

Syntax Highlighting

Article Number: 1538 | Last Updated: Fri, Dec 7, 2012 4:34 PM

The Syntax Highlighting branch under Configuration - Editor Display allows the user to configure syntax highlighting options:

Syntax highlighting is the ability to recognize predefined words and display them in different colors. This is particularly useful for programmers and can also be useful to other users who may want certain words in a document to show up in a different color.

Support for many languages or word sets that may be recognized is provided. Each word set may be configured to have a different color for the following groups of words:

Normal Text (Words that are not recognized)

Words that appear in a comment line, or block comments

Alternate set of block comments

Strings

Number values (must start with a digit (0-9))

Up to eight groups of different recognized words

The Configuration Dialog allows the colors to be selected for the different language and color groups. This also allows the user to specify the location of the "wordfile" used for highlighting.

For any of the color groups the foreground/text color and the background color may be set. By default the background will revert to the background specified for normal text (Background color automatic will be checked). This must be unchecked to activate the background color.

Each color may be changed by clicking on the colored box beside the text description. When clicked, a dialog box will appear that allows the new color to be selected.

Additionally for each color group (except Normal Text) font styles of Bold, Italic and Underline are available. These may be selected individually for each color group. With some fonts the underline may not always show correctly, and with bold, the spacing may not be correct for non-fixed pitch fonts.

Note - The highlighting is determined by the file extension (details below). The file extension of the file being modified must be defined in the "wordfile" for one of the languages.

The predefined words may be configured by the user as follows:

UltraEdit reads configuration files (the default file extension is "*.UEW") from the directory specified under Advanced -> Configuration -> Editor Display -> Syntax Highlighting, to configure the syntax highlighting. These files are read each time the editor starts up. Each file may be up to 372KB in size. The syntax for these files is as follows:

Language Definition

The word set, or language is specified by "/Ln". This must be at the beginning of the line. A description or name of the language may be specified immediately following the / Ln in quotes. This description if present is displayed when setting up the colors for the language. The description may be up to 24 characters.

Line Comments

The comment characters used for line comments are specified by the string "Line Comment = " followed by the comment characters. Five characters are allowed; if there are less than five then the last character must be followed by a space.

A second set of line comments may be specified by the string "Line Comment Alt = " followed by the comment characters. Five characters are allowed; if there are less than five then the last character must be followed by a space.

As some instances may require that a space is a part of the line comment an alternative method to describe the line comment is available. Using the alternative method, the number of characters used is specified by the user with the following syntax:

"Line Comment Num = xCC "

where x specifies the number of characters (1 to 5) and IMMEDIATELY following are the characters to be used as line comments. In the example above, x would be 3, and the line comment would be "CC " (note the space after "CC").

Additionally, there are times when qualifiers are required for line comments such that the comments are only valid if they occur at certain columns, or after certain characters (well, more often they are not valid if they follow certain characters). To cater for this, two additional line comment commands are provided:

"Line Comment Preceding Chars = [...]"

and

"Line Comment Valid Columns = [1-7,10]"

The default for "Line Comment Preceding Chars" is that all characters are valid. Therefore this entry would include the characters that are not valid following a tilde character as in ...[~a-z]. This would say that the comment is not valid if it immediately follows a character in the range a-z. Any characters are valid between the brackets.

The default for "Line Comment Valid Columns" is that all columns are valid if this is not defined. If this is defined, then only the columns specified are valid. There can be up to 10 column ranges, or columns specified separated by a comma as in ...[1-7, 10]. This indicates that the comment characters are valid if they occur at columns 1 through 7, or at column 10.

Block Comments

The characters used for block comments may also be configured (i.e. /* ... */ for 'C'). These are in the form "Block Comment On = " and "Block Comment Off = " followed by up to nineteen characters each that define the comment designators. For compatibility with previous releases, the '/' and '*' are used for file types that have '.C' as one of the extensions. This may be overridden. The first character of the block comments may be a space.

If a "Block Comment On" is defined but the "Block Comment Off" is not defined the commenting will stop at the end of the line. This effectively allows the block comments to be used as line comments also.

Additionally, a second set of block comments may be defined for languages that require it. This is particularly useful for ASP allowing HTML comments to be maintained with the addition of adding highlighting for ASP blocks.

