**putdocx findreplace** — Find and replace text in an Office Open XML (.docx) file[+]

[+]This command is part of StataNow.

Description Quick start Syntax Options
Remarks and examples Also see

## Description

`putdocx findreplace set` specifies the settings for searching and replacing text; this includes the text to be matched, the text used to replace the matched text, the document in which to find and replace text, and the font style. Configurations set by this command will be saved in memory and can be queried.

`putdocx findreplace replace` conducts the replacing operation. This can be used as a standalone command for a simple find-and-replace operation. This command can also apply the settings specified with `putdocx findreplace set` and temporarily modify those settings for the current operation.

`putdocx findreplace reset` resets some or all find-and-replace configurations.

`putdocx findreplace query` displays the saved configurations.

## Quick start

Find instances of `a` in file `src.docx`, replace with `b`, and save the changes in file `dest.docx`

```
putdocx findreplace replace, docx(src.docx) from(`"a"') to(`"b"') ///
    saving(dest.docx) replace
```

Specify font color and scope settings for find and replace

```
putdocx findreplace set, fontcolor(red) underline
putdocx findreplace set, scope(1, header body)
putdocx findreplace set, docx(src.docx) from(`"a"')
```

Find and replace text using the settings specified above

```
putdocx findreplace replace, to(`"b"') ///
    inherit saving(dest.docx)
```

Query find-and-replace configurations specified with `putdocx findreplace set`

```
putdocx findreplace query
```

Reset the font color and scope, previously specified with `putdocx findreplace set`, to their defaults

```
putdocx findreplace reset, fontcolor scope
```

Reset all saved configurations

```
putdocx findreplace reset, all
```

# Syntax

*Set up the find-and-replace configurations*

```
putdocx findreplace set [ , set_options ]
```

*Conduct the replace operation*

```
putdocx findreplace replace [ , replace_options set_options ]
```

*Reset the current saved configurations*

```
putdocx findreplace reset [ , reset_options ]
```

*Query the current saved configurations*

```
putdocx findreplace query
```

| *set_options* | Description |
|---|---|
| `docx(`*filename*`)` | specify the source document in which to find text to be replaced |
| `from(`` `"`*text*`"' `` | set text to be replaced |
| `to(`` `"`*text*`"' `` | set new text to replace old text |
| `scope(`*section*`[ , `*story1*` [`*story2*` [...]]])` | set scope of the document in which to replace text |
| `[no]matchwholeword` | specify whether to match whole words; default is `nomatchwholeword` |
| `[no]matchcase` | specify whether to match case; default is `nomatchcase` |
| `[no]regex` | specify whether to allow regular expressions in `from()` and `to()`; default is `noregex` |
| `occurrences(`#`)` | number of occurrences to replace |
| `backward(on\|off)` | find and replace starting from the end of the document; default is `backward(off)` |
| `fontname(`*fontname*`)` | set font name for replacement text |
| `fontcolor(`*color*`)` | set font color for replacement text |
| `fontsize(`#`)` | set font size for replacement text |
| `shading(`*sspec*`)` | set background color, foreground color, and fill pattern for replacement text |
| `bold(on\|off\|default)` | specify whether to bold text, remove bold formatting, or retain original formatting |
| `italic(on\|off\|default)` | specify whether to italicize text, remove italics, or retain original formatting |
| `underline[ (`*upattern*`) ]` | underline replacement text using specified pattern |
| `underlinecolor(`*color*`)` | set underline color for replacement text |
| `strikeout[ (`*strikespec*`) ]` | specify whether to add a strikeout mark to text or retain original formatting |

*fontname* may be any valid font installed on the user's computer. If *fontname* includes spaces, then it must be enclosed in double quotes.

*color* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`.

*sspec* is

*bgcolor*[ , *fgcolor*[ , *fpattern* ] ]

*bgcolor* specifies the background color.

*fgcolor* specifies the foreground color. The default foreground color is `black`.

*fpattern* specifies the fill pattern. The most common fill patterns are `solid` for a solid color (determined by *fgcolor*), `pct25` for 25% gray scale, `pct50` for 50% gray scale, and `pct75` for 75% gray scale. A complete list of fill patterns is shown in *Shading patterns* of [RPT] **Appendix for putdocx**.

*upattern* may be any of the patterns listed in *Underline patterns* of [RPT] **Appendix for putdocx**. The most common underline patterns are `double`, `dash`, and `none`.

| *replace_options* | Description |
|---|---|
| `saving(`*filename*`)` | specify the target file to be saved |
| `replace` | replace the target file |
| `inherit` | apply the saved find-and-replace configurations |

| *reset_options* | Description |
|---|---|
| `docx` | clear the *filename* |
| `from` | clear the matching text |
| `to` | clear the replacement text |
| `scope` | clear the replacing scope |
| `matchwholeword` | disable whole-word-only matching |
| `matchcase` | disable case sensitivity |
| `regex` | disable regular expressions |
| `occurrences` | reset the number of occurrences to all occurrences |
| `backward` | turn backward search off |
| `fontname` | reset the *fontname* to `default` |
| `fontcolor` | reset the *fontcolor* to `default` |
| `fontsize` | reset the *fontsize* to `default` |
| `shading` | reset the background color, foreground color, and fill pattern to `default` |
| `bold` | change the bold setting to `default` |
| `italic` | change the italic setting to `default` |
| `underline` | change the underline setting to `default` |
| `underlinecolor` | change the underline *color* to `default` |
| `strikeout` | change the strikeout setting to `default` |
| `all` | clear all saved find-and-replace configurations |

