my_control->free
  EXCEPTIONS cntl_error = 1  
cntl_system_error = 2.

### Methods of Class CL_GUI_CONTROL

The class CL_GUI_CONTROL contains methods that you need to set control attributes (for example, displaying the control), register events, and destroy controls.

### constructor

This method is called by the control wrapper when you instantiate a control.

To instantiate a SAP control, always call the constructor of its class.

CREATE OBJECT my_control
  EXPORTING   clsid = clsid  
  lifetime = lifetime  
  shellstyle = shellstyle  
  parent = parent  
  autoalign = autoalign
  EXCEPTIONS cntl_error = 1  
cntl_system_error = 2  
create_error = 3  
lifetime_error = 4.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clsid</td>
<td>ID of the class</td>
</tr>
</tbody>
</table>
**finalize**

Lifetime management parameter. The following values are permitted:

- **my_control->lifetime_imode**: The control remains alive for the duration of the internal session (that is, until the session is ended by one of the following statements: `leave program. leave to transaction. set screen 0, leave screen`). After this, the **finalize** [Page 73] method is called.

- **my_control->lifetime_dynpro**: The control remains alive for the lifetime of the screen instance, that is, for as long as the screen remains in the stack. After this, the **free** [Page 72] method is called. Using this mode automatically regulates the visibility of the control. Controls are only displayed when the screen on which they were created is active. When other screens are active, the controls are hidden.

- **my_control->lifetime_default**: If you create the control in a container, it inherits the lifetime of the container. If you do not create the control in a container (for example, because it is a container itself), the lifetime is set to **my_control->lifetime_imode**.

**Shellstyle**

Controls the appearance and behavior of the control. You can pass any constants from the ABAP include `<CTLDEF>` that begin with WS. You can combine styles by adding the constants together. The default value sets a suitable combination of style constants internally.

**parent**

Container in which the SAP Picture Control can be displayed [see also SAP Container [Ext.]].

**autoalign**

- `' ': Control is not automatically aligned
- `'X': Control is automatically aligned. This uses the maximum available space within a container.

**finalize**

This method is redefined by the relevant control wrapper. It contains specific functions for destroying the corresponding control. This method is called automatically by the **free** [Page 72] method, before the control is destroyed at the frontend.

CALL METHOD my_control->finalize.

**set_registered_events**

Use this method to register the events of the control. [See also: Event Handling [Ext.]]

CALL METHOD my_control->set_registered_events
  EXPORTING  events            = events
  EXCEPTIONS cntl_error        = 1
                  cntl_system_error = 2
                  illegal_event_combination = 3.
get_registered_events

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Table of events that you want to register for the custom control my_control.</td>
</tr>
</tbody>
</table>

The table events is a list of the events that you want to register. It is defined with reference to table type CNTL_SIMPLE_EVENTS. The table type is based on the structure CNTL_SIMPLE_EVENT, which consists of the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENTID</td>
<td>Event name</td>
</tr>
<tr>
<td>APPL_EVENT</td>
<td>Indicates whether the event is a system event (initial) or an application event (X).</td>
</tr>
</tbody>
</table>

The values that you assign to the field EVENTID are control-specific and therefore described in the documentation of the individual controls.

get_registered_events

This method returns a list of all events registered for custom control my_control.

CALL METHOD my_control->get_registered_events
   IMPORTING events = events
   EXCEPTIONS cntl_error = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Table of events that you want to register for the custom control my_control.</td>
</tr>
</tbody>
</table>

The table events is a list of the events that you want to register. It is defined with reference to table type CNTL_SIMPLE_EVENTS. The table type is based on the structure CNTL_SIMPLE_EVENT, which consists of the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENTID</td>
<td>Event name</td>
</tr>
<tr>
<td>APPL_EVENT</td>
<td>Indicates whether the event is a system event (initial) or an application event (X).</td>
</tr>
</tbody>
</table>

The values that you assign to the field EVENTID are control-specific and therefore described in the documentation of the individual controls.

