

AltSearch

v 1.3.2

Introduction

AltSearch works just like a normal *Find-and-Replace* function of Writer, but the difference is that AltSearch offers easy access to document features such as paragraph/character styles, footnotes, tables, frames, bookmarks, hyperlinks, etc. and also powerful syntax to specify typical search and replace conditions.

AltSearch provides many extra features compared to the standard OOo Find dialog:

- Fast selection of preset RegEx (Regular Expressions) and extended regular expressions
- Counting of the number of occurrences of the found expression using the **Count** button
- Search or replace strings can contain one or more paragraphs
- Search or replace strings can contain hexadecimal or decimal characters
- Searching for manual page and column breaks, and adding or removing them
- Searching for a block of paragraphs delimited by two text marks
- Once text string is found, the selection can be expanded or reduced by a specified number of characters
- Multiple searches and replacements can be done in one step
- Searching can be done in hyperlinks, bookmarks, Notes, Text fields, Cross-references and Reference marks, either for their content, their name or target marker.

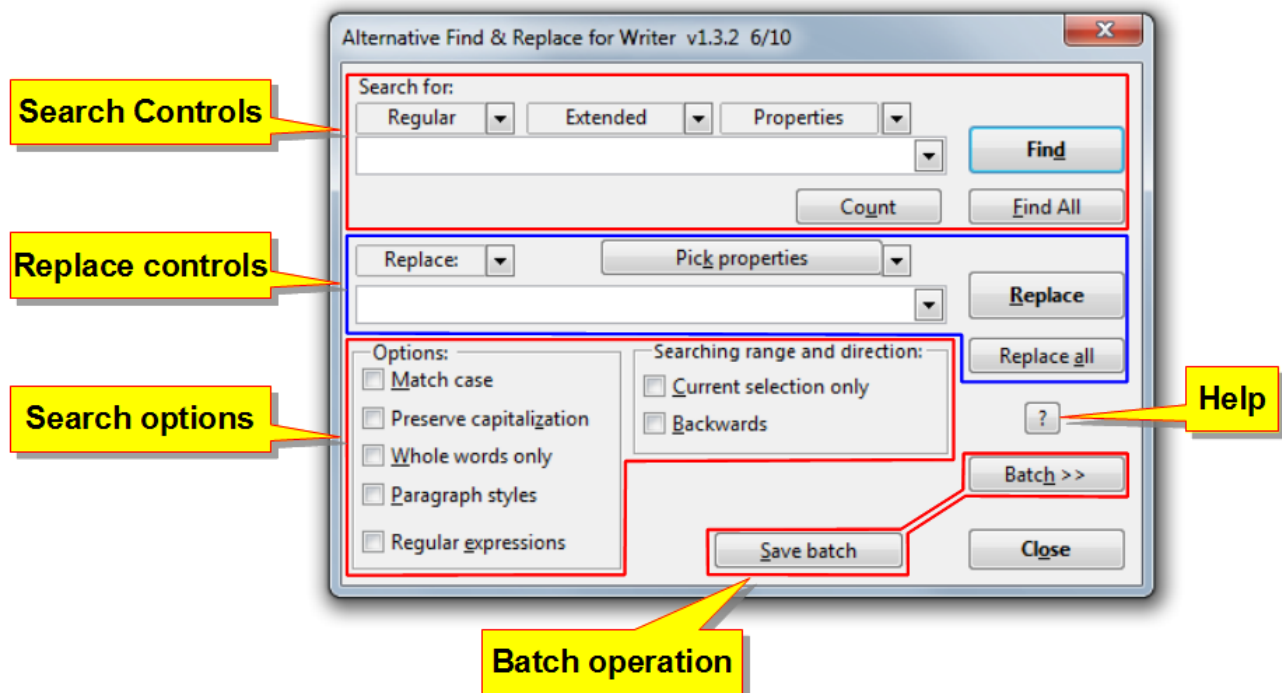
You can also use these properties in the “Replace” operation.

- Tables, Pictures and Text frames can be searched according to their name. You can substitute them with any text, text content, object's name or the clipboard contents.
- Searching and inserting of Footnotes and Endnotes (from OOo version 2.3). You can select the text of all footnotes or endnotes at once and to assign them any style.
- Searching for text that is similar to, or has the same formatting as, the text at the cursor point
- Subexpressions within the search string can be replaced individually by placing them inside parentheses () and referring to them in the replacement string as \1, \2 ...to \9 in order
- The paragraph style, character style, list style, text properties and Hyperlink URL can be set for the search or replace text

- You can replace the found text with clipboard contents, a counter of the number of replacements, or the page number where it is found.
- You can redirect the results of replacement to another text file
- **Batch mode operation:** The search and replace parameters can be saved and re-loaded.
You can save several *search-and-replace* operations in a sequence and quickly execute the whole set later with a single command.
- You can assign hotkeys (keyboard shortcuts) to the saved batch operations.
- **Preserve capitalization option:** If the found text begins with a capital letter, you can optionally capitalize the first word of the replacement text.
- While searching for character style, you can choose a style from the drop-down list.
- You can search for list and paragraph styles using the 'Properties' drop-down list created from the styles used in the target document.
- Instead of using the *Replace* section in AltSearch, you can simply press the **Find all** button. This selects all instances of text that matches the search condition.

Now switch to Writer and carry out any formatting operation, which will be applied to all the instances of the found text simultaneously.