# Options

Options are presented under the following headings:

*Options for putdocx findreplace set*
*Options for putdocx findreplace replace*
*Options for putdocx findreplace reset*

## Options for putdocx findreplace set

`docx(`*filename*`)` specifies the source file where to perform find and replace. If you do not specify the source file with `docx()`, `$ N/A $` will be displayed with `putdocx findreplace query`. To perform find and replace, you must specify this option with either `putdocx findreplace set` or `putdocx findreplace replace`.

`from(`*text*`)` specifies the matching text. The text must be enclosed in compound quotes (`` `" `` and `"'`). If nothing is specified, `$ N/A $` will be displayed with `putdocx findreplace query`. To perform find and replace, you must specify this option with either `putdocx findreplace set` or `putdocx findreplace replace`.

`to(`*text*`)` specifies the replacement text. The text must be enclosed in compound quotes (`` `" `` and `"'`). If nothing is specified, `$ N/A $` will be displayed with `putdocx findreplace query`. To perform find and replace, you must specify this option with either `putdocx findreplace set` or `putdocx findreplace replace`.

`scope(`*section*[ `,` *story1* [*story2* [ ... ]]]`)` sets the scope of the document in which to find and replace text. If the specified scope does not match any parts of the document, an error will occur showing the specified scope is not applicable. The default is `scope(-1, all)`, meaning find and replace is performed throughout the whole document. `scope(default)` is a synonym for `scope(-1, all)`.

*section* specifies the section number and should be within the range of the total number of sections in the document specified in `docx()`. By default, the section number is `-1`, which refers to all sections in the document, in other words, the whole document. `scope(1)` sets the scope to the first section.

*story* specifies the desired story, and multiple stories may be specified. Acceptable stories include `all`, `body`, `header`, `header_odd`, `header_even`, `header_first`, `header_after_first`, `footer`, `footer_odd`, `footer_even`, `footer_first`, and `footer_after_first`.

By default, *story* is `all`. If *section* is specified, while *story* is not, *story* defaults to `all`. If *story* is specified, while *section* is not, `section` defaults to `-1`. Hence, `scope(2)` is equivalent to `scope(2, all)`; `scope(,body)` is equivalent to `scope(-1, body)`. You can specify multiple stories, such as `scope(-1, body header footer_even)`.

If you specify a story that has the suffix `"_odd"`, `"_even"`, `"_first"`, or `"_after_first"`, it needs to be consistent with the Word document page setup (“Different First Page” and “Different Odd & Even Pages”). If the original Word document does not have “Different First Page” enabled, the stories with a suffix including `"_first"` and `"_after_first"` will be ignored. If the original Word document does not have “Different Odd & Even Pages” enabled, the stories with suffix `"_odd"` and `"_even"` will be ignored. Below, we provide details on how different stories behave with different Word document settings.

| Different First Page | Different Odd & Even Pages | Story | Description |
|---|---|---|---|
| Disabled | Disabled | `header` | all headers |
| | | `footer` | all footers |
| Disabled | Enabled | `header` | all headers |
| | | `header_odd` | only odd headers |
| | | `header_even` | only even headers |
| | | `footer` | all footers |
| | | `footer_odd` | only odd footers |
| | | `footer_even` | only even footers |
| Enabled | Disabled | `header` | all headers |
| | | `header_first` | only header on the first page |
| | | `header_after_first` | all headers, excluding the first page |
| | | `footer` | all footers |
| | | `footer_first` | only footer on the first page |
| | | `footer_after_first` | all footers, excluding the first page |
| Enabled | Enabled | `header` | all headers |
| | | `header_odd` | all headers on the odd pages, except for the first page |
| | | `header_even` | all headers on the even pages, except for the first page |
| | | `header_first` | only the header on the first page |
| | | `header_after_first` | all headers, excluding the first page |
| | | `footer` | all footers |
| | | `footer_odd` | all footers on the odd pages, except for the first page |
| | | `footer_even` | all footers on the even pages, except for the first page |
| | | `footer_first` | only the footer on the first page |
| | | `footer_after_first` | all footers, excluding the first page |

`nomatchwholeword` and `matchwholeword` specify whether the finding operation should match only whole words. `nomatchwholeword`, the default, specifies that the finding operation not be restricted to matching only whole words. `matchwholeword` specifies that the finding operation match only whole words. For example, with the default `nomatchwholeword`, `putdocx findreplace set, from(`"eat"') to(`"Eat"')` would replace instances of "seat" with "sEat".

`nomatchcase` and `matchcase` specify whether the finding operation should be case sensitive. `nomatchcase`, the default, specifies that the finding operation be case insensitive. `matchcase` specifies that the finding operation be case sensitive.