For general information about event handling, refer to the Event Handling [Ext.] section of the SAP Control Framework documentation.

is_alive

This method informs you whether a custom control for an object reference still exists at the frontend.
### set_alignment

CALL METHOD my_control->is_alive
  RETURNING state = state.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>my_control-&gt;state_dead: Custom control is no longer active at the frontend</td>
</tr>
<tr>
<td></td>
<td>my_control-&gt;state_alive: Custom control is active on the current screen.</td>
</tr>
<tr>
<td></td>
<td>my_control-&gt;state_alive_on_other_dynpro: Custom control is not active on the current screen, but is still active (but invisible) at the frontend.</td>
</tr>
</tbody>
</table>

### set_alignment

Use this method to align the custom control within its container:

CALL METHOD my_control->set_alignment
  EXPORTING alignment = alignment
  EXCEPTIONS cntl_error = 1
  cntl_system_error = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignment</td>
<td>Control alignment</td>
</tr>
</tbody>
</table>

The `alignment` parameter may consist of combinations of the following alignments:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_control-&gt;align_at_left</td>
<td>Alignment with left-hand edge</td>
</tr>
<tr>
<td>my_control-&gt;align_at_right</td>
<td>Alignment with right-hand edge</td>
</tr>
<tr>
<td>my_control-&gt;align_at_top</td>
<td>Alignment with top edge</td>
</tr>
<tr>
<td>my_control-&gt;align_at_bottom</td>
<td>Alignment with bottom edge</td>
</tr>
</tbody>
</table>

You can combine these parameters by adding the components:

alignment = my_control->align_at_left + my_control->align_at_top.

### set_position

Use this method to place the control at a particular position on the screen.

⚠️ The position of the control is usually determined by its container.

CALL METHOD my_control->set_position
  EXPORTING height = height
  left = left
  top = top
  width = width


**set_visible**

Use this method to change the visibility of a custom control.

```plaintext
CALL METHOD my_control->set_visible
  EXPORTING visible = visible
  EXCEPTIONS cntl_error = 1
                   cntl_system_error = 2.
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>visible</td>
<td>X: Custom control is visible</td>
</tr>
<tr>
<td></td>
<td>’ ’: Custom control is not visible</td>
</tr>
</tbody>
</table>

**get_focus**

This static method returns the object reference of the control that has the focus.

```plaintext
CALL METHOD cl_gui_control=>get_focus
  IMPORTING control = control
  EXCEPTIONS cntl_error = 1
                   cntl_system_error = 2.
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>Object reference (TYPE REF TO cl_gui_control) to the control that has the focus.</td>
</tr>
</tbody>
</table>

**set_focus**

Use this static method to set the focus to a custom control.

```plaintext
```
get_height

CALL METHOD cl_gui_control=>set_focus
  EXPORTING  control           = control
  EXCEPTIONS cntl_error        = 1
                   cntl_system_error = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>Object reference (TYPE REF TO cl_gui_control) to the control on which</td>
</tr>
<tr>
<td></td>
<td>you want to set the focus.</td>
</tr>
</tbody>
</table>

get_height

This method returns the height of the control.

CALL METHOD control->get_height
  IMPORTING  height            = height
  EXCEPTIONS cntl_error        = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Current height of the control</td>
</tr>
</tbody>
</table>

get_width

This method returns the width of the control.

CALL METHOD control->get_width
  IMPORTING  width            = width
  EXCEPTIONS cntl_error        = 1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>Current width of the control</td>
</tr>
</tbody>
</table>

Methods of the Class CL_DRAGDROP

The class CL_DRAGDROP contains methods that describe the drag and drop [Ext.] behavior of a custom control.

constructor

The constructor creates an instance for the description of the drag and drop behavior of a control.
CREATE OBJECT dragdrop.