The second set of block comments are in the form "Block Comment On Alt = " and "Block Comment Off Alt = " followed by up to nineteen characters each that define the comment designators. The first character of the block comments may be a space.

If a "Block Comment On Alt" is defined but the "Block Comment Off Alt" is not defined the commenting will stop at the end of the line. This effectively allows the block comments to be used as line comments also.

UltraEdit/UEStudio by default will end any block comment whenever a single block comment end string is encountered. To instruct UltraEdit/UEStudio to explicitly match all block comment start strings with all block comment end strings, the nest block comment command may be specified. This is accomplished by adding the **NestBlockComments** command to the language definition line (i.e., /L3"HTML"
NestBlockComments ...).

File Extensions/Types

Syntax Highlighting is determined either by the name of the file or its extension. More commonly the extension is used and to specify the extensions for which this language is applicable the following string

should be used: "File Extensions = " and each extension is separated by a space. File extensions specified here are NOT case sensitive.

The file extension is everything AFTER last dot in the name of a file. Up to 97 bytes may follow the "File Extensions = " declaration. If a list of file extensions exceeding 97 bytes is specified here, they will not all be parsed.

Users may specify a language as the default for syntax highlighting by placing an asterisk "*" as the last file extension specified. This would also be used as the highlighting file type for unsaved files or files saved without an extension.

To specify that a filename rather than file extension is to be used to determine the language the following string should be used: "File Names = " and each name is separated by a space. This must include the extension, i.e.: "File Names = ThisFileName.xml". Up to 125 bytes may follow the "File Names = " declaration.

File Names/Extensions Considerations

The following items must be considered if using both File Names and File Extensions options in the same wordfile:

1. Names including spaces may not be specified with the **File Names** option.
2. The **File Names** option may not be used in conjunction with the **File Extensions** option.
3. If a language is defined using the **File Names** option, and a different language within the same wordfile is specified using the **File Extensions** option and the definitions overlap, the language defined with the **File Names** option must have a lower language number, i.e.:

Example 1 - won't work:

```
/L6"XML General" XML_LANG Noquote ... File Extensions = XML XUL XSD XSL XSLT
```

```
/L12"XML Special" XML_LANG Noquote ... File Names = ThisIsJustAFile.xml
```

Example 2 - will work:

```
/L6"XML Special" XML_LANG Noquote ... File Names = ThisIsJustAFile.xml
```

```
/L12"XML General" XML_LANG Noquote ... File Extensions = XML XUL XSD XSL XSLT
```

Color Selection

Color codes may be specified by adding a line with /Cn at the beginning of the line, where n is the color index of 1 to 20. A description or name of the section may be specified immediately following the /Cn in quotes. This description if present is displayed when setting up the colors for the language. The description may be up to 24 characters.

All information specified remains in effect until overridden with new command information.

The following example specifies the first language to be used with files with the extensions 'C', 'CPP', 'H' or 'HPP'. The color used for the words is the first selectable color and the comment characters are //.

```
/L1"C/C++" Line Comment = // Block Comment On = /* Block Comment Off = */ File Extensions = CPPC H HPP
```

```
/C1
```

```
auto
```

```
break
```

```
case char const continue chr$
```

```
default do double
```

Case Sensitivity

If the language is not case sensitive, the keyword "Nocase" may be added to the command line i.e.:

```
/L1 Line Comment = // Nocase File Extensions = CPPC H HPP
```

Strings

When using many programming languages, characters in single quotes and double quotes are treated as literal strings, and word and comment recognition should be ignored. This is the default behavior for UltraEdit. There are some languages, (i.e. HTML and others) where this behavior is undesirable. To facilitate such languages the keyword "Noquote" may be added to the command line to override the default behavior i.e.:

```
/L1 Noquote File Extensions = HTM
```

UltraEdit now supports configurable characters for quotes strings. The default characters for strings are single and double quotes (') and ("). These may be overridden with the keywords "String Chars = " followed by up to two characters. This is only required if you wish to use different characters from the default. If for example you wish to use the double quote only for strings you would have a line similar to the following:

```
/L1"C/C++" Line Comment = // Block Comment On = /* Block Comment Off = */ String Chars = " File Extensions = CPPC H HPP
```

Note the "String Chars" portion.

