Braille2000, LLC
6801 Southfork Cir, Lincoln, NE 68516
Phone: (402) 423-4782, Fax: (402) 423-5154
http://www.braille2000.com

Applies to version 2.30 and beyond

All About Find and Replace

The Find/Replace mechanism of Braille2000 supports several related manipulations of prose based on content, in the nature of first matching some designated content (“find”) and then changing it (“replace”). Of course you can merely Find something without changing it and by doing “Find All” you can generate a selection (such as the highlight) showing all the locations in the document that match the “find” prose (and by having the selection, you can do things to all its elements, such as change their format or change their typeform).

Find/Replace and its other actions can operate in both braille view and in print view. The actions are appropriately different: in print view, it adjusts the print text with associated changes to the braille as determined by the active braille code; in braille view, it adjusts cells with associated changes to the print interpretation as determined by the active braille code. There are thus some Find/Replace actions that are best suited for one view versus the other.

The basic capabilities of Find/Replace involve the use of its five dialog-box tabs:

Find	Ctrl+f	You provide a “find string” (braille or print) that the mechanism is to try to locate in the document. An alternative behavior (see below) happens when the “find string” is empty.
Replace	Ctrl+r	You provide a “replace string” (braille or print, but the same mode as the “find string”) that the mechanism will substitute for the prose of the “find string”. If the “replace string” is empty, the prose of the “find string” is deleted (replaced by nothing).
Insert Before		You provide an “insert string” (braille or print, but the same mode as the “find string”) that the mechanism will insert into the document just before the location of the “find string”. If the “insert string” is empty, nothing happens (useless).
Insert After		You provide an “insert string” (braille or print, but the same mode as the “find string”) that the mechanism will insert into the document just after the location of the “find string”. If the “insert string” is empty, nothing happens (useless).
Extend		You provide an “extend string” (braille or print, but the same mode as the “find string”) that the mechanism will search for (like find) immediately after the location of the “find string” and if the “extend string” is found (in the same paragraph), the selected string is extended to cover from the beginning* of the “find string” to the end* of the “extend string”, i.e., you can locate a word or phrase by giving its beginning (“find”) and its ending (“extend”) prose. The option “Interior” changes the effect (the words above marked with *) so that

the resulting selection covers the prose from just after the “find” string to just before the “extend” string, i.e., it does not cover either the “find” string nor the “extend” string prose themselves.

The above summaries are meant to explain that the five tabs of the Find/Replace dialog box work together (as needed) to perform various document modifications. (But note that Replace, Insert After, Insert Before, Extend do not work with each other, they each work with Find.)

If you merely wish to locate some prose, you use the Find panel alone.

If you want to replace one phrase of prose with a different phrase, you use Find and Replace tabs, putting the “find string” on the Find tab, then switching to the Replace tab and putting in the “replace string” and then using its buttons to cause the operation to be performed.

A similar scenario applies to Find/Insert After and Find/Insert Before and Find/Extend.

If the “find string” is empty, the above behaviors are adjusted somewhat (the “alternative behavior” referenced above). An empty “find string” is useful only when there is a multi-element selection (such as a multi-element highlight or the use of a multi-element named selection; see the write-up on “*Selection Manipulation in Braille2000*”). For this discussion, think of a highlight with several disjoint highlighted passages (each is an “element”). When the “find string” is empty, it “matches” the elements of the selection one whole element at a time. For example if you do Find from the Top, it will highlight the first element (all of it). If you then do Next, it will highlight the second element (all of it)… you can work your way through all of the elements. If the “find string” is empty when you use Replace or Insert After or Insert Before or Extend, it applies to a whole selection element, one element at a time… the behaviors are similar to normal behavior but apply differently. If Find/Replace is not being used in a multi-element-selection way, an empty “find string” is an error.

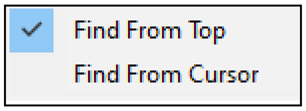
If you highlight a word or short phrase before using Ctrl+f or displaying the Find dialog box, the highlighted word or phrase will be copied into the "find-string" box; if you don't want to search for that string, just over-type (or braille) a replacement.