Understanding the AltSearch Window



As shown above, the window has four sections:

Section	What functionality is involved
Search controls	This section has six controls:

	<ol style="list-style-type: none"> 1. Regular drop-down list 2. Extended drop-down list 3. Properties drop-down list 4. Count button 5. Find button 6. Find all button 	<p>Provides RegEx patterns to be used as flexible search patterns.</p> <p>Extends RegEx to Writer-specific search-situations, such as searching inside footnote, endnote, frame, hyperlink, bookmark, cross-reference, etc.</p> <p>Provides properties found in a typical Writer document (such as paragraph/character style, hyperlink, etc.)</p> <p>Counts the occurrence of the search pattern in the selection (or in the entire document, if no selection is made before launching the search.)</p> <p>Finds the next occurrence of the search pattern in the selection (or in the entire document, if no selection is made before launching the search).</p> <p>Selects all the occurrences of the search pattern in the selection (or in the entire document, if no selection is made before launching the search).</p> <p>The main use of this button is to highlight all instances of matching text, and then switch to Writer, and apply some formatting to all instances at once.</p>
Replace controls	<p>This section has four controls:</p> <ol style="list-style-type: none"> 1. Replace 2. Pick properties 3. Replace 	<p>Manipulates the found text block in various ways and presents it for replacement.</p> <p>This control is composed of two parts: a button and a drop-down list.</p> <p>The button actually scans the document and creates a master list of all properties related to page, paragraphs, character, table, frames, footnotes, endnotes, indexes, etc. This list is now available to you as a drop-down list.</p> <p>The idea is to allow you to use one or more of these properties in the replacement.</p> <p>This button replaces the next occurrence of the found text block with the text and/or properties specified in this section.</p>

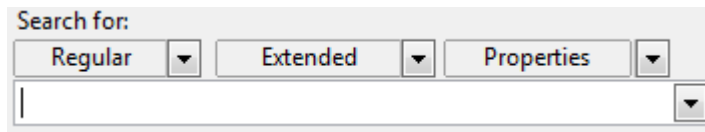
	<p>(Note that sometimes you may want to only change some <i>properties</i> of the found text, rather than replacing with the <i>text</i> itself.)</p> <p>4. Replace all</p> <p>This button replaces the found text block with the replacement text and/or properties. (Sometimes you may only change the <i>property</i> of the found text, rather than replacing with another text block.)</p>
Search options	These check boxes provide quick options for searching (similar to the options found in the Writer's native Find dialog.)
Batch operation controls	<p>A batch means multiple search-and-replace operations arranged in a particular sequence.</p> <p>AltSearch allows you to create any number of batches, save them and run any batch whenever you want.</p> <p>This section has two buttons:</p> <ol style="list-style-type: none"> 1. The Batch>> button launches another window to manage batch operations (creating batch, naming them, saving them and running any batch). 2. The Save batch button saves the current search as a batch. <p>@@@How does this work?</p>

In the following chapters, we will see these sections in details.

The Search Section

To compose a search pattern, follow these two steps:

1. Select a value from the **Regular**, **Extended** or **Properties** drop-down list.

The image shows a search interface with a label 'Search for:' above three drop-down menus. The first menu is labeled 'Regular', the second 'Extended', and the third 'Properties'. Below these menus is a text input box with a cursor inside, and a small drop-down arrow on its right side.

The equivalent markup for the selected option will appear in the Input box below.

For example, if you click the **Regular** list and select the **Tabulator** (=Tab) option, AltSearch adds **\t** to the input box where your cursor is.

In fact, it is not necessary to use the three drop-down lists at all: If you know the RegEx syntax, it is easier to enter the markups directly. For example, you can enter **\t** in the input box instead of using the **Regular** drop-down list.

The lists make the job much easier for you, especially if you use AltSearch rarely.

2. Now place the cursor in the input box and then enter the literal string you want to find, or the value of the property (e.g. the font height) you want to find.

For example, suppose you want to find the string “here.” at the end of a paragraph. (Note that there is a period at the end.)

In that case, you should follow these steps:

- (a) Put the AltSearch in RegEx mode by ticking the **Regular Expressions** check box (located at the bottom of the AltSearch window)
- (b) Enter the RegEx string **here\.** in the input box.
(The RegEx expression **\.** represents the period.)
- (c) Click on the **Regular** drop-down list and select the **Paragraph (ending mark)** option.

At this stage, the input box contains the entry **here\.****p**.

In that string, the **here\.** part is literal (AltSearch will look for its match exactly), and the **p** is a RegEx position marker that represents a paragraph-end.

Thus AltSearch will look for the string “here.” that is placed just before the end of a paragraph. But if the *same* string is found anywhere else, AltSearch will ignore it.

3. Sometimes you may have to build a longer expression by repeating these two steps, as appropriate.

Regular drop-down list

This list contains “standard” RegEx (regular expressions). Refer to Appendix to know how to use RegEx in AltSearch.

The Extended drop-down list

This drop-down menu gives a few frequently occurring search conditions. When you select a particular search condition, AltSearch inserts the equivalent code in the input box.

Sometimes the inserted code contains dummy parameters, which you have to edit. You may also have to add RegEx patterns and/or literal strings to complete the search expression.

