# Quick Start

To {quickly Text=”Quick Start”} test the program works, do the following steps:

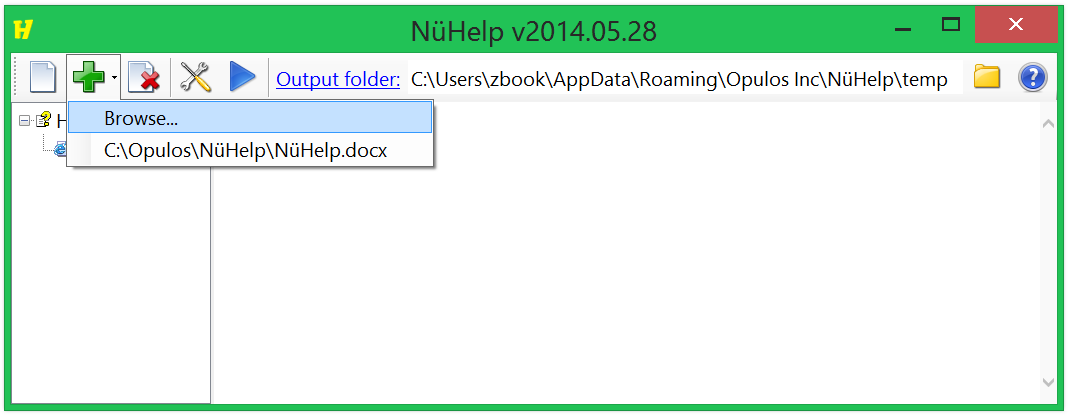
1. Download and install Microsoft’s HTML Help Workshop
   1. URL: <http://www.microsoft.com/en-ca/download/details.aspx?id=21138>
   2. It is also included in the zip file you downloaded (htmlhelp.exe)
2. Run NuHelp.exe.
3. Click the Compile button (blue triangle)
   1. This will compile the default ‘Welcome’ page
4. Click the Output folder: link, this will open the folder.
5. In that folder, there will be a help file called ‘Help Welcome.chm’
6. Done.

# About [CHM InvariantName=”About”]

**{NüHelp}** is a program developed and supported by {**Opulos Inc.**} Our website is [www.opulos.com](http://www.opulos.com). The initial beginning of this program was based on the codeproject Word2CHM. A large number of improvements to the code and features were added in order to create a richer help file experience for the end user. Also, for programmers, an easy to use framework is provided that links an application’s screens to specific sections in a help file, see [Help File Integration](#_Help_File_Integration).

To read about all the {features} added, see the [NüHelp.chm](file://NüHelp.chm) included in the release.

# Main Screen [CHM InvariantName=”MainScreen”]



**New Project.** Removes existing nodes from the project tree and deletes [temporary files](#_Temporary_Files) (if that option is checked).

 **Add.** Browse for an existing html or doc/docx file to add into the current project. Both Word 97-2003 doc and Word 2007 to 2013 docx formats are supported. The drop down menu remembers 10 of the most recent files added. You can right click on the files to have the option to remove them from the list, open the containing folder, or open the file for editing.

**Remove**. Removes the selected nodes from the project tree. Any [temporary files](#_Temporary_Files) for the removed nodes will be deleted (if that option is checked).

**Options**. Additional settings for the project and to pass to the compiler. The drop down menu remembers 10 different compile option configurations.

**Compile.** Compiles all HTML nodes in the project tree into a single CHM file. The output of the CHM file will be in the Output Folder (unless an explicit filename is used).

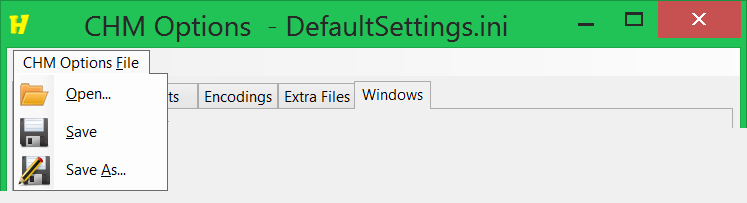
**Browse.** Opens a folder explorer window to pick a new output folder. You can also type or paste the output folder directly in the text box.