If you want to repeat a find (or replace) that you did previously, pressing the up-arrow key will load the "find-string" (and separately the "replace-string") with previously-used data.

Find From…

Each panel (Find/Replace/Insert Before/Insert After/Expand) has "from" buttons (e.g., "Find from…") that can initiate the action starting at the Top of the file, or at the current Cursor position, or just after the last-found item (Next). There is also an "all" button (e.g., Find All) that causes all item to be found or processed all at once. After adjusting the setting, you should click one of these buttons (i.e., Top, Cursor, Next, or All).

One of Top/Cursor/Next is the default button: the button that operates if you press the Enter key.



You can choose which button (Top or Cursor) will be the default button: point the mouse at the buttons sub-panel (say at the Cursor button) then click the right-hand mouse button. A popup menu will appear. Choose which button is to be the default.

Note that after any successful use of Find/Replace, the subsequent time the Find/Replace dialog box appears it will be configured to do "Next", i.e., the Next button will be the default and the various strings will be as they were before. You click Next to iterate to the next item. (The default button setting discussed above does not operate during "Next" processing. But if you change the string (to do a different Find/Replace), the Next button is grayed and Top or Cursor will become the default button as per your setting.

Remember that Ctrl+n can be used in lieu of opening the Find/Replace dialog and clicking Next.

The basic behaviors presented above can be embellished in several ways that are described below. Some of the additional mechanisms can be complicated… you may seldom use them… you may choose to ignore what follows.

Iterations of the Find/Replace Process

It may very well be that you don't want to Find just one matching item of prose, but many or all of them, or that you don't want to Replace the prose in just one spot, but at many or all such spots in all or a selected part of the document. For those situations, you use Find/Replace iteratively.

Ctrl+n
After a successful Find/Replace/InsertAfter/InsertBefore/Extend operation, pressing Ctrl+n will repeat the operation for the next position in the document. It is the same as clicking the Next button.

All
Each tab panel offers an "All" button (Find All, Replace All, Insert All, Extend All) you can use to exhaustively repeat the operation throughout the designated selection. In the case of Find All, it generates a (potentially multi-element) highlight showing all the matching prose throughout the document. This highlight may be useful for doing various selection-based manipulations (see *Selection Manipulation in Braille2000*, a separate write-up).

Y/N
The "Y/N" checkbox works with the "All" button. When Y/N is active, the above behavior does not happen. Rather, the Find/Replace mechanism operates from the Top of the file, doing the designated operation and then pausing with the "Y N Esc" overlay control showing in the extreme upper-left corner of the Braille2000 window. You can click one of the choices or press the y or n or esc keys. The document on the screen will show the result of the operation (highlighted prose with replace/insert/extend having been performed as directed). Clicking Y or pressing the y key accepts the operation and Find/Replace iterates to the next situation and the cycle repeats. Clicking N or pressing the n key rejects the operation (the document is not actually changed) and Find/Replace iterates to the next situation and the cycle repeats. Clicking Esc or pressing the esc key terminates Find/Replace cycles.

Search In

The Find/Replace operations operate in a designated part of the document as specified by the choice for "Search In". Those choices are different "selections" (see *Selection Manipulation in Braille2000*, a separate write-up). You are probably familiar with the Highlight (one of various selections available in Braille2000). Find/Replace often operates on the "Whole File." selection, meaning that it considers the document from top to bottom. By choosing a different selection via "Search In" you can constrain Find/Replace to consider only prose within the elements of that selection.

When there exists a multi-element selection whose elements need to be manipulated, as a whole, element-by-element, by Find/Replace, you leave the "find string" empty and then Replace, InsertBefore, InsertAfter, and Extend operate on each whole selection element. You can use Y/N and the "all" button in this mode, if you wish.

Choosing the String Mode and Code

The "find string" can be input as ASCII, Braille, or Print prose. The initial mode of the "find string" box is the mode of the document (if the document mode is "S" the "find string" mode will be "P"). The modes are "A" (ASCII), "B" (Braille), and "P" (Print) with a button for each. You can change the mode at any time. When changing between print and non-print, the content of the "find string" is translated using the designated code choice. The code is initially that of the document being edited (the code at the location of the cursor when Find/Replace is displayed). By clicking the code button (to the right of the A B P mode buttons) you can change the code used for adjusting the “find string”.