The details are given below:

To find-	Inserted code	How to use
Block of paragraphs (of <i>any</i> length) occurring between two specified text strings	[::BigBlock::]	<p>start[::BigBlock::]end</p> <p>In the above formula, enter a RegEx expression or literal text in place of start and end.</p> <p>AltSearch searches for the start string, and when it is found, it looks for the end string. If both are found, the whole block between them is selected.</p> <p>In the replace string, you can use the following parameters:</p> <p>\b contents of the start string & all the found block of paragraphs \e contents of the end string</p> <p>In the start and end expressions, you cannot combine multiple expressions (using delimiters).</p>
Expands the found selection by specified number of characters on both sides	[::Grow 1,1::]	<p>The default pair of numbers is dummy (replace them with actual numbers you want).</p> <p>[::Grow n1,n2::] Expr</p> <p>AltSearch finds a match for the Expr RegEx pattern. Then the found block of text will be expanded by n1 characters to the left and n2 characters to the right.</p> <ul style="list-style-type: none">If you specify a negative number, the selection will be <i>shrunk</i> instead of <i>expanded</i> (those many characters are removed from the original selection). <p>For example, if we use the search string to</p>

To find-	Inserted code	How to use
		<p>[::Grow -1,-1::]text and if the word text exists in the text, it will be found, but only ex will be returned because the expression cuts off one letter from both ends.</p> <p>Limitations:</p> <p>If n1 or n2 are negative, and if they exceed the size of the actual found block, the current search operation will move the cursor position <u>before</u> the found block. As a result, the next search will find the same block. This will repeat endlessly.</p>
Append mark for multiple find-and-replace operations in one step		<p>This trick applies to a situation where you want to search for multiple patterns, and if each of the found string is to be replaced by a corresponding replacement string.</p> <ul style="list-style-type: none"> • Delimit all search strings with the sign. • Also delimit all replacement strings with the sign. <p>For example,</p> <p>Search for: text1 text2 text3 Replace: neco1 neco2 neco3</p> <p>This will search for text1 and will replace it with neco1, then continue the search for text2, replace it by neco2, and then continue the search for text3, replace it by neco3.</p> <p>Limitation:</p> <p>You cannot use with <u>subexpressions</u>.</p>
Text between () (inside of one paragraph)	(([^\)])+)	Finds the text enclosed within a pair of parenthesis ().
Text between [] (inside of one paragraph)	([([^\]])+)	Finds the text enclosed within a pair of square brackets [].
Text between {} (inside of one paragraph)	({([^\}])+)}	Finds the text enclosed within a pair of curly brackets {}.

To find-	Inserted code	How to use
		<p>have to edit the basic code as follows:</p> <p>[::Footnote::] (without any argument) will find the anchor spot of the next footnote².</p> <p>[::Footnote::]LiteralString will find the anchor that contains the literal string³ LiteralString.</p> <p>[::Footnote::]\\ will find the text of the next footnote.</p> <p>[::Footnote::]\\text will find the footnote that contains the string text in its text (it won't search in the anchor)</p> <p>If you enter [::Footnote::]\\ and click the Find all button, AltSearch will select the text of all footnotes. This is handy for assigning a paragraph style to all footnotes at once.</p> <p>If you enter [::Footnote::]\\text and click the Find all button, AltSearch will select the text of all footnotes that contain the literal string text.</p>
Cross-reference marker	[::ReferenceMark::]	<p>Depending upon the desired functionality, you have to edit the basic code as follows:</p> <p>[::ReferenceMark::] will find any text set as a reference marker</p> <p>[::ReferenceMark::]text will find any text set as a reference marker that contains substring text</p> <p>[::ReferenceMark::]\\ref1 will find any text set as a reference marker whose name contains the substring ref1</p> <p>[::ReferenceMark::]\\\\ will find any text set as a reference marker whose text is empty</p>
Cross-reference	[::Reference::]	<p>Depending upon the desired functionality, you have to edit the basic code as follows:</p> <p>[::Reference::] will find all text fields of the</p>

² This refers to a footnote that is inserted without highlighting a text first. The anchor of such a footnote appears as a thin vertical line where the cursor was placed while inserting the footnote. This footnote itself is an example of such a footnote.

³ This refers to a footnote that was inserted after highlighting some text. Such a footnote is anchored to (-and applies to-) the entire highlighted text. This footnote itself is an example of such a footnote.

To find-	Inserted code	How to use
		<p>cross-reference type</p> <p>[::Reference::] above will find cross-references that contain the substring above</p> <p>[::Reference::] \\ref1 will find cross-references whose name contain the substring ref1</p> <p>[::Reference::] \\\\ will find cross-references whose text is empty</p> <p>Tip: First click in text that is a Reference Mark and then choose the Reference option from the Extended list. AltSearch copies the corresponding source name to the Search for box automatically. Now you can search for it immediately.</p>

The Properties drop-down list

This list allows you to search using type or values of the properties (attributes) of paragraphs, fonts, etc.

When you select an option from the drop-down list, AltSearch inserts basic code in the **[:::PropertyName::]** pattern. (Note that the number of colons are not equal on both sides.)

- In case of styles, it inserts the basic code in **[:::PropertyName=::]** pattern, where you have to specify the desired style name after the = sign. AltSearch also pops up another window listing the styles available in the current document. Select the desired style to complete the search expression.

For example, **[:::ParaStyleName=Heading 1::]**.

You can search for diverse types of properties and specify their values in a single operation. For this, you have to delimit the *property=value* pairs with a | character.

For example,

[:::PropertyName1=value1|PropertyName2=value2|PropertyName3=value3::]

Limitation: The OOO search engine does not support all the existing paragraph and character properties. Not all combinations work as expected.