**Help.** Opens the help file for this application.

# CHM Options [CHM InvariantName=”CHMOptions”]

**The CHM Options are for advanced users** who want to customize the look of the help file. Examples might be changing the {font} used in the TOC or Index, making the help file window {Topmost} or customizing which {buttons} are available. All values are explained in this section. Beginner users do not have to worry about the {CHM Options} at all because the default values produce an all-around good purpose help file.

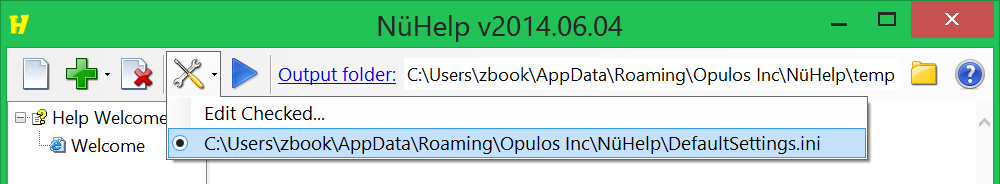
At the top of the CHM Options dialog, there is a “CHM Options File” menu, with the options to Open, Save & Save As. This allows users who need different compile options for different documents to be able to save the compile options to a configuration ini file.



**DefaultSettings.ini** is the default XML text file that saves the options. The file is created in your roaming directory, for example on Windows 8:

C:\Users\**<Username>**\AppData\Roaming\Opulos Inc\NüHelp\**DefaultSettings.ini**

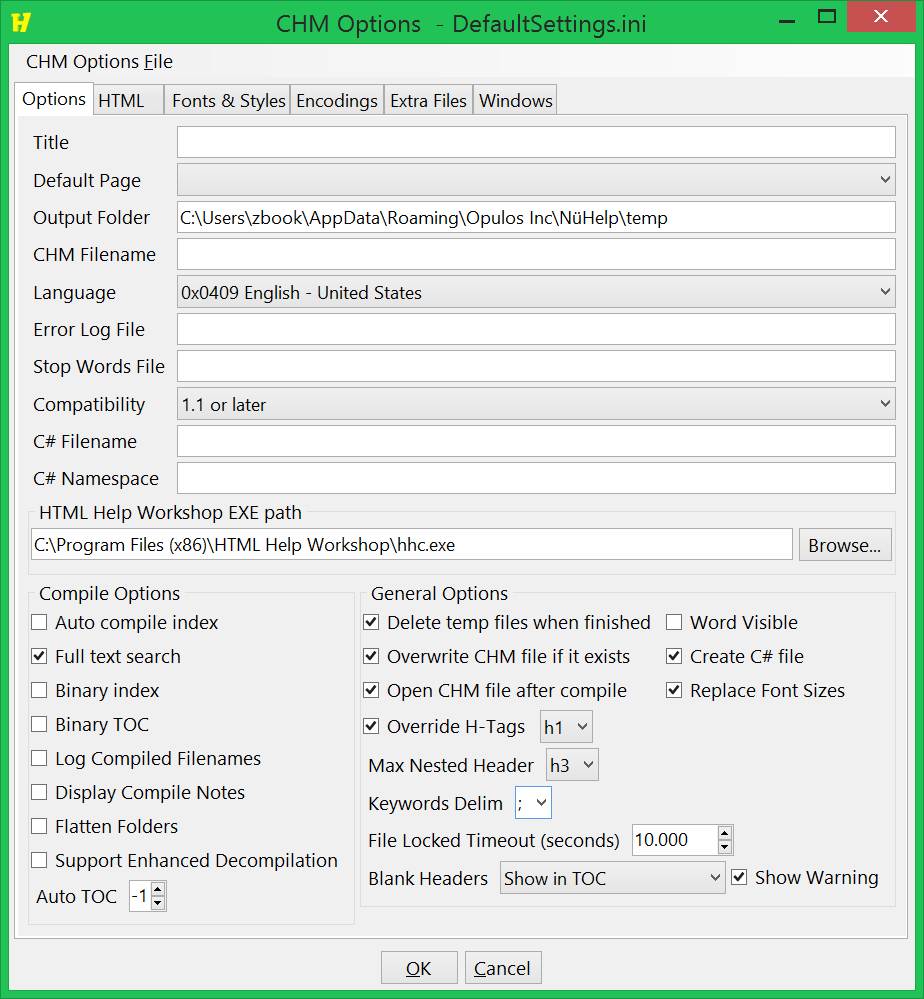
When a CHM Options file is opened, it is added to the drop down menu on the CHM Options button:



As you can see, there is a radio button, which indicates the current settings used when compiling. As CHM Options files are saved, more items are added to the drop down list. This makes it very easy to quickly switch between different compile options. You can also edit the DefaultSettings.ini and save it.

Users working with a large number of documents and different compile settings may wish to use the [command line](#_Command_Line_[CHM) {Text=”Command Line”} instead.

## Options [CHM InvariantName=”Options”]



**Title:** Optional. The title that appears at the top of the Help window. If left blank, then the title is automatically replaced with the document’s filename. Under the Windows tab, if the primary window has a title then that title takes precedence.

**Default Page:** Optional. The initial page displayed when the Help window is opened. If not specified then the first page is used as the default page. It must be a local page and cannot be a link to a website.

**Output Folder:** Required. The location where the temporary html files and CHM file are saved to.

**CHM Filename:** Optional. Can be just a filename, relative path or absolute path. If it’s an absolute path then the CHM file is saved to that file. If it’s a relative path then it is saved relative to the Output Folder.

**Language:** Required. Important to set this correctly since HH 1.x is an ANSI application and must know the correct language, Font, Charset etc. to use when sorting index entries and displaying text.

**Error Log File:** Optional. Can be an absolute or relative path. If specified, the compiler output is appended to the log file.

**Stop Words File:** Optional. A file containing a list of Stop words. Possibly not well implemented by Microsoft. Create a txt file containing a word on each line. Those words will be excluded from Full-text search. The idea behind this option might be to reduce the size of the search dictionary by excluding words like 'the', 'a', 'in', 'of', etc.

**Compatibility:** Required. Recommended to always use 1.1 or later. 1.0 is probably for very early versions of the help file viewer that are no longer used.

**C# Filename:** Optional. For Programmers. Can be a filename, relative path or absolute path. A text file will be created and saved to the filename. The contents of the text file will be C# code that contains the list of constant references to the topics in the help file. This option is only used if ‘Create C# file’ is checked.

**C# Namespace:** Optional. For Programmers. Inside the C# text file there is a namespace declaration line that will append the specified namespace value. This option is only used if ‘Create C# file’ is checked.

**HTML Help Workshop EXE Path:** Required. The location where you installed the html help workshop. This program will also try to detect the location by checking the default installation locations.

Compile Options

**Auto compile index:** Set true to automatically create the index.hhk file. In order for this to work, the html files must contain the <object> tags with the appropriate <param> children. The <object> tags will automatically be placed in a generated .hhk file. Default value is false. The much easier way to create an index is to use the curly brackets around words. See the NüHelp.doc for more information.

**Full text search:** If false the search tab will be completely blank, i.e. buttons, results list and search text box will be invisible. There doesn't seem to be a way to provide a custom search list, so this option seems like it should always be set true if the search tab is going to be visible. Default value is true.

**Binary Index:** Certain (generally not needed) HHC Help features require this to be true to work. With a binary index, the hhc compiler automatically sorts the hhk entries, but the font attribute no longer has any effect. If set to false, then Index tab will use the order of the items in the hhk file. Default value is false.

**Binary TOC:** Set true for older MSDN/VS6 Collections and big TOCs. Must be set false for Merge, Info Types, Custom Icon and other HTML Help features to work. If true, custom font and background color in TOC won't be used. Default value is false.

**Log Compiled Filenames:** Set true to see the files listed in the compiler log output. Default value is false.

**Display Compile Notes:** Seems to do nothing. Possibly never fully implemented by Microsoft. Default value is false.

**Flatten Folders:** Don't include folders in compiled file. If true, all files will be stored in the CHM's root folder. Default value is false since generally it's a good idea to keep files organized. Warning: If you set to true and your html files contain links to folders, E.g. images\Image.png, those links will be broken.