`noregex` and `regex` specify whether regular expressions be allowed. `noregex`, the default, disables regular expressions. `regex` specifies that regular expressions be allowed.

`occurrences(#)` specifies the number of occurrences that should be replaced. The default is `occurrences(-1)`, indicating all occurrences. `occurrences(default)` is a synonym for `occurrences(-1)`. # must be greater than or equal to 1. If # exceeds the total number of occurrences,

all available instances will be replaced. While the find-and-replace operation handles the number of occurrences effectively in the body of the document, it does not work as well with headers and footers because these elements link across entire sections instead of operating on a page-by-page basis. If you need to find and replace text for a certain number of occurrences in the header or footer, it is best to write a macro using Visual Basic for Applications (VBA).

Additionally, note that the reported number of matched occurrences may seem slightly off if you are replacing text in the header or footer, but the occurrences will be consistent with what is reported when you search for text directly in Microsoft Word. For example, text that is used in a repeating header, and thus is found across multiple pages, may be counted only as a single occurrence.

`backward(off)` and `backward(on)` specify the direction of the find-and-replace operation. `backward(off)`, the default, specifies that find and replace be performed from the beginning of the file to the end of the file. `backward(on)` specifies that find and replace be performed from the end of the file, working backward, to the beginning of the file. This option can be combined with `occurrences(#)`, for instance, to find and replace text in the last three instances.

`fontname(`*fontname*`)` sets the font name for the replacement text. *fontname* may be any valid font installed on the user's computer. If *fontname* includes spaces, then it must be enclosed in double quotes. If *fontname* is not available, the fallback font will be used. `fontname(default)` specifies that the replacement text use the same font name as the original text.

`fontcolor(`*color*`)` sets the color for the replacement text. *color* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`. `fontcolor(default)` specifies that the replacement text use the same font color as the original text.

`fontsize(#)` sets the font size. # is a numeric value that represents font size measured in points. The value should be greater than or equal to 1. `fontsize(default)` specifies that the replacement text use the same font size as the original text.

`shading(`*bgcolor*`[ , `*fgcolor*`[ , `*fpattern*` ] ])` sets the background color, foreground color, and fill pattern for the replacement text.

*bgcolor* and *fgcolor* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`. `shading(default)` specifies that the replacement text use the same shading properties as the original text.

Although `shading()` specifies three properties, it is treated as a single option when overriding previous specifications. For example, we might specify a blue background color, green foreground color, and horizontal stripes for the fill pattern by typing `putdocx findreplace set, shading(blue, green, horzStripe)`. Later, we might change the background color to red by typing `putdocx findreplace set, shading(red)`; this will also clear out the foreground color and fill pattern we specified before. In other words, results will be the same as if we typed `putdocx findreplace set, shading(red, default, default)`.

`bold(on)`, `bold(off)`, and `bold(default)` specify whether replacement text should be formatted as bold. `bold(on)` specifies that the replacement text be formatted as bold. `bold(off)` specifies that the replacement text not be formatted as bold. `bold(default)` specifies that the replacement text retain the formatting of the original text.

`italic(on)`, `italic(off)`, and `italic(default)` specify whether the replacement text should be formatted as italic. `italic(on)` specifies that the replacement text be formatted as italic. `italic(off)` specifies that the replacement text not be formatted as italic. `italic(default)` specifies that the replacement text retain the formatting of the original text.

`underline`[ `(`*upattern*`)` ] specifies that the replacement text be underlined and optionally specifies the format of the line. `underline(default)` specifies that the replacement text use the same underline pattern as the original text.

*upattern* may be any of the patterns listed in *Underline patterns* of [RPT] **Appendix for putdocx**. The most common underline patterns are `double`, `dash`, and `none`. `underline` is a synonym for `underline(single)`.

`underlinecolor(`*color*`)` specifies the underline color for the replacement text. *color* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`. `underlinecolor(default)` specifies that the replacement text use the same underline color as the original text.

`strikeout`[ `(`*strikespec*`)` ] specifies whether the replacement text should have a strikeout mark. *strikespec* is one of `single`, `double`, `none`, or `default`. `single` adds a single strikeout mark. `double` adds two strikeout marks. `none` removes the strikeout mark. `default` specifies that the replacement text use the same strikeout mark as the original text. `strikeout` is a synonym for `strikeout(single)`.

## Options for putdocx findreplace replace

`saving(`*filename*`)` specifies that the results of find and replace be written to the new document, *filename*. If *filename* is specified without an extension, `.docx` is assumed. If `saving()` is not specified, changes are made to the filename specified with `docx()` but only if `replace` is specified to allow the `.docx` file to be overwritten.

`replace` specifies to overwrite *filename*, if it exists, with the result of find and replace. If `saving()` is specified, the filename specified with `saving()` is overwritten. If `saving()` is not specified, the filename specified with `docx()` is overwritten.

`inherit` specifies to use the saved `putdocx findreplace` configurations, as reported by `putdocx findreplace query`.

`docx(`*filename*`)` specifies the source file where to perform find and replace. If you do not specify the source file with `docx()`, `$ N/A $` will be displayed with `putdocx findreplace query`. To perform find and replace, you must specify this option with either `putdocx findreplace set` or `putdocx findreplace replace`.