Here are the specific details. (In the following table, the code looks folded only because the column is not wide enough. The actual code is a continuous string.)


Option	Inserted code	Actual usage
Paragraph style	[:::ParaStyleName=::]	[:::ParaStyleName=::] will find all whole paragraph with paragraph style <i>other</i>

		<p>than the Default style</p> <p>[:::ParaStyleName=Example:::] will find whole paragraph with paragraph style Example</p> <p>[:::ParaStyleName=Example:::] something will find text something, but <i>only if</i> it is formatted with paragraph style Example.</p> <p>Limitations:</p> <p>Cannot find some parts of the text with zero length, for example, an empty paragraph.</p> <p>Can not be combined simultaneously with other text properties.</p>
Character style	[:::CharStyleName=:::]	<p>[:::CharStyleName=:::] will find part of the text with character style another than the Default style</p> <p>[:::CharStyleName=Example:::] will find part of the text with character style Example</p> <p>[:::CharStyleName=Example:::] sometext will find text sometext, but <i>only if</i> it is formatted with character style Example.</p> <p>Limitations:</p> <p>It works in forward direction only- The Backward option does not work (AltSearch gives a “<i>not found</i>” message even when matching text exists).</p> <p>Cannot find some parts of the text with zero length, for example, an empty paragraph.</p> <p>Cannot be combined simultaneously with other text properties.</p>
List style	[:::NumberingStyleName=:::]	Use is similar to searches for the paragraph style (see above).
Hyperlink	[:::HyperLinkURL:::]	<p>[:::HyperLinkURL:::] will find all hyperlinks</p> <p>[:::HyperLinkURL:::] link will find the part of the hyperlink containing the text link.</p>
String in URI ⁴	[:::HyperLinkURL=file:::]	[:::HyperLinkURL=file:///c:/

	<code>e:///::]</code>	<p><code>pokus.odt::]</code> will search for a hyperlink with the specified URI of the file (<code>c:/pokus.odt</code>).</p> <p><code>[:::HyperLinkURL=file:///c:/pokus.odt::]link</code> will find part of a hyperlink in which the URL is the string <code>file:///c:/pokus.odt</code> and in which the text contains the text <code>link</code>.</p>
Italic	<code>[:::CharPosture::]</code>	To search for bold <u>and</u> italic text, use- <code>[:::CharPosture CharWeight::]</code>
Bold	<code>[:::CharWeight::]</code>	
Font Name (manually changed name)	<code>[:::CharFontName::]</code>	
Font Size	<code>[:::CharHeight::]</code>	
Font Color	<code>[:::CharColor::]</code>	
Font background (Highlighting)	<code>[:::CharBackColor::]</code>	
Underline	<code>[:::CharUnderline::]</code>	
Index (any)	<code>[:::CharEscapement::]</code>	AltSearch will search for a subscript or superscript.
Subscript (Auto)	<code>[:::CharAutoEscapement=true CharEscapementHeight=58 CharEscapement=-101::]</code>	This code looks for the subscripts with the default settings.
Superscript (Auto)	<code>[:::CharAutoEscapement=true CharEscapementHeight=58 CharEscapement=101::]</code>	This code looks for the subscripts with the default settings.
Index defined by font size and escapement	<code>[:::CharEscapementHeight=70 CharEscapement=30::]</code>	This setting will search for superscript that has relative font size of 70% and is raised/lowered by 30%. (If the <i>CharEscapement</i> is -ve, AltSearch will search for a subscript)
Similar format of	<code>[:::CharWeight::]</code>	This function loads any of the character

characters (based on cursor)		<p>attributes of the current selection that have been manually changed and searches for similarly formatted places.</p> <p>For example, if the cursor is on text for which the name of the font has been manually changed, all places with a changed name of font will be found.</p> <p>In this example, the search box will contain <code>[:::CharFontName:::]</code>.</p> <p>@@@What if multiple things are changed manually?</p> <p>For example, what if font size, type are changed; and also multiple font effects are applied manually?</p>
Same format of characters (based on cursor)	<code>[:::CharWeight=100:::]</code>	<p>This function loads any of the character attributes of the current selection that have been manually changed and searches for <u>identically</u> formatted places.</p> <p>For example, if the cursor is on text where the name of the font has been manually changed, AltSearch will search for the same font name.</p> <p>In our example, the search box will contain <code>[:::CharFontName=Arial:::]</code>.</p>

The Pick Properties Button

AltSearch can find all text attributes and their values used in the current selection **@@@or current document?**. Just click on the **Pick properties** button. AltSearch scans the document and loads all attributes and their values used in the current selection **@@@or current document?** in the list box next to this button. Now click on the down-arrow  and select any attribute/value.

A brief description of most of these properties can be found [here](#).

The Replacement Section

This section derives much of its content from what is *actually* found during the search.

Let us see how the various controls in this section handle the found text.

The Replace drop-down list

This is a heart of the replace section.