The “replace string”, “insert string”, “extend string” modes and codes cannot be individually adjusted: they will be the same mode and code as the “find string”.

Metacharacters

You may have discovered that Find/Replace in MS Word has what are called “wildcard characters” that can designate things you cannot type explicitly, such as “tab” or “line break”. A similar but different concept has existed for decades in the UNIX system, called “regular expressions”. (“Regular” refers to a class of formal grammars studied by computer scientists in the discipline of automata theory.) This kind of extended behavior, in Braille2000, is called “Metacharacters”. This feature was first available in version 2.30. It is most useful for doing Find/Replace on unwanted tab characters.

To replace each tab character with a space
Press Ctrl+f (to get the Find/Replace dialog box); Press Ctrl+T (to get the tab metacharacter symbol $\boxed{T}$); Click the Replace tab and enter one space in the “replace string”; Click Replace All.

Presentation
In Find/Replace, metacharacters are special symbols that show as a symbol in a box, such as $\boxed{+}$ or $\boxed{b}$. (Both MS Word and UNIX use normal Roman characters for the special operators… the unusual notation used by Braille2000 avoids all that confusion!)

Metacharacter symbols are input to the "find-string" or the "replace-string" either via the Metacharacter Menu button, or by pressing the Ctrl key (e.g., Ctrl+R enters the $\boxed{R}$ metacharacter).

In Find/Replace the button “Metacharacter Menu” will show a popup menu of metacharacter choices. You can also type a metacharacter by holding down Ctrl… the two metacharacters shown above can be input as Ctrl++ and Ctrl+b, respectively. Note that there is syntax to metacharacters: some patterns of use are illegal and will produce an error statement in the Find/Replace dialog box.

There are a couple dozen available metacharacters to achieve a variety of behaviors. They are all defined further below. But before that there is a discussion of the kinds of behaviors metacharacters can achieve (most behaviors relate to searching for specific prose using Find).

Metacharacters for categorical search
Suppose you want to find a multi-digit number… any number. You can’t search for “123” and expect it to find “123” as well as “49”. But there is a metacharacter ($\boxed{\#}$) for the category of “a digit” and using that you can find all the digits. (How to find a multi-digit number will be explained below.)

The following categorical metacodes are available. Each will match *one* cell or *one* character, depending on the mode of the “find string”. To match a sequence of cells or characters by category, use multiple codes or follow a code with an iterator (see below).

Categorical codes

$\boxed{A}$	Alphanumeric
$\boxed{C}$	Contraction
$\boxed{D}$	Upper digit
$\boxed{H}$	Hyphen
$\boxed{I}$	Indicator
$\boxed{L}$	Lowercase letter
$\boxed{N}$	Lower digit
$\boxed{P}$	Punctuation
$\boxed{R}$	Letter

$\boxed{T}$	Tab
$\boxed{U}$	Uppercase letter
$\boxed{W}$	Whitespace
$\boxed{Y}$	Symbol
$\boxed{.}$	Any character
$\boxed{\#}$	Digit

The Metacharacter "enumeration" code
Some items of prose are composed of a mix of cells/characters from a given subset. For example if you wanted to find "up" or "down" indicators for super/sub-script you would search for 5 or 9. The metacharacter operator $\boxed{[}$ 5 9 $\boxed{]}$ will match a single cell that is specified in the enumeration (i.e., a cell or character identical to one of the cells/characters inside $\boxed{[}\,\boxed{]}$. (The order of symbols inside the enumeration does not matter.) As another example the enumeration $\boxed{[}$9876543210$\boxed{]}$ is the same match as $\boxed{\#}$ (in print they are the same; in braille $\boxed{\#}$ matches cells that express digits, which might well be A-J). If you wanted to match a vowel you could use $\boxed{[}$aeiou$\boxed{]}$. If you wanted to match a sequence of vowels you could use $\boxed{[}$aeiou$\boxed{]}\,\boxed{+}$.