**Support Enhanced Decompilation:** If the source files (html files, image files) of the chm file were lost, this option can make it easier to decompile the chm file. Default value is false.

**Auto TOC:** Set to an integer 0 or higher to automatically create the toc.hhc file. This uses the heading tags <h1> to <h9> in your HTML files to generate the TOC. The number is the maximum level of tag to place in the contents. e.g. 1 = , 2 = , 9. Warning: hhc.exe modifies the toc.hhc file if you specify this. The default value is -1.

General Options

**Delete temp files when finished:** Deletes the temporary files when a new projected is started, nodes are removed from the current project or the program is closed.

**Overwrite CHM file if it exists:** The previously created CHM file will be overwritten when the compile button is pressed. Otherwise a prompt appears to overwrite the file.

**Open CHM file after compile:** The CHM file will be opened for viewing after it has been created.

**Word Visible:** When Microsoft Word is used to save the document as HTML, you can choose show or hide the process of Word opening and saving the file. The default value is false.

**Create C# file:** Option for software programmers that want to create a C# code file.

**Replace Font Sizes:** In order for the Font button in the help file to work, the numeric value font sizes must be replaced with an equivalent size name, otherwise clicking the button has no effect. However, if you don’t want the font sizes to be replaced, then uncheck this option. The user can still use CTRL + mouse wheel to change the size. Here is the default mapping from size value to name value:

<= 5 : xx-small

<= 6 : x-small

<= 10 : small

<= 12 : medium

<= 14 : large

<= 16 : x-large

> 16 : xx-large

**Override H-Tags:** If checked, all pages will use the specified h-size from the dropdown box for their page titles. This creates a consistent look when switching between topics. If unchecked, then the header size and style will be the same one used in the Word document, so nested sub-topics will have titles with a smaller font size.

**Max Nested Header:** The smallest header size that will be turned into its own page. For example, if h3 is chosen, and a header h3 section contains an h4 section, then the h4 section will appear on the same page as the h3. The higher the value the deeper the TOC can be nested.

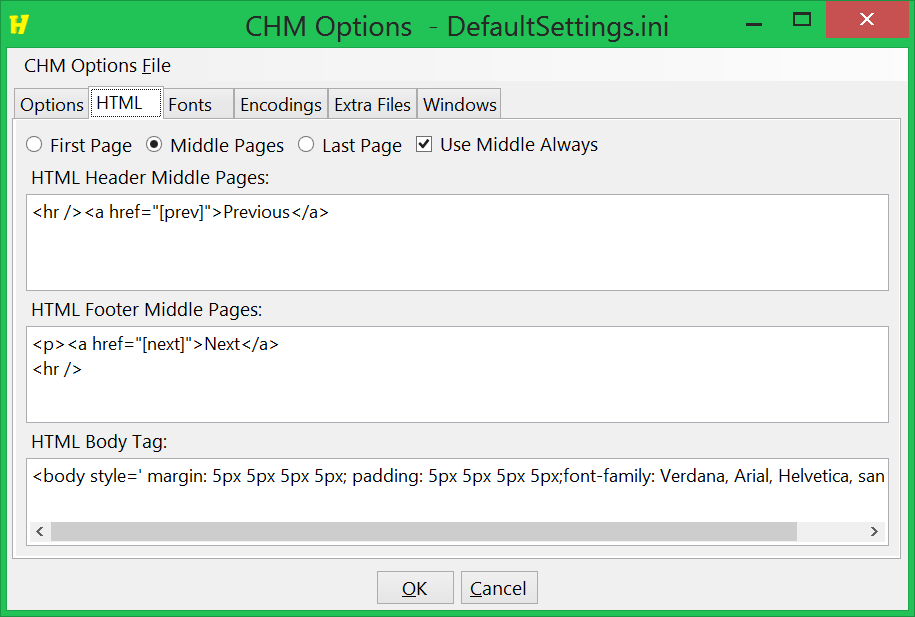
**Keywords Delim:** The delimiting character used when the Keywords feature is used. See the Keywords topic in the [NüHelp chm help file.](file://NüHelp.chm)

**File Locked Timeout (seconds):** If another application has locked an HTML file, preventing NüHelp from reading its contents, then NüHelp will pause for the specified amount of time and try to re-read the file. This value applies only when running NüHelp from the command line. The user-interface sleeps for 3 seconds, which currently does not have an option to be changed. This feature was added because it was observed that Word documents that have vector graphics images cause Word to exit but keep a lock on the file as background threads convert the vector graphics into image files.