Option	Inserted Code	Actual usage
Whole found text	&	<p>The symbol & represents the entire block of text captured by the Search section, without any alterations.</p> <p>If the search expression used [::BigBlock::], & represents only the block of paragraphs found between the <i>start</i> and <i>end</i> texts.</p> <p>If the search expression used [::Note::], [::Field::], [::TextFrame::], [::Picture::], or [::TextTable::], & represents exactly these objects (which are inserted using the clipboard).</p> <p>This code is useful if you want to change only the formatting of the found text.</p>
Text content of object (frame, table, etc)	\o	<p>Same as &. (see above)</p> <p>@@@But what is the difference? Is \o a more limited version of &, in the sense that \o works only within objects, and not in the main running text?</p>
Content of start block	\b	Recall that the Search section uses RegEx pattern (and not literal text) around some of the blocks.
Content of end block	\e	<p>A given RegEx pattern can match a huge variety of actual text blocks. That is why the matched text is represented by these codes.</p> <p>Limitation: these cannot be used together with backreferences.</p> <p>@@@Does AltSearch find all possible matches for a RegEx Pattern as it scans a document, or does it stop looking for other matches when it finds its first match?</p>
Delimiting series of	\b\e	@@@What does this actually represent: (a)

paragraphs		<p>the actual found text blocks at the beginning and end, or-</p> <p>(b) The text found between the beginning and end blocks?</p>
Content of nth group found inside () in search expression	<p>\n</p> <p>Where n is serial number of the backreference ($n \leq 9$)</p>	<p>Note that there is no offset in the value of n (for example, n=2 means the <i>second</i> backreference.)</p> <p>Only 1st level parentheses are valid; nested levels inside them are ignored.</p> <p>Example: In order to find dates in the <i>dd.mm.yyyy</i> format and replace them with dates in the <i>yy.mm.dd</i> format, use the search expression (\d{1,2})\.*(\d{1,2})\.*\d{2,2}(\d{2,2}) and use the replace expression \3-\2-\1.</p> <p>If you need to switch off the processing of subexpressions (e.g. to preserve compatibility with the regular expressions in Writer), you must put the whole search expression within an additional pair of parentheses. Then all other nesting levels of parentheses for replacement will be ignored.</p> <p>Limitation: Using subexpressions is relatively slow and not fully compatible with the original search function in Writer.</p> <p>There is an incompatibility with search wildcards placed immediately after a subexpression, such as (opak)*, which is caused by the principle of sequential searching of sequential blocks of text; see here. In these cases the Count and Find all functions return the correct counts, but other functions (without switching to compatibility mode) will not find anything. In more complicated cases you will need to examine what happens, and experiment to get the best results.</p>
End of paragraph (new paragraph)	\p	Use to begin a new paragraph at any specific place in the replacement text.
Tabulator	\t	Use to insert a tab at any specific place in the replacement text. (equivalent to \x0009 and \#9)
Non-breaking space	\S	Use to insert a non-breaking space at any specific

		place in the replacement text. (equivalent to \x00A0 and \#160)
Manual line break	\n	Use to insert a manual line break at any specific place in the replacement text. (equivalent to \x000A and \#10)
Manual column break before	\c	Use to insert a manual column break at any specific place in the replacement text.
Manual page break before	\m	Use to insert a manual page break before the replacement text.
Manual page break after	\M	Use to insert a manual page break after the replacement text.
Dissolve manual page or column break	\r	@@@How is this used?
A space inserted by code	\x0041 (hexadecimal) \#65 (decimal)	@@What is a typical use?
Replace the found URL by specified URL	\h{URL}	\h{} , \h (without arguments) changes the found text to a hyperlink with a URL of an empty string. (It effectively deletes the found URL. The text of hyperlink stays unchanged.
Replace found SubString by 'ReplString' in URL	\H{ReplString}	To be used when search field has [:::HyperLinkURL=substr:::] . only hyperlinks will be found whose URLs include the substring substr . Now use \H{repl} in the replace box to replace substr in the URLs with replString .
Returns hyperlink's URL of found link	\u	If a URL is found, this expression represents the found URL. If no URL is found, the expression returns nothing. @@@Does it capture URIs also (references to file in the folder system)?
Paragraph style	\P{Text}	Sets up Paragraph style Text in the found paragraph(s). The style is applied to the paragraph containing the text of the replaced expression.

		<p>To set the style to "Default", use <code>\P</code> or <code>\P{}</code>.</p> <p>If this parameter is used a number of times with inserted paragraph(s), the style is changed with every new parameter, and is valid as far as the end of the paragraph.</p> <p>Example: If the expression is replaced using <code>block1\P{Subtitle}\p block2\P{Heading 1}</code> so <code>block1</code> will be inserted and assigned the style <code>Subtitle</code>, and after it a new paragraph with text <code>block2</code> will be inserted and assigned the style <code>Heading 1</code>.</p>
Character style	<code>\C{Quotation}</code>	<p>sets up the Character style <code>Quotation</code> in the found text</p> <p>The style is applied on the whole text of the replaced expression. To set to the "Default" style use <code>\C</code> or <code>\C{}</code>. If this parameter is used a number of times, the character style is changed with every new parameter, and the last is valid as far as the end of replacing expression. Example: If the expression was replaced using <code>block1\C{Quotation}block2\C{Example}</code>, <code>block1</code> will be inserted with the character style <code>Quotation</code>, and after it <code>block2</code> will be inserted and assigned the character style <code>Example</code>.</p>
List style	<code>\N{List 3}</code>	<p>Sets up List style <code>List 3</code> in the found paragraph(s)</p> <p>Applies analogous usage rules to those for the parameter <code>\P{}</code>. List style can be removed with <code>\N</code> or <code>\N{}</code>.</p>
Set up property of object to value	<code>\A{CharBackColor=sHffcc00}</code>	
Set up default formatting by char style	<code>\D</code>	<p>sets up the default formatting for the found text, just like using Ctrl+Shift+Space</p> <p>Applies analogous usage rules to those for the parameter <code>\C{}</code>.</p>
ResetAttributes in place of use	<code>\d</code>	resets text attributes to default only in the place of use.