`from(`*text*`)` specifies the matching text. The text must be enclosed in compound quotes (`` `" `` and `"' ``). If nothing is specified, `$ N/A $` will be displayed with `putdocx findreplace query`. To perform find and replace, you must specify this option with either `putdocx findreplace set` or `putdocx findreplace replace`.

`to(`*text*`)` specifies the replacement text. The text must be enclosed in compound quotes (`` `" `` and `"' ``). If nothing is specified, `$ N/A $` will be displayed with `putdocx findreplace query`. To perform find and replace, you must specify this option with either `putdocx findreplace set` or `putdocx findreplace replace`.

`scope(`*section*`[ , `*story1*` [`*story2*` [...]]])` sets the scope of the document in which to find and replace text. If the specified scope does not match any parts of the document, an error will occur showing the specified scope is not applicable. The default is `scope(-1, all)`, meaning find and replace is performed throughout the whole document. `scope(default)` is a synonym for `scope(-1, all)`.

*section* specifies the section number and should be within the range of the total number of sections in the document specified in `docx()`. By default, the section number is `-1`, which refers to all sections in the document, in other words, the whole document. `scope(1)` sets the scope to the first section.

*story* specifies the desired story, and multiple stories may be specified. Acceptable stories include `all`, `body`, `header`, `header_odd`, `header_even`, `header_first`, `header_after_first`, `footer`, `footer_odd`, `footer_even`, `footer_first`, and `footer_after_first`. The combination of section number and story allows you to narrow your search from the whole document to a single section and to the header or footer of a section.

By default, *story* is `all`. If *section* is specified, while *story* is not, *story* defaults to `all`. If *story* is specified, while *section* is not, `section` defaults to `-1`. Hence, `scope(2)` is equivalent to `scope(2, all)`; `scope(,body)` is equivalent to `scope(-1, body)`. You can specify multiple stories, such as `scope(-1, body header footer_even)`.

In Microsoft Word, you can enable the "Different First Page" setting, which allows you to have a different header and footer for the first page of a section. This setting is specific to each section, so you might have a document with several sections, some of which have this setting enabled and others that have it disabled. Additionally, in Microsoft Word, you can enable the "Different Odd & Even Pages" setting, which allows you to have a different header and footer for odd and even pages in your document. This setting is applied to the whole document. The behavior of the stories depends on whether these two settings are enabled. For example, `scope(, header_odd)` refers to all headers on odd pages if you have "Different Odd & Even Pages" enabled and "Different First Page" disabled; however, `scope(, header_odd)` refers to all headers on odd pages, except for the first page, if both settings are enabled.

If you specify a story that has the suffix `"_odd"`, `"_even"`, `"_first"`, or `"_after_first"`, it needs to be consistent with the Microsoft Word document page setup ("Different First Page" and "Different Odd & Even Pages"). If the original Microsoft Word document does not have "Different First Page" enabled, the stories with a suffix including `"_first"` and `"_after_first"` will be ignored. If the original Microsoft Word document does not have "Different Odd & Even Pages" enabled, the stories with suffix `"_odd"` and `"_even"` will be ignored. Below, we provide details on how different stories behave with different Microsoft Word document settings.

| Different First Page | Different Odd & Even Page | Story | Description |
|---|---|---|---|
| Disabled | Disabled | `header` | all headers |
| | | `footer` | all footers |
| Disabled | Enabled | `header` | all headers |
| | | `header_odd` | only odd headers |
| | | `header_even` | only even headers |
| | | `footer` | all footers |
| | | `footer_odd` | only odd footers |
| | | `footer_even` | only even footers |
| Enabled | Disabled | `header` | all headers |
| | | `header_first` | only header on the first page |
| | | `header_after_first` | all headers, excluding the first page |
| | | `footer` | all footers |
| | | `footer_first` | only footer on the first page |
| | | `footer_after_first` | all footers, excluding the first page |
| Enabled | Enabled | `header` | all headers |
| | | `header_odd` | all headers on the odd pages, except for the first page |
| | | `header_even` | all headers on the even pages, except for the first page |
| | | `header_first` | only the header on the first page |
| | | `header_after_first` | all headers, excluding the first page |
| | | `footer` | all footers |
| | | `footer_odd` | all footers on the odd pages, except for the first page |
| | | `footer_even` | all footers on the even pages, except for the first page |
| | | `footer_first` | only the footer on the first page |
| | | `footer_after_first` | all footers, excluding the first page |

`nomatchwholeword` and `matchwholeword` specify whether the finding operation should match only whole words. `nomatchwholeword`, the default, specifies that the finding operation not be restricted to matching only whole words. `matchwholeword` specifies that the finding operation match only whole words. For example, with the default `nomatchwholeword`, `putdocx findreplace replace, from(`"eat"') to(`"Eat"')` would replace instances of "seat" with "sEat".

`nomatchcase` and `matchcase` specify whether the finding operation should be case sensitive. `nomatchcase`, the default, specifies that the finding operation be case insensitive. `matchcase` specifies that the finding operation be case sensitive.