### add

This method adds a new description to the drag and drop behavior. You can store any number of descriptions, but you may not add the same description more than once.

```plaintext
CALL METHOD dragdrop->add
    EXPORTING flavor         = flavor
dragsrc        = dragsrc
droptarget     = droptarget
effect         = effect
effect_in_ctrl = effect_in_ctrl
EXCEPTIONS already_defined = 1
    obj_invalid      = 2.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flavor</td>
<td>Description of the new flavor</td>
</tr>
<tr>
<td>dragsrc</td>
<td>'x': The description is a drag source</td>
</tr>
<tr>
<td>droptarget</td>
<td>'x': The description is a drop target</td>
</tr>
<tr>
<td>effect</td>
<td>Drop effect of the description between different custom controls. The following effects are supported:</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;copy</strong>: Appearance of the mouse when using drag and drop to copy.</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;move</strong>: Appearance of the mouse when using drag and drop to move.</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;none</strong>: Drag and drop is not possible.</td>
</tr>
<tr>
<td></td>
<td>Drop effect of the description in the same custom control. The following effects are supported:</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;copy</strong>: Appearance of the mouse when using drag and drop to copy.</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;move</strong>: Appearance of the mouse when using drag and drop to move.</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;none</strong>: Drag and drop is not possible.</td>
</tr>
<tr>
<td></td>
<td><strong>dragdrop-&gt;use_default_effect</strong>: Uses the same effect specified in the effect parameter.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>already_defined</td>
<td>The specified flavor has already been defined.</td>
</tr>
<tr>
<td>obj_invalid</td>
<td>The object has already been destroyed using the method destroy [Page 79].</td>
</tr>
</tbody>
</table>
clear

If you use the \texttt{copy} and \texttt{move} effects when you define the flavor, the system uses the \texttt{move} effect when the user drags an object normally, and the \texttt{copy} effect when the user presses and holds the CTRL key while dragging.

\section*{clear}

Deletes the contents of the instance. Once you have called this method, you cannot perform any more drag and drop operations on the corresponding custom control.

\begin{verbatim}
CALL METHOD dragdrop->clear
EXCEPTIONS obj_invalid = 1.
\end{verbatim}

\begin{tabular}{|c|c|}
\hline
Exceptions & Description \\
\hline
obj_invalid & The object has already been destroyed using the method \texttt{destroy} [Page 79]. \\
\hline
\end{tabular}

\section*{destroy}

Deletes the contents of the instance. The instance itself is also destroyed. Once you have called this method, you cannot perform any more drag and drop operations on the corresponding custom control.

\begin{verbatim}
CALL METHOD dragdrop->destroy.
\end{verbatim}

\section*{get}

Returns the complete description of a flavor.

\begin{verbatim}
CALL METHOD dragdrop->get
EXPORTING flavor = flavor
IMPORTING isdragsrc = isdragsrc
  isdroptarget = isdroptarget
  effect = effect
  effect_in_ctrl = effect_in_ctrl
EXCEPTIONS not_found = 1
  obj_invalid = 2.
\end{verbatim}

\begin{tabular}{|c|c|}
\hline
Parameters & Description \\
\hline
flavor & Name of the flavor \\
\hline
dragsrc & 'X': The description is a drag source \\
\hline
droptarget & 'X': The description is a drop target \\
\hline
\end{tabular}
get_handle

| effect | Drop effect of the description between different custom controls. The following effects are supported:
| dragdrop->copy: Appearance of the mouse when using drag and drop to copy.
| dragdrop->move: Appearance of the mouse when using drag and drop to move.
| dragdrop->none: Drag and drop is not possible.

| effect_in_ctrl | Drop effect of the description in the same custom control. The following effects are supported:
| dragdrop->copy: Appearance of the mouse when using drag and drop to copy.
| dragdrop->move: Appearance of the mouse when using drag and drop to move.
| dragdrop->none: Drag and drop is not possible.
| dragdrop->use_default_effect: Uses the same effect specified in the effect parameter.

**Exceptions**

| already_defined | The specified flavor has already been defined.

If you use the copy and move effects when you define the flavor, the system uses the move effect when the user drags an object normally, and the copy effect when the user presses and holds the CTRL key while dragging.

**get_handle**

This method returns the handle of the drag and drop position. In most cases, you will not need to use this method. However, for tabular mass data interfaces (such as the SAP Tree), you must copy this handle into the interface table.