		Contrary to \D it has no effect on the previously inserted text.
Insert Footnote	\F{NewFootnote}	<p>Inserts a new footnote that contains the text New footnote in the place of replacement</p> <p>Inside the curly brace it is possible to use any of following parameters: \i, \I, &, or \1</p>
Insert Endnote	\E{NewEndnote}	Inserts a new endnote in the place of replacement; similar to \F
Insert marker+text for cross-reference	\B{ref1 text}	<p>Inserts the text text with the marker ref1 for a cross-reference</p> <p>Inside the curly brace it is possible to use any of following parameters: \i, \I, &, or \1</p>
Insert cross-reference to marker	\L{0,0,ref1}	<p>inserts a cross-reference (field) with the parameters 0,0 and reference marker ref1</p> <p>Meaning of numeral parameters:</p> <p><i>first number</i> Type of reference: 0 - page number in Arabic numerals, 1 - chapter number, 2 - the reference text, 3 - above/below, 4 - page number using the numbering type defined in the page style, 5 - category and number of a caption, 6 - caption text, 7 - number of a sequence field (caption)</p> <p><i>second number</i> Type of the source of a reference field; the source is : 0 - a reference mark, 1 - a number sequence field, 2 - a bookmark, 3 - a footnote, 4 - an endnote</p>
Text content of object (frame, table, picture, etc)	\o	<p>Inserts the text content of the found object</p> <p>If the expression was searched for using [::Note::], [::Footnote::], [::Endnote::], [::TextFrame::], [::Picture::], or [::TextTable::], the text inside this object will be inserted. Tables come out with tabs between columns and paragraphs between rows.</p> <p>Limitation: The maximum size of the whole</p>

		<p>resulting text after converting a table is limited to 65 kB.</p> <p>If the expression was searched for using <code>[::Field::]</code>, <code>[::Reference::]</code>, or <code>[::ReferenceMark::]</code>, the displayed text of the anchor or field will be inserted.</p>
Name of object (frame, table, picture, etc)	<code>\O</code>	<p>Inserts the name of the found object</p> <p>If the expression was searched for using:</p> <p><code>[::TextFrame::]</code>, <code>[::Picture::]</code>, or <code>[::TextTable::]</code>, the name of this object will be inserted.</p> <p><code>[::Note::]</code> or <code>[::Field::]</code>, the type of the text field will be inserted</p> <p><code>[::Reference::]</code> or <code>[::ReferenceMark::]</code>, the name of the reference mark will be inserted</p> <p><code>[::Footnote::]</code> or <code>[::Endnote::]</code>, the displayed text of the anchor will be inserted</p>
Paste content of clipboard	<code>\v</code>	inserts the contents of the clipboard
Paste unformatted text only	<code>\V</code>	inserts contents of the clipboard as unformatted text
Preserve formatting with replace (replace & by clipboard)	<code>\f</code>	<p>If <code>&</code> or <code>\O</code> is used in the replace expression, replacement will be realized using the clipboard. If the found text contains text fields, notes, references etc, they will be preserved in their original state.</p>
Insert counter of replace (with Replace all only)	<code>\i</code>	Inserts the occurrence number of the found object or text in a count of the occurrences in the text - this works only if Replace all button is clicked.
Insert page number	<code>\I</code>	
Page number on which search expression is found	<code>\r</code>	<p>Inserts the page number on which the search expression is found.</p> <p>If redirection of the replace expression to another file (<code>\R</code>) is used (see below), the number of the page containing the starting position of the found</p>

		<p>text is inserted in the other file.</p> <p>Limitation: this does not work correctly in footnotes, headers and footers.</p>
Redirect of replacing expression to another file	\R	<p>Redirects the replace expression to another text file.</p> <p>This option causes the replace expression to be inserted into new .odt file instead of replacing the found text. The original file will stay as it is, without changes. To enter the name of the output file, use the format \R{filename}. The name must have the accurate OOo window header format, including the text " - OpenOffice.org Writer". New records from this redirection are always added to the end of the file.</p> <p>Example:</p> <p>If the search expression was searched for using [:::HyperLinkURL::] and the replace expression is Link \i, page \I: & (URL: \u) \p\R, when you click the <u>[Replace all]</u> button all the hyperlinks found in original file will be listed in a new file in the form <i>Link 1, page 1: textOfHyperlink (URL: URLaddress)</i> in separate paragraphs.</p>
Insert a character using the hexadecimal character code	\xhhhh	
Insert a character using the decimal character code	\#dddd	

Pick properties Button and Drop-List

Using the **Pick properties** button, you can update the list of properties and their values for the currently selected object. Now browse the drop-list and choose a property. This adds the corresponding code in the Replace field in this format: **\A{properties=value}**

The rules for this are same as for **\C{}** (see the table above).

Running AltSearch

To launch AltSearch, click on the  button in the Writer's toolbar.

Running AltSearch in Batch mode

Batch mode enables saving and loading of preset search and replace parameters. You can save several *search-and-replace* operations, arrange them in a sequence and then quickly load and execute the whole set.