`noregex` and `regex` specify whether regular expressions be allowed. `noregex`, the default, disables regular expressions. `regex` specifies that regular expressions be allowed.

`occurrences(#)` specifies the number of occurrences that should be replaced. The default is `occurrences(-1)`, indicating all occurrences. `occurrences(default)` is a synonym for `occurrences(-1)`. # must be greater than or equal to 1. If # exceeds the total number of occurrences,

all available instances will be replaced. While the find-and-replace operation handles the number of occurrences effectively in the body of the document, it does not work as well with headers and footers because these elements link across entire sections instead of operating on a page-by-page basis. If you need to find and replace text for a certain number of occurrences in the header or footer, it is best to write a macro using Visual Basic for Applications (VBA).

Additionally, note that the reported number of matched occurrences may seem slightly off if you are replacing text in the header or footer, but the occurrences will be consistent with what is reported when you search for text directly in Microsoft Word. For example, text that is used in a repeating header, and thus is found across multiple pages, may be counted only as a single occurrence.

`backward(off)` and `backward(on)` specify the direction of the find-and-replace operation. `backward(off)`, the default, specifies that find and replace be performed from the beginning of the file to the end of the file. `backward(on)` specifies that find and replace be performed from the end of the file, working backward, to the beginning of the file. This option can be combined with `occurrences(#)`, for instance, to find and replace text in the last three instances.

`fontname(`*fontname*`)` sets the font name for the replacement text. *fontname* may be any valid font installed on the user's computer. If *fontname* includes spaces, then it must be enclosed in double quotes. If *fontname* is not available, the fallback font will be used. `fontname(default)` specifies that the replacement text use the same font name as the original text.

`fontcolor(`*color*`)` sets the color for the replacement text. *color* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`. `fontcolor(default)` specifies that the replacement text use the same font color as the original text.

`fontsize(#)` sets the font size. # is a numeric value that represents font size measured in points. The value should be greater than or equal to 1. `fontsize(default)` specifies that the replacement text use the same font size as the original text.

`shading(`*bgcolor* [ , *fgcolor* [ , *fpattern* ] ]`)` sets the background color, foreground color, and fill pattern for the replacement text.

*bgcolor* and *fgcolor* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`. `shading(default)` specifies that the replacement text use the same shading properties as the original text.

Although `shading()` specifies three properties, it is treated as a single option when overriding previous specifications. For example, we might specify a blue background color, green foreground color, and horizontal stripes for the fill pattern by typing `putdocx findreplace replace, shading(blue, green, horzStripe)`. Later, we might change the background color to red by typing `putdocx findreplace replace, shading(red)`; this will also clear out the foreground color and fill pattern we specified before. In other words, results will be the same as if we typed `putdocx findreplace replace, shading(red, default, default)`.

`bold(on)`, `bold(off)`, and `bold(default)` specify whether replacement text should be formatted as bold. `bold(on)` specifies that the replacement text be formatted as bold. `bold(off)` specifies that the replacement text not be formatted as bold. `bold(default)` specifies that the replacement text retain the formatting of the original text.

`italic(on)`, `italic(off)`, and `italic(default)` specify whether the replacement text should be formatted as italic. `italic(on)` specifies that the replacement text be formatted as italic. `italic(off)` specifies that the replacement text not be formatted as italic. `italic(default)` specifies that the replacement text retain the formatting of the original text.

`underline`[ `(`*upattern*`)` ] specifies that the replacement text be underlined and optionally specifies the format of the line. `underline(default)` specifies that the replacement text use the same underline pattern as the original text.

*upattern* may be any of the patterns listed in *Underline patterns* of [RPT] **Appendix for putdocx**. The most common underline patterns are `double`, `dash`, and `none`. `underline` is a synonym for `underline(single)`.

`underlinecolor(`*color*`)` specifies the underline color for the replacement text. *color* may be one of the colors listed in *Colors* of [RPT] **Appendix for putdocx**; a valid RGB value in the form ### ### ###, for example, `171 248 103`; or a valid RRGGBB hex value in the form ######, for example, `ABF867`. `underlinecolor(default)` specifies that the replacement text use the same underline color as the original text.

`strikeout`[ `(`*strikespec*`)` ] specifies whether the replacement text should have a strikeout mark. *strikespec* is one of `single`, `double`, `none`, or `default`. `single` adds a single strikeout mark. `double` adds two strikeout marks. `none` removes the strikeout mark. `default` specifies that the replacement text use the same strikeout mark as the original text. `strikeout` is a synonym for `strikeout(single)`.

## Options for putdocx findreplace reset

`docx` specifies that the source document setting specified with `putdocx findreplace set, docx()` be removed from memory.

`from` specifies that the settings for the text to be replaced, specified with `putdocx findreplace set, from()`, be removed from memory.

`to` specifies that the replacement text settings, specified with `putdocx findreplace set, to()`, be removed from memory.

`scope` resets the scope for find and replace to the default, meaning find and replace will be performed for the entire document.

`matchwholeword` specifies that the finding operation not be restricted to matching only whole words.

`matchcase` specifies that case sensitivity be turned off for the finding operation.