Additionally, if you have two characters for the strings defined (or using the defaults) and you wish to have a different color for each type of string, you may now include the character (i.e. double quote) in one of the color groups in a line by itself. This will override the configurable color for the strings that are encapsulated by this character.

```
/L1"C/C++" Line Comment = // Block Comment On = /* Block Comment Off = */ String Chars = "" File  
Extensions = CPPC H HPP
```

```
/C1
```

```
auto
```

```
break
```

```
"
```

```
...
```

In the above example, all strings that have double quotes (") around them will be colored with the color group C1 while strings with a (') around them will use the default color for strings.

Escape Character

In many languages there is a special designated character that is referred to as an Escape Character. This is used to override the normal string characters (and other characters). If a string is defined that itself contains a quote character normally used to define the string, the syntax highlighting would interpret this quote as the end of the string. If however this quote character is preceded with an Escape Character the quote character would be appropriately treated as part of the string and not the end of it.

The Escape Character is defined as follows:

```
/L1"C/C++" Escape Char = \ ...
```

In this case the Escape character is defined as a backslash.

Keywords

Note that ALL words starting with the same character may be on the same line or spread across multiple lines, however if they are spread across multiple lines the lines must be one after the other with no empty lines or other lines between them.

If the language is case sensitive, the letter 'A' is different from 'a' and so words starting with 'A' MUST be on

a different line from words starting with 'a'. If the language is case insensitive words starting with the letter 'A' must be on the same line as words starting with the letter 'a'.

Keywords beginning with a Sub String

There are instances in some languages where it is desirable to highlight keywords that begin with a particular sub-string, however the complete word is not known. UltraEdit provides the ability to define sub-strings that are used to determine if a word should be highlighted. If such sub-strings are defined for a particular language under a color group UltraEdit will determine if a word begins with one of the sub-strings. If it does, it will be highlighted accordingly.

The sub-strings should be defined as with any other set of keywords however the line containing the sub-strings should start with "** " and all sub-strings should be on the same line as in:

```
** aaa bbb
```

The example above would highlight any words beginning with "aaa" or "bbb".

Words Starting with '/'

As UltraEdit uses '/' as a command character words to be highlighted require special handling. To highlight words beginning with a '/' the line should begin with '// ' followed by the keywords as in:

```
// / mykeyword / anotherkeyword
```

HTML Specific

HTML is considerably different from other languages, and to better facilitate the use of UltraEdit for editing HTML files the "HTML_LANG" keyword was added. When this keyword is present, the special characters '<' and '/' may be placed in front of any keyword as desired without all keywords with the special ('<' and '/') characters all having to be on the same line. In this case, words starting with the same letter (a-z etc) must be on the same or contiguous lines as is normally required.

Below is an example portion of a word file for HTML:

```
/L3"HTML" Line Comment = // HTML_LANG Block Comment On = <!-- Block Comment Off = --> File  
Extensions = HTM HTML
```

```
/C1
```

```
<A </A> <ADDRESS> </ADDRESS> <APPLET </APPLET>
```

```
<B> </B> <BASE <BASEFONT <BGSOUND <BIG> </BIG> <BLINK> </BLINK> <BLOCKQUOTE>  
</BLOCKQUOTE>
```

<CAPTION> </CAPTION> <CENTER> </CENTER> <CITE> </CITE> <CODE> </CODE>

Additional enhancements for HTML will be added in the future.

FORTRAN Specific

FORTRAN is quite different from other languages regarding comments, and to better facilitate the use of UltraEdit for editing FORTRAN files the "FORTRAN_LANG" keyword was added. When this keyword is present, UltraEdit treats a 'C', 'c' or '*' in the first column as a line comment indicator and the rest of the line is highlighted as if it were commented out.

Below is an example portion of a word file for FORTRAN:

```
/L4"Fortran" FORTRAN_LANG File Extensions = FOR FTN
```

```
/C1
```

```
...
```

Note: Any of the normal comment indications may also be used (line comments, block comments).