You can set all parameters using the **Save Batch** button. In the dialog that is then shown, you will be offered the name used for the last batch, which can be renamed. If you enter a name that already exists, you can choose whether the old content will be overwritten or whether it will be preserved and new content added onto the end. At the same time, the command "ReplaceAll" will automatically be saved, with which the batch will be subsequently executed. This command can later be changed by manually editing the batch rule file.

The **Batch >>** button the **Batch manager** dialog where you can run and edit batches. To return back to the search dialog, use the [**<< Searching**] button.

All batch parameters are saved to the text file `AltSearchScript.txt` into the user's directory `.../OpenOffice.org2/user/config/`, and you can open and edit it using the [**Edit**] button in the **Batch manager** dialog. For editing the text, the program **notepad** is used by default, but you can set it to use any other text editor by editing the file `AltSearchEditor.ini` in the same directory. After manual changing and saving the file using the batch manager you can then refresh the list of batch names using the [**Refresh**] button. The syntax used in the file `AltSearchScript.txt` is described at the beginning of the same file, using UTF-8 encoding (from v1.1.1).

When you double-click on an item in the list, or click the [**Execute**] button, the chosen sequence will be loaded and the search and replace operations will be executed. When using batches on selections I advise leaving 1-2 empty paragraphs at the beginning and the end of the selection.

The **Transfer** button is used for transferring the parameters for searching, replacing and setting to the search dialog without executing them. If the batch contains a sequence of several searches and replacements, only the last part of the sequence will be transferred.

The **Key shortcut** button opens a dialog that allows you to assign a keyboard shortcut to an existing batch. To use this:

First select the name of batch from the drop-down menu box

Second, if required, change the name of the auxiliary subroutine for OOo Basic

Third set the desired keyboard shortcut

Finally press the button [**Assign**]

In order for the shortcut to function, at the time of assignment an auxiliary procedure is created in the Basic module `Standard.AltSearchBatches` with a name that is adjusted according to Basic syntax. This name is displayed in the second drop-down menu box of the dialog. When this auxiliary procedure is run, the AltSearch dialog will be opened and immediately the specified batch will be executed. Correct functioning depends on the compliance of the batch name listed inside the procedure and the name of existing batch. If you change the name of batch to which a shortcut key was previously assigned, you will need to re-assign a key shortcut (the old auxiliary procedure to the original name can be deleted by selecting it in the second drop-down box and using the side button [**X**]). Any keyboard shortcut that is used in OOo writer can be released using the lower

button `[x]`. So be careful not to inadvertently remove an important shortcut.

Limitations:

If limitations are known, they are mostly mentioned near of the description of individual parameters. Generally applicable limitations:

- If the option "Current selection only" is active, pieces of text that are inside frames or tables will not be found, even if they are inside selected blocks. Searching inside blocks is limited to only the same text area as the selected block. Multiple selection of blocks isn't supported.
- The function "Find all" will fail to select the paragraph mark `\p`, because the property "Highlighting" (the character's background colour) is used for selection, and you cannot use this to highlight a paragraph mark. It follows that, in addition, this function is limited to texts in which highlighting isn't used. If the document contains highlighting, a warning dialog will appear when "Find all" is used.
- With replacement with more complicated expressions, the function "Undo" is fragmented into partial replacement steps, so that it can easily happen that the number of undo steps needed to restore the document to its original state will not match the expected number.

Customizing LibreOffice for frequent use of AltSearch

if you use AltSearch frequently, it is recommended to customize LibreOffice, by following these steps:

1. In Writer, use the menu **Tools - Customize...**
2. In the window that pops up, select the **Keyboard** tab
3. In the **Category** field, navigate to **OpenOffice.org Macros - User - AltSearch - AltSearch**

3. In the **Function** field, select and assign the following (suggested) shortcuts using the **Modify** button:

_AltSearch: Ctrl+H (to open the Search dialog);

_FindNext: Ctrl+L (to find the next occurrence of the search string after the cursor point, without opening the search dialog);

_FindBack: Ctrl+Shift+L (to find the occurrence of the search string before the cursor point, without opening the search dialog).

[v1.2] From version 1.2 onwards you also can assign shortcuts directly from the AltSearch dialog - see [Batch mode](#)

Limitations of the find-and-replace process:

1. When using the **Find** button, you will find the next text frame only if the frame is selected or if the cursor is inside the frame. If the cursor is a long way away in the text, the first text frame from the internal list of frames in the document is found. **@@@Note clear: Please explain**

The **Current selection only** option doesn't work currently.

2. As a consequence of the problem above, the practical usability of the **Replace** button is very limited.
3. The order of searching matches the order in which the text frames have been inserted into the document and **not** the order within the pages of the document from the start to the end.
4. You can search only for a substring in the name of frame - you cannot use full regular expressions. **@@@what is a "frame"?**
5. The regular expression syntax used in this addon is not fully compatible with OOo.

There are problems especially with searching when using the wildcard *****, **+**, **?** or **{n,n}** just after subexpressions determined by parentheses ().

For example, **(Mi) ?ster** will not be found (however, when using **Count** button the true count will be returned - this function works only when using the compatible mode).

@@@explain what is "compatible mode"

Further, subexpression of the type **(.*) any** or **(.+) any** are searched for, the shortest matching occurrence is found (In non-Greedy mode, contrary to the OOo standard search, which will find the longest matching occurrence (in Greedy mode)).