`regex` disables regular expressions.

`occurrences` resets the number of occurrences to all, `occurrences(all)`.

`backward` turns the backward search off; in other words, find and replace will be performed in a forward direction, from the beginning of the document to the end.

`fontname` specifies that the font name settings for replacement text, specified with `putdocx findreplace set, fontname()`, be removed from memory.

`fontcolor` specifies that the font color setting for replacement text, specified with `putdocx findreplace set, fontcolor()`, be removed from memory.

`fontsize` specifies that the font size setting for replacement text, specified with `putdocx findreplace set, fontsize()`, be removed from memory.

shading specifies that the shading properties for replacement text, specified with putdocx findreplace set, shading(), be removed from memory.

bold specifies that the bold formatting setting for replacement text, specified with putdocx findreplace set, bold(), be removed from memory.

italic specifies that the italic formatting setting for replacement text, specified with putdocx findreplace set, italic(), be removed from memory.

underline specifies that the underline setting for replacement text, specified with putdocx findreplace set, underline(), be removed from memory.

underlinecolor specifies that the underline color setting for replacement text, specified with putdocx findreplace set, underlinecolor(), be removed from memory.

strikeout specifies that the strikeout mark setting for replacement text, specified with putdocx findreplace set, strikeout(), be removed from memory.

all specifies that all settings specified with putdocx findreplace set be removed from memory. All formatting settings, such as font name, font size, and case sensitivity, will be reset to their defaults. In other words, the font name and font size for replacement text will be the same as the original text. Additionally, the document settings specified with docx(), from(), and to() will be cleared out. This is the same as Stata's initial state when it first starts.

# Remarks and examples

Remarks are presented under the following headings:

*Introduction*
*Basic find and replace*
*Set the scope for find and replace*
*Replacing text using the saved configurations*
*Reset configurations*

## Introduction

putdocx findreplace is used to find and replace text in .docx files. For a simple, one-time find and replace, you can use putdocx findreplace replace; simply specify the file name you want to work on, the original text, the replacement text, and optionally how to format the replacement text. However, if you have several changes you need to make, or you want to apply changes to multiple files, you can first specify the find-and-replace settings with putdocx findreplace set and then perform find and replace with putdocx findreplace replace. The configurations you specify with putdocx findreplace set will be saved and can be queried with putdocx findreplace query.

After performing find and replace on one file, you may want to reset the configurations back to their defaults before working on your next file. For example, you can customize the font color, font size, bold formatting, and more for the replacement text. With putdocx findreplace reset, you can quickly set all of these configurations back to their defaults; or you can specify which configurations to reset. For example, you might choose to reset the font name to its default but not the font size.

## Basic find and replace

We can perform a simple find and replace with `putdocx findreplace replace`. For example, below we replace all instances of `money` with `cash` in the `header_footer.docx` file and save the changes in the `mydoc.docx` file.

```
. putdocx findreplace replace, docx(header_footer.docx)
> from(‘"money"’) to(‘"cash"’) saving(mydoc.docx) replace
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc.docx
          Original text: money
       Replacement text: cash
                  Scope: {section index: -1, story: all} (default)

Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
successfully replaced file "C:/mypath/mydoc.docx"
(6 matched occurrences)
```

The first section of the output lists the file that we worked on, the original text and the replacement text, and the scope for find and replace. By default, we find and replace text throughout the entire document, but in the next section we demonstrate how to narrow the search.

The second section lists details about how find and replace was performed, including whether we matched only whole words, whether the search was case sensitive, whether regular expressions were allowed, whether we searched backward in the document, and how many occurrences of the original text we wanted to replace. Because we did not specify any of these properties, they are set to their defaults.

The third section lists the formatting for the replacement text, including the font name and font size to be used, whether to bold and italicize text, and whether to underline text. Because we did not specify any of these properties, they are set to their defaults. In other words, the replacement text is formatted the same as the original text.

We can also replace the text with the same text but using a different font format. Below, we find instances of “a” and add bold formatting, a red font, an orange background, and remove the italic formatting. We do so using the regular expression `$0`, which refers to the string we searched for, “a”. By default, regular expressions are not allowed, but we use the `regex` option to enable them.

```
. putdocx findreplace replace, docx(header_footer.docx)
> from(‘"a"’) to(‘"$0"’) regex bold(on) fontcolor(red)
> shading(orange) italic(off) saving(mydoc2.docx) replace
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc2.docx
          Original text: a
       Replacement text: $0
                  Scope: {section index: -1, story: all} (default)

Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: on
        Backward search: off
            Occurrences: -1 (default)

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: red
                Shading: {bg: orange, fg: default, pattern: default}
                   Bold: on
                 Italic: off
         Font underline: default
   Font underline color: default
        Font strikeout: default
successfully created file "C:/mypath/mydoc2.docx"
(48 matched occurrences)
```

In fact, regular expressions could also be used to rephrase text. For example, below we search for instances of “*Name1* took money from *Name2*” and replace that with “*Name2* gave money to *Name1*”, where `[A-z]+` indicates one or more letters and `$1` and `$2` refer to the text matched from the first and second set of parentheses.

```
. putdocx findreplace replace, docx(header_footer.docx)
> from(‘"([A-z]+) took money from ([A-z]+)"’) to(‘"$2 gave money to $1"’)
> regex saving(mydoc3.docx) replace
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc3.docx
          Original text: ([A-z]+) took money from ([A-z]+)
       Replacement text: $2 gave money to $1
                  Scope: {section index: -1, story: all} (default)

Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: on
        Backward search: off
            Occurrences: -1 (default)

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
        Font strikeout: default
successfully created file "C:/mypath/mydoc3.docx"
(6 matched occurrences)
```

In the source file, there were 6 occurrences of "Paul took money from Jason", and in the resulting file, `mydoc3.docx`, you will see that they were replaced with "Jason gave money to Paul".