Metacharacters for text attributes
Suppose you want to find the word "Must" when it appears as bold. "Bold" is a text attribute, along with other typeforms and "G1" and "number". (Think about it, G could be a number or a letter, depending on different attributes.)

The following attribute metacodes are available. Each will constrain subsequent matching to prose with the designated attribute(s). These codes do not match anything; they affect what follows them.

Attribute codes

$\boxed{a}$	Capital
$\boxed{b}$	Bold
$\boxed{d}$	Indicator
$\boxed{g}$	Grade 1
$\boxed{i}$	Italics
$\boxed{n}$	Number
$\boxed{p}$	Punctuation
$\boxed{r}$	Letter
$\boxed{s}$	Script
$\boxed{u}$	Underlined
$\boxed{y}$	Symbol

Note that ctrl+c, ctrl+v, ctrl+x continue to do Copy, Paste, Cut, respectively, in the find-string and the replace-string.

Attribute codes operate on the match tokens that follow them, to the end of the containing group. You can use more than one, for example to search for bold italics via $\boxed{b}\,\boxed{i}$. Note that attribute codes are lower-case, while categorical codes (the above group) are upper-case. The difference is that categorical codes each match a single cell or character (of the given type) where attribute codes constrain the matching of subsequent codes, i.e., attribute codes are no good used alone.

Metacharacters for within-string iteration
Some forms in prose involve repeated symbols, such as guide dots in Contents, (3,6) cells in page turn indicator lines, and separator lines. Most of the time those kinds of things will be automated (Find/Replace does not operate on yellow generated prose), but there may be situations in which character or phrase repetition is present.

In Metacharacter notation, the general iteration operator is $\boxed{\{}$2,5$\boxed{\}}$ where the first number (e.g., 2) is the minimum number of repetitions, and the second number (e.g., 5) is the maximum number of repetitions, of the match token that immediately precedes this notation. If you want a particular number of matches, use just one number such as $\boxed{\{}$4$\boxed{\}}$. (a$\boxed{\{}$4$\boxed{\}}$ is the same as aaaa.) The notation $\boxed{\{}$0$\boxed{\}}$ is illegal.

For example the “find-string” of abc{2,5} is going to match the character sequence a, b, and two to five adjacent c’s, such as abcc abccc abcccc abccccc. If the document contained abccccccccccc this would match on the first seven characters only. (If you wanted the find-string to match on repetitions of “abc” as a whole, you would need to put “abc” inside a group and immediately follow the group with the iteration code; see below.) If you want as many matches as possible, omit the second number. For 3 or more matches, write {3,}. There are several commonly used iteration amounts that have their own symbols. They are listed here. (They are much the same as those proposed by Stephen Kleene in the 1930's and still used in regular expression notation in UNIX.)

Iteration codes

?	Zero or one (same as {0,1})
*	Zero or more (same as {0,}
+	One or more (same as {1,})
{2,5}	Two to five (for any minimum/maximum count specified)

For example, if you wanted to search for a top boxing line (a number of 7’s) you could use the “find string” 7+ (one or more 7 ‘s).

Not
The Not code ~ can be placed before any plain cell or character or before a category code or before a group code to signal that the following item must not occur. For example if you wanted to search for a character other than a vowel you could use ~[aeiou]. The meaning of not is complex and depends on what is being negated.

~x	This find-string matches any single cell or character that is not x.
~C	This find-string matches any single cell or character that is not a contraction.
~[ ]	This find-string matches any single cell or character that is not in the enumeration.
~$	This find-string is satisfied if the end of the paragraph has not been reached.

Elements that match a single cell or character, when negated by the "not" operator, continue to match a single cell or character, but of the opposite nature.

~(abc)	This find-string matches any sequence of letters other than abc.
~x+	This find-string matches any sequence of letters other than x's.
~1	This find-string matches any sequence of letters other than those referenced.

When "not" is applied to a group or to a varying-length match token (such as x+), it will match as much prose as possible without matching what it negates.