6. If it is necessary to preserve compatibility, you can delimit the whole search expression with an extra pair of parentheses: **((Mi) ?ster)**. But this will lose the capability to cite the subexpression in the replace expression as a reference, i.e. **\#** where # is a reference number (max. 9) of the subexpression.
7. It is also not possible to use a reference on the subexpression (determined by parentheses **()**) in the *search* expression at the same time as in the *replace* expression. See also [subexpressions](#). **@@@explain: How can back-references be used in search field?**

The **Find all** and **Replace all** functions are fully functional, including the **Current selection only** option. **@@@explain**

Appendix: Regular Expressions (RegEx)

The RegEx can match the literal text that you enter, just like the normal search function. RegEx expressions are always case-sensitive. Thus **cat**, **Cat**, **cAt**, **caT**, **cAT** are not equivalent.

But its power lies in specifying the precise search conditions, using special RegEx patterns.

You can also create a RegEx pattern that matches a lot of different text in the document. For example, a RegEx pattern can match any email address. Another pattern can match any type of URL, regardless of its domain (.com, .org, .gov, etc.)

The following sections explain the RegEx syntax. For more explanation, see OOo help [List of Regular Expressions](#).

Characters

To represent-	Use RegEx-	Remarks
Any character (letter/number)	.	c.t matches cat , cbt...czt , cAt , cBt...cZt , c0t...c9t ,
Any character from a set	[characters]	b[ae]t matches with bat or bet , but not bit . Notes: <ol style="list-style-type: none">1. The characters are not to be separated with comma or spaces.2. The actual sequence of the character does not matter ([ae] and [ea] are equivalent)3. If any character is repeated, the duplicates are ignored. ([ae] and [eaeaa] are equivalent)

Repetition of a pattern

RegEx has tokens that specify repetition of a single character or a string. The repetition token acts on the character that *precedes* it.

To represent-	Use RegEx-	Remarks
Repetition by 0 or more times	*	ab*c matches ac , abc , abbc , abbbc , etc.
Repetition by 0 or 1 times	?	colou?r matches both color and colour
Repetition for one or more times (at least once)	+	ab+c matches abc , abbc , abbbc , etc, but not ac . (b has to occur at least once).

To represent-	Use RegEx-	Remarks
Repetition between a min and a max number	{min,max}	<p>p{2,3} means pp or ppp; but not p, pppp, etc.</p> <p>You can omit either of the numbers.</p> <p>For example, {,3} means <i>repeat ≤ 3 times</i> and {2,} means <i>repeat ≥ 2 times</i>.</p>

The Escape character \

The character **** is called *escape character*.

It reverses the usual meaning of the character that follows.

Let us understand this with two examples:

- A period **.** represents a single character in RegEx. But **\.** means a period (literally).
- The character **t** (without the ****) means just the literal character **t**, but **\t** means a tab.

Here are some important RegEx patterns:

To represent-	Use RegEx-	Remarks
Any letter (capital/small)	\l	Same as [[:alpha:]]{1,1}
Any decimal digit	\d	Same as [0-9]
Tab	\t	
Manual line break	\n	n stands for <i>newline</i>
Manual column break	\c	Limitation: Slow when used separately.
Manual page break	\m	<p>Limitations:</p> <p>Slow when used separately.</p> <p>If subexpression () is used, the parameter \m must be at the <u>start</u> of the search string, and it must not be alone: \m(. . .) but not (. . .) \m.</p>
White space	\s	Same as [\xA0\x9\xA]
Non-breaking space	\S	\x00A0 or \#160
Custom hyphens	\x00AD	

To represent-	Use RegEx-	Remarks
Non-breaking dash	\x2011	
A space inserted by decimal code	\#65	
Dot (period)	\.	
Parentheses ()	\()	
Square brackets []	\[]	

Some useful ReEx examples are as follows:

Expression	Function	Remarks
\p{1,}	Will find the next end of paragraph followed by any number of empty paragraphs.	Same as \p* .
\p{2,4}	Will find the next end-of-paragraph, followed by at least one and at most three <i>consecutive</i> empty paragraphs.	Limitations: Slow when used separately. Sometimes there are problems when searching backwards.
\xhhhh	A character's code as a hexadecimal number (hhhh)	
\#dddd	A character's code as a decimal number (dddd).	If the next character is a digit, it is necessary to fill in zeros from the left (pad with zeros to make up five digits). Otherwise it is not necessary to keep all 5 places of dddd.

Place markers

RegEx syntax also contains some position-markers, which you can embed in the RegEx patterns:

To represent-	Use RegEx-	Remarks
Beginning of a paragraph	^	^This searches for the word This , but <i>only if</i> it occurs at the beginning of a paragraph.
End of a paragraph	\$	here\. will search for the string here. only if it occurs at the end of a paragraph.
	\p	Similar to \$ (see above)

To represent-	Use RegEx-	Remarks
Beginning of a word	\<	\<father will match father but not grandfather or godfather.
End of a word	\>	father\> will match grandfather or godfather but not fatherhood.

You can combine the markers in a single expression. For example, ^\$ represents an empty paragraph, because ^ is the position-marker for the beginning of a paragraph, and \$ is the position-marker for the end of a paragraph. So this combination (without any content in-between) represents an empty paragraph.

Credits:

Author: [Tomas Bilek](#) – © 2007-2008

Licence: LGPL, see http://www.volny.cz/macrojtb/0gnu-lgpl_en.html

Disclaimer:

This macro is distributed WITHOUT ANY WARRANTY. Use at your own risk!