We have specified different find-and-replace configurations in the examples above using `putdocx findreplace replace`, but these do not get saved. We verify that below:

```
. putdocx findreplace query
Basic settings
            Source file: $ N/A $
          Original text: $ N/A $
       Replacement text: $ N/A $
                  Scope: {section index: -1, story: all} (default)
Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)
Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
```

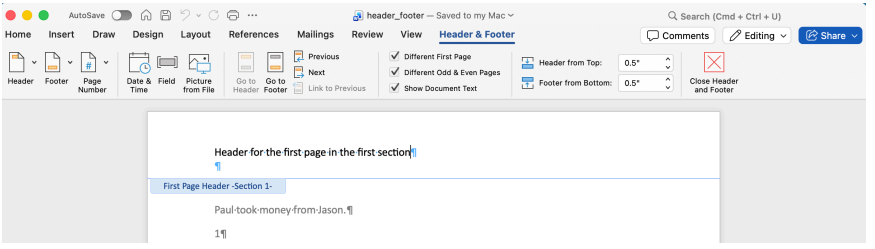
Only the configurations specified with `putdocx findreplace set` get saved and queried.

## Set the scope for find and replace

By default, we find and replace text throughout the entire document. However, for certain changes you may want to narrow the scope of the operation to the body of the document, the header, or the footer. Additionally, you may have section breaks in your document, and each section could have its own header and footer. With the `scope()` option, you can restrict the find-and-replace operation to a specific section or to the headers and footers of a section.

In Microsoft Word, the "Different First Page" setting allows you to have a different header and footer for the first page of a section. This setting is specific to each section and thus can be either disabled or enabled for each section. Additionally, the "Different Odd & Even Pages" setting allows you to have a different header and footer for odd and even pages in your document. This setting is applied to the entire document. Therefore, if you have these settings enabled, you can, for example, narrow the scope to only the headers on the even pages or only the footer on the first page of each section.

For example, in our document we have both settings enabled. You can double-click on the header or footer to see whether you have these settings enabled.




Below, we want to find instances of "page" in the first section of the document and replace them with "pg". We perform this change only for the header on the first page of the section (`header_first`) and the footers on even pages (`footer_even`), except for the first page of the section.

```
. putdocx findreplace replace, docx(header_footer.docx)
> from(`"page"') to(`"pg"') scope(1, header_first footer_even)
> saving(mydoc4.docx) replace
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc4.docx
          Original text: page
       Replacement text: pg
                  Scope: {section index: 1, story: header_first footer_even}

Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
successfully created file "C:/mypath/mydoc4.docx"
(2 matched occurrences)
```

If we instead wanted to perform this change for all sections, meaning for the header on the first page of each section and the footers on even pages, except for the first page of each section, we would simply omit the section number as in `scope(, header_first footer_even)`.

## Replacing text using the saved configurations

If there are configurations you want to use for multiple find-and-replace operations, you can save them once and then apply them in the find-and-replace operations.

For example, below we set the document to work on, scope, and matching and replacement text. We specify that we want the finding operation to be case sensitive and that the replacement text should have two strikeout marks. Additionally, we specify that the replacement text should be bold with a wavy underline and that we want to search backward, beginning from the end of the document and replace only the last two occurrences (the first two in our backward search). Then we query the settings and confirm that they were saved.

```
. putdocx findreplace set, docx(header_footer.docx) scope(, body)
> from(‘"a"’) to(‘"b"’) matchcase strikeout(double)
. putdocx findreplace set, underline(wave) bold(on) backward(on) occurrences(2)
. putdocx findreplace query
Basic settings
            Source file: C:/mypath/header_footer.docx
          Original text: a
       Replacement text: b
                  Scope: {section index: -1, story: body}
Find-and-replace properties
 Match whole words only: off
             Match case: on
    Regular expressions: off
        Backward search: on
            Occurrences: 2
Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: on
                 Italic: default
         Font underline: wave
   Font underline color: default
        Font strikeout: double
```

Below, we conduct the find-and-replace operation, and we use `strikeout(none)` to temporarily remove any strikeout marks. We use the `inherit` option to inherit the configurations specified with `putdocx findreplace set`.

```
. putdocx findreplace replace, inherit strikeout(none)
> saving("mydoc5.docx") replace
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc5.docx
          Original text: a
       Replacement text: b
                  Scope: {section index: -1, story: body}
Find-and-replace properties
 Match whole words only: off
             Match case: on
    Regular expressions: off
        Backward search: on
            Occurrences: 2
Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: on
                 Italic: default
         Font underline: wave
   Font underline color: default
        Font strikeout: none
successfully created file "C:/mypath/mydoc5.docx"
(2 matched occurrences)
```

As a result, all configurations we saved, which can be queried by `putdocx findreplace query`, are adopted in the replacing operation, except for the strikeout mark, which was overridden temporarily by `putdocx findreplace replace`.