LaTeX/Tex Specific

TeX/LaTeX is quite different from other languages and each command starts with a '\'. To better facilitate the use of UltraEdit for editing TeX/LaTeX files the "LATEX_LANG" keyword was added. When this keyword is present, UltraEdit has special handling for syntax highlighting to allow words to be appropriately handled and highlighted with the '\', and with consecutive words.

This also allows the recognized words to be placed in the wordfile without all of them being on the same line. If the word begins with '\' then the second character is used to determine which line the word should be on. All words beginning with '\a' should be on the same line as other words beginning with '\a' or 'a'. In the same way, all words beginning with '\b' should be on the same line as other words beginning with '\b' or 'b' but on a different line from those starting with '\a' etc.

Below is an example portion of a word file for TeX/LaTeX:

```
/L4"TeX/LaTeX" LATEX_LANG File Extensions = TEX LATEX
```

```
/C1
```

```
...
```

Note: Any of the normal comment indications may also be used (line comments, block comments).

Delimiters

UltraEdit has built in delimiters that are used to determine when a new word starts and when a word finishes so that it may be matched against the set of words for a given language. With the exception of the '<' and '>' characters in HTML a character that is a delimiter may not also be part of a word; that is, you can not say the @ symbol is to be considered part of a word, and it is a delimiter between words. (Future enhancements may allow this).

With release 4.1 and later, UltraEdit allows the delimiter characters to be configurable by the user. For compatibility, the existing delimiters are retained if the user does not specify the delimiters for a given language. Each language may have its own set of delimiters. It is not necessary to configure the delimiters for languages if you are using the defaults.

To specify the delimiters, add a NEW line similar to the following to your wordfile:

```
/Delimiters = ~!@$%^&*()_-=|V{ }[];'"<> ,?/
```

Note that the Delimiters should include a space and a TAB character if you want them to be considered delimiters. The line must begin with "/Delimiters =".

It is possible to assign the delimiter characters to the color sections. If you have a character that is a delimiter, such as a '+', and you wish this to be colored with one of the group colors you may add this character to a line of its own under the color section, and this will retain its recognition as a delimiter and be highlighted with the appropriate color. A delimiter may be included at the beginning of a keyword and be highlighted accordingly but may not be included in the middle of a keyword. If a "compound" keyword, or a keyword that includes a delimiter character between two sections is desired, the delimiter character would need to be removed from the Delimiters list, or the two portions of the keyword would need to be defined separately to highlight correctly.

Function Definition Strings

UltraEdit allows the user to show a list of functions in the active document, or all project documents. As the definition of a function may be different for different languages it is necessary to allow this to be configured based on the language.

This is accomplished by modifying the "wordfile" and defining the string UltraEdit uses for each language. The string will be used by UltraEdit as a Regular Expressions search string to find the functions.

By default the regular expression string uses UltraEdit style regular expressions. If desired, Perl compatible Regular Expressions may be used for the function strings by adding the following:

```
/Regexp Type = Perl
```

Please note: Perl regular expressions may only be used for the flat function list (as opposed to the hierarchical function list) and these may only be updated by manually modifying the wordfile. They cannot currently be changed through the Modify Groups dialog.

Up to six functions strings per language may now be defined allowing more function/procedure formats to be specified.

To specify the function string add a line similar to the following for the specific language:

```
/Function String = "%[ a-zA-Z]*")"
```

or

```
/Function String 1 = " ..."
```

...

```
/Function String 5 = " ..."
```

This string to be searched for MUST be in quotes. If required, you can specify that only part of the resultant string is displayed in the function list. To do this, use the tagged expressions as defined under the Regular Expressions and enclose the portion of the expression that is to be displayed between "^(" and "^)" as in:

```
/Function String = "%[ a-zA-Z]+^(*^))"
```

This example would ignore the first word and display the rest of the line.

Indentation

UltraEdit provides for automatic indentation based on the specific language to indent a block of code or to unindent a block of code.

For indentation for a specific language add a line similar to the following line under the particular language section:

```
/Indent Strings = "{ "
```

Any number of words may be specified in quotes (each word/string must be in a separate set of quotes ""). If an indent string occurs anywhere on a line it will be used for indenting (except when it occurs in quoted/commented text). The indentation is the next TAB stop from the indentation of the preceding line (same as if a TAB key was pressed).