```
CALL METHOD dragdrop->get_handle
    IMPORTING  handle = handle
    EXCEPTIONS obj_invalid = 1.
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>Handle of the drag and drop description</td>
</tr>
</tbody>
</table>

**Exceptions**

| obj_invalid | The object has already been destroyed using the method destroy [Page 79]. |
modify

Use this method to change an existing flavor.

CALL METHOD dragdrop->modify
    EXPORTING  flavor         = flavor
                  dragsrc        = dragsrc
                  droptarget     = droptarget
                  effect         = effect
                  effect_in_ctrl = effect_in_ctrl
    EXCEPTIONS not_found   = 1
                        obj_invalid = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flavor</td>
<td>Name of the flavor</td>
</tr>
<tr>
<td>dragsrc</td>
<td>'X': The description is a drag source</td>
</tr>
<tr>
<td>droptarget</td>
<td>'X': The description is a drop target</td>
</tr>
<tr>
<td>effect</td>
<td>Drop effect of the description between different custom controls. The following effects are supported:</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;copy: Appearance of the mouse when using drag and drop to copy.</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;move: Appearance of the mouse when using drag and drop to move.</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;none: Drag and drop is not possible.</td>
</tr>
<tr>
<td>effect_in_ctrl</td>
<td>Drop effect of the description in the same custom control. The following effects are supported:</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;copy: Appearance of the mouse when using drag and drop to copy.</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;move: Appearance of the mouse when using drag and drop to move.</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;none: Drag and drop is not possible.</td>
</tr>
<tr>
<td></td>
<td>dragdrop-&gt;use_default_effect: Uses the same effect specified in the effect parameter.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>not_found</td>
<td>The specified flavor does not exist</td>
</tr>
<tr>
<td>obj_invalid</td>
<td>The object has already been destroyed using the method destroy [Page 79].</td>
</tr>
</tbody>
</table>
If you use the copy and move effects when you define the flavor, the system uses the move effect when the user drags an object normally, and the copy effect when the user presses and holds the CTRL key while dragging.

remove

Use this method to delete a flavor.

CALL METHOD dragdrop->remove
  EXPORTING flavor = flavor
  EXCEPTIONS not_found = 1
                   obj_invalid = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flavor</td>
<td>Name of the flavor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>not_found</td>
<td>The specified flavor does not exist</td>
</tr>
<tr>
<td>obj_invalid</td>
<td>The object has already been destroyed using the method destroy [Page 79].</td>
</tr>
</tbody>
</table>

Methods of the Class CL_DRAGDROPOBJECT

The class CL_DRAGDROPOBJECT describes the context of a drag and drop operation [Ext.]. It contains information about the source object, the flavor of the drag and drop operation, and information about the source and target.

set_flavor

You can only use this method within event handling for the ONGETFLAVOR event. Use the newflavor parameter to determine the flavor that you want to use in the drag and drop operation. You receive a list of available flavors as an event parameter.

CALL METHOD dragdropobject->set_flavor
  EXPORTING newflavor = newflavor
  EXCEPTIONS illegal_state = 1
                  illegal_flavor = 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>newflavor</td>
<td>Name of the flavor</td>
</tr>
</tbody>
</table>
**abort**

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_state</td>
<td>You did not call the method from within event handling for <strong>ONGETFLAVOR</strong>.</td>
</tr>
<tr>
<td>obj_invalid</td>
<td>You used a flavor that is not supported by the current drag and drop situation.</td>
</tr>
</tbody>
</table>

**abort**

Terminates the drag and drop operation immediately. No further events are triggered.

CALL METHOD dragdopobject->abort.