Metacharacters for end effects
The metacharacter ^ matches the zero-width virtual “begin paragraph” mark at the beginning of every paragraph. For example ^hello will match “hello” only when it begins a paragraph. Similarly the metacharacter $ matches the zero-width virtual “end paragraph” mark at the end of each paragraph.

Metacharacter groups
Segments of the “find string” can be grouped together, for example (from above) to search for “abc” repeated a number of times. The metacharacters ( ) enclose a sub-expression group, such as (abc). If you wanted to search for “abc” “abcabc” “abcabcabc” etc. you would write (abc)+.

As mentioned above, metacharacter attribute codes affect the tokens that follow, to the end of the enclosing group. (The entire “find string” acts as a main group; it may have one or more sub-group (sub-expression) components.)

A group can play an optional special role, when desired. If a group begins with @ it becomes the (one and only) "focus" of the search. If there is a focus group, matching that occurs before or after the focus-group is considered "context" and not the result of the match. For example if you used the "find-string" up (@and) down, it would match within "He ran up and down the stairs." with just "and" being the result, and if your "replace-string" were "&" then occurrences of "up and down" would be changed to "up & down". The point is that only the prose matching the focus group would be changed (or highlighted or appended to or extended, etc.). (Note that this kind of replacement can also be done using references but using the concept of focus is much simpler.)

A group may also be “marked”, if it begins with the metacharacter “mark” symbol : . A group need not be marked. If it is marked, then the prose that it matches can be referred-to via a “reference” token. There are nine reference tokens: 1 2 3 4 5 6 7 8 9 . They each refer to the prose matched by a marked group, working left-to-right, mark by mark, through the “find string”. If the “find string” contains two “mark” metacharacters, then reference tokens 1 2 (only) are defined thereby.

(Examples for using reference tokens are somewhat esoteric.)

Suppose you wanted to find “girl and boy” (as a phrase) as well as “girlfriend and boyfriend” as well as “girl’s and boy’s”, where the ending on both girl and boy were the same. You could do such a Find operation using girl(: R *) and boy1 where “girl” followed by zero more letters, followed by “ and boy” is followed by the same prose that matched the group containing R * , i.e., those same zero or more letters that immediately followed “girl” and immediately preceded “and”.

Metacharacter reference tokens in “replace string”
When the “find string” contains metacharacter mark symbols, the replace/insert/extend strings may use a thus-defined reference token. This capability, among other neat things, can be used to reorder the prose in a phrase(!).

Suppose you have a list of lastname, firstname (paragraph by paragraph), but you want it to read firstname lastname instead. You would set your “find string” to (: R +) , (: R +) namely one or more letters, comma, space, one or more letters, in two marked groups. You would then set your “replace string” to 2 1 that causes the “find string” to get replaced with the matched prose from group 2 (the firstname), a space, and the matched prose from group 1 (the lastname).

Neat Things You Can Do With Find/Replace

Here are several examples of the things you can do with Find/Replace

Replace "this" with "that"
Find: this Replace: that

Replace one or more tabs (in sequence) with a single space
Find: $\boxed{T}$ $\boxed{+}$ Replace: (enter one space)

Replace multiple blanks with a single space
Find: $\boxed{+}$ (a space and then $\boxed{+}$) Replace: (enter one space)

Find all bold words
Find: $\boxed{b}$ $\boxed{R}$ $\boxed{+}$ click Find All

Change all bold words to italics
Use “Find all bold words” above. This highlights all bold words. Then enter Ctrl+b to turn off bold, and then enter Ctrl+i to turn on italics.

In a transcription with embedded Nemeth material (phrases where the code is set to “Nemeth”), visit each such passage, one by one, throughout the manuscript
Use Find. Leave the “find string” empty. Set "Search In" to “Code #2 Nem,Con”. Put a checkmark in “Visit each element”. Put a checkmark in “Y/N”. Click Find All. Note: the "Search In" choices will have a selection choice for each code used in the document, as well as "Whole File." and perhaps "Highlight." and "Suspect Words.". You can supply additional selections of your own making using Named Selections (see “*Selection Manipulation in Braille2000*”, a separate write-up).