For out-denting for a specific language add a line similar to the following line under the particular language section:

```
/Unindent Strings = "}"
```

Any number of words may be specified in quotes (each word/string must be in a separate set of quotes ""). If an indent string occurs anywhere on a line it will be used for indenting (except when it occurs in quoted/commented text). If the line that contains the string is indented LESS than the preceding line an indentation does not occur, otherwise the matching character/word is out-dented to the preceding TAB stop and the new line is indented to the same point.

For reindenting of files it may be undesirable to indent certain lines that are commented out, or compiler directives. This can be avoided by specifying lines that should not be indented. To do this add a line similar to the following to the appropriate language section in the wordfile:

```
/Ignore Strings SOL = "#" "/"
```

Any number of words may be specified in quotes (each word/string must be in a separate set of quotes ""). For the word to match it must be the FIRST character(s) of the line. If a line matches this, it would not be indented, however the indenting of the next line would be performed as if this line was not present.

Please note: If no indent/unindent strings are specified, UltraEdit will use "{" as a default indent string and "}" as a default unindent string.

Brace Matching

Open/Close brace strings may be for each language defined in the wordfile. This extends and enhances the functionality of the Match Brace command in the Search menu and auto-brace matching which is enabled in the Syntax Highlighting dialog.

To define an open brace add a line similar to the following under the particular language section:

```
/Open Brace Strings = "{ " "(" "["
```

or

```
/Open Brace Strings = "If" "For" "Select Case" "Else" " ElseIf"
```

To define a close brace add a line similar to the following under the particular language section:

```
/Close Brace Strings = "}" " )" "]"
```

or

```
/Close Brace Strings = "End If" "Next" "End Select" "End If" " ElseIf"
```

Please note: Open and close brace strings must be positionally matched in their lists for this to work as desired. If a "{" is defined as the first Open Brace String then "}" should be defined as the first Close Brace String. They must also be unique. For example, "endcase" cannot be used as the close brace for "case" and "casez".

If open/close brace strings are not defined UltraEdit uses a standard group of characters when performing brace matching functions as in previous versions.

Marker Characters

There are times when it is desirable to highlight all characters between two characters. UltraEdit provides for "marker characters" that mark the first and last part of a string that UltraEdit highlights between. All characters between the two characters are highlighted.

To define marker characters for a specific language add a line similar to the following line under the particular language section:

```
/Marker Characters = "ab"
```

where 'a' is the first character of the string to be highlighted and 'b' is the last character. Note that all characters on a line will be highlighted including spaces. If the line is a comment or string this is ignored. Alphanumeric characters may be used, but whitespace characters (space/tab) may not be used for marker characters.

Additionally, you may define up to 4 pairs of characters to highlight between as in:

```
/Marker Characters = " abcdefgh"
```

where strings starting with 'a' and ending with 'b' are highlighted as are strings starting with 'c' and ending with 'd' etc.

The color of the highlighted string must be configured. To specify the color, add the two characters under the appropriate color section as if they were a word such as "ab", "cd", etc.

Open/Close Fold Strings

UltraEdit has defaults for Open and Close folding strings for many languages. If no strings are specified in the Wordfile, the defaults are used. Otherwise, the specified strings are used for folding.

/Open Fold Strings = "{ "

/Close Fold Strings = "}"

Ignore Fold Strings

In specific cases it is necessary for the Fold logic to ignore certain lines which contain a string. If an Ignore Fold String is found on a line of source

code (outside a line comment or block comment) the fold logic will ignore any other open or close fold strings on that line. No defaults are provided

for Ignore Fold Strings.

/Ignore Fold Strings = "Exit Function"

Open/Close Comment Fold Strings

UltraEdit provides a means to specify unique Open and Close Fold Strings which are recognized in line or block comments only. No defaults are provided for Open and Close Comment Fold Strings.

/Open Comment Fold Strings = "#Region"

/Close Comment Fold Strings = "#End Region"

Multi-line Strings

UltraEdit has defaults for multi-line string highlighting capability for many languages. If no strings are specified in the wordfile, the defaults are used. Otherwise, one of the following strings may be added to the language definition line (i.e., /L8 PHP ...)