We can also replace different portions of text in a file using a loop. For example, below we first reset all find-and-replace configurations to their defaults, because we do not want the bold formatting, backward search, and other settings we specified previously. Then we set the document to work on and the background color. Finally, we loop over values 1, 2, and 3, adding the "Stata" prefix to these values in the document. In the output that follows, you can confirm that "1" gets replaced with "Stata 1", "2" gets replaced with "Stata 2", and so on.

```
. putdocx findreplace reset, all
. putdocx findreplace set, docx(header_footer.docx) shading(red)
. forvalues j = 1(1)3 {
  2.     putdocx findreplace replace, from(`""`j'""') to(`""Stata `j'""')
>             saving(mydoc6.docx) replace inherit
  3. }
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc6.docx
          Original text: 1
       Replacement text: Stata 1
                  Scope: {section index: -1, story: all} (default)
Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)
Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: {bg: red, fg: default, pattern: default}
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
successfully created file "C:/mypath/mydoc6.docx"
(6 matched occurrences)
```

```
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc6.docx
          Original text: 2
       Replacement text: Stata 2
                  Scope: {section index: -1, story: all} (default)

Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: {bg: red, fg: default, pattern: default}
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
successfully replaced file "C:/mypath/mydoc6.docx"
(6 matched occurrences)
Basic settings
            Source file: C:/mypath/header_footer.docx
       Destination file: C:/mypath/mydoc6.docx
          Original text: 3
       Replacement text: Stata 3
                  Scope: {section index: -1, story: all} (default)

Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: {bg: red, fg: default, pattern: default}
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
successfully replaced file "C:/mypath/mydoc6.docx"
(6 matched occurrences)
```

Similarly, you could use a nested loop to replace different portions of text for multiple files, if the filenames share the same pattern. For example, if we wanted to make the same change as above for files `myreport.docx`, `myreport2.docx`, and `myreport3.docx`, we could use this loop:

```
putdocx findreplace reset, all
putdocx findreplace set, shading(red)
foreach x in myreport1.docx myreport2.docx myreport3.docx {
  forvalues j = 1(1)3 {
    putdocx findreplace replace, docx(‘x’) from(‘"‘j’"’) to(‘"Stata ‘j’"’) ///
    replace inherit
  }
}
```

## Reset configurations

You can reset one or more configurations. For example, below we add two strikeout marks, a wavy underline, and bold formatting to the replacement text. We also specify that find and replace be performed backward and that only two occurrences be replaced.

```
. putdocx findreplace set, docx(header_footer.docx) scope(, body)
> from(‘"a"’) to(‘"b"’) matchcase strikeout(double)

. putdocx findreplace set, underline(wave) bold(on) backward(on) occurrences(2)

. putdocx findreplace query

Basic settings
            Source file: C:/mypath/header_footer.docx
          Original text: a
       Replacement text: b
                  Scope: {section index: -1, story: body}

Find-and-replace properties
 Match whole words only: off
             Match case: on
    Regular expressions: off
        Backward search: on
            Occurrences: 2

Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: on
                 Italic: default
         Font underline: wave
   Font underline color: default
         Font strikeout: double
```

And now we reset the direction of the search and number of occurrences back to their defaults. The wavy underline, bold formatting, and strikeout marks will still be applied when we replace text.

```
. putdocx findreplace reset, backward occurrences
. putdocx findreplace query
Basic settings
            Source file: C:/mypath/header_footer.docx
          Original text: a
       Replacement text: b
                  Scope: {section index: -1, story: body}
Find-and-replace properties
 Match whole words only: off
             Match case: on
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)
Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: on
                 Italic: default
         Font underline: wave
   Font underline color: default
         Font strikeout: double
```

We can also reset all configurations. In our last query, we had bold formatting, a wavy underline, and two strikeout marks. Below, we reset all configurations to their default:

```
. putdocx findreplace reset, all
. putdocx findreplace query
Basic settings
            Source file: $ N/A $
          Original text: $ N/A $
       Replacement text: $ N/A $
                  Scope: {section index: -1, story: all} (default)
Find-and-replace properties
 Match whole words only: off
             Match case: off
    Regular expressions: off
        Backward search: off
            Occurrences: -1 (default)
Formatting
              Font name: default
              Font size: -1 (default)
             Font color: default
                Shading: default
                   Bold: default
                 Italic: default
         Font underline: default
   Font underline color: default
         Font strikeout: default
```

## Also see

[RPT] **putdocx intro** — Introduction to generating Office Open XML (.docx) files+

[RPT] **Appendix for putdocx** — Appendix for putdocx entries

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.



For suggested citations, see the FAQ on citing Stata documentation.