EnableMLS

DisableMLS

String Literal Prefix

UltraEdit supports specification of the string literal format by specifying a string prefix character in the language definition line. For example, in C# .NET 2.0, the string literal prefix would be defined as follows:

String Literal Prefix = @

In this case, the @ before a string literal indicates that a backslash is not an escape character which is really useful for entering path literals. To illustrate, the following two statements are equivalent:

```
"c:\\data\\"
```

```
@"c:\data\"
```

The only special character in a @"..." literal is the ", which is simple doubled if you want to embed one.

Language Markers

UltraEdit has improved syntax highlighting and it now supports multiple languages within a single file. This is specifically for HTML type files. To help facilitate this, we have added additional language indicators that should be added to the wordfiles (*.uew) to indicate the type of language for any languages that may be included within another. Our default wordfiles have these modifications.

Example:

If an HTML file includes PHP then the syntax highlighting section must exist in the main wordfile and the PHP section should include in the definition line: PHP_LANG

Unambiguous language markers have been added to the default wordfile for the following languages:

C_LANG	// C/C++
COBOL_LANG	// Cobol
FORTTRAN_LANG	// Fortran
PASCAL_LANG	// Pascal
PERL_LANG	// Perl
PLB_LANG	// Plb
VB_LANG	// Visual Basic
VBSCRIPT_LANG	// Vb Script
ASP_LANG	// ASP

–

CSHARP_LANG	// C Sharp
CSS_LANG	// CSS
LATEX_LANG	// TeX / LaTeX
HTML_LANG	// HTML
JAVA_LANG	// Java
JSCRIPT_LANG	// Javascript
ECMA_LANG	// Ecma / EcmaScript
PHP_LANG	// PHP
MATLAB_LANG	// MATLAB
PUREBASIC_LANG	// PureBasic
PYTHON_LANG	// Python
SQL_LANG	// SQL
XML_LANG	// XML
XSL_LANG	// XSL
MASM_LANG	// Microsoft Assembler
AASM_LANG	// AT&T Assembler
NASM_LANG	// Netwide Assembler

Currently, UltraEdit uses the above language markers to correctly syntax highlight multiple languages within a file. In the future UltraEdit may make further use of these languages markers in the Wordfile.

NOTE: If your wordfile has multiple occurrences of the HTML_LANG language marker, this must be corrected or some languages may not highlight correctly. UltraEdit allows only one instance of a language marker (i.e., C_LANG, COBOL_LANG, HTML_LANG) in a wordfile.

The following file extensions were moved to the HTML section of the default wordfile: php, asp, and jsp. This was done to facilitate correct HTML highlighting at the outer language level of the file even when no <HTML> language tag is specified within the file. All php, asp, and jsp sections of the files are highlighted correctly based on language start and end tags (<? and ?> for example), however a "View As" will show the file as HTML type.

IntelliTip support

Style keywords have been added to the wordfile to identify color groups for IntelliTip support. Currently the use of

the style elements is limited, but will find expanded use in future enhancements. A complete list of style elements follows:

STYLE_KEYWORD	// Keyword
STYLE_FUNCTION	// Function
STYLE_EXTENSION	// Extensions
STYLE_IDENTIFIER	// Identifier
STYLE_OPERATOR	// Operator
STYLE_METHOD	// Methods
STYLE_EVENT	// Events
STYLE_STATEMENT	// Statements
STYLE_TAG	// Tag
STYLE_VARIABLE	// Variable
STYLE_ATTRIBUTE	// Attributes
STYLE_ELEMENT	// Element
STYLE_COMMAND	// Command

Highlight new file as

Using the dropdown for this feature the user may set a default language for syntax highlighting from the list of languages defined in the active wordfile. This language will be used as the default for unsaved files.

Spell checking

By default UltraEdit and UESstudio will not perform "spell check as you type" on any syntax-highlighted files. To change this setting and allow "spell check as you type" for a syntax-highlighted file, add the following to the language definition line (e.g., /L3"HTML" ...) of the wordfile:

EnableSpellasYouType

The appropriate spell check option must also be selected in Configuration to support this.

The wordfile(s) may be modified by the user.

Posted - Thu, Oct 13, 2011 7:45 PM. This article has been viewed 2691 times.

Online URL: <http://www.ultraedit.com/help/article/syntax-highlighting-1538.html>