**Blank Headers:** This option controls the behavior of how pages with blank headers are treated. A blank header is any header that does not have any visible text. For example, <h1></h1> is a blank header and so is <h1><b>&nbsp;&nbsp;</b></h1>. See the [Blank Headers](#_Blank_Headers) topic for more detailed information.

## HTML [CHM InvariantName=”HTML”]

The HTML options provide a way to add some HTML to the start, body and end of each HTML page.



**First Page / Middle Pages / Last Page:** It’s possible to use different headers/footers for the first and last pages. Note: If there’s only a single page, then the header from the first page is used, and the footer from the last page is used.

**Use Middle Always:** Option to only use the Middle Pages header/footer on all pages. If checked, the First Page and Last Page values aren’t used.

**HTML Header\*:** The html code added to the beginning of every page. The default value <hr /><a href=\"[prev]\">Previous</a> is a horizontal rule with a link to the previous topic.

**HTML Footer\*:** The html code added to the end of every page. The default value <p><a href=\"[next]\">Next</a><hr /> is a horizontal rule with a link to the next topic.

**HTML Body:** The body that is used for the start of every page. Body does not have first and last page options.

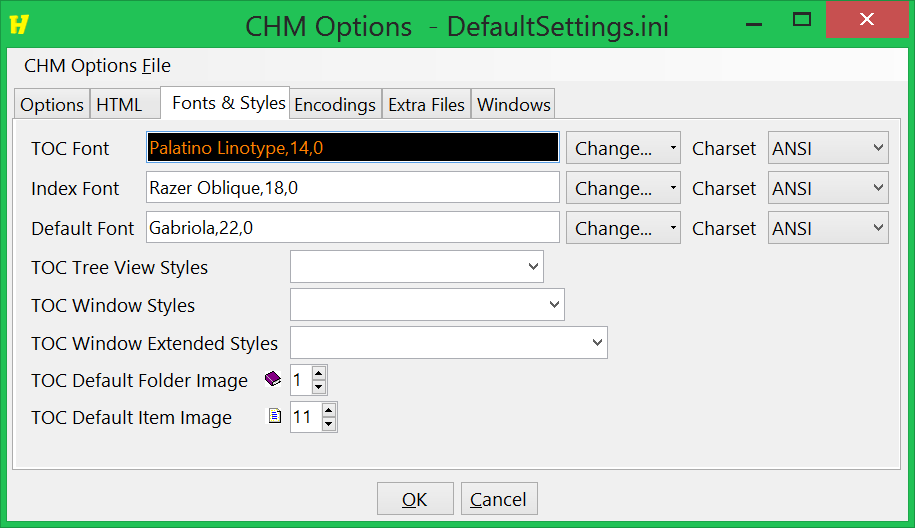
\*The Header and Footer can contain special tags, [next] and [prev] which will be substituted with the filename of the next and previous topic respectively. If you have ideas for other tags you would like then let us know. If the header and footer refer to images, then use the Extra Files tab to add those images to the project.

## Fonts & Styles [CHM InvariantName=”Fonts”]

It’s possible to change the font family and the font size of the TOC (Table of Contents) tab and the Index tab of the CHM file. The font size must be an integer value (a limitation of the CHM format). As well, only regular font style is supported. The font styles bold, italic, and underline are not supported by the CHM format.

To change back to the original system font, click the drop down arrow on the “Change…” button and click “Clear”.

For international text, you may have to change the {charset} to the appropriate Local ID.



**TOC Font**: Changes the font used in the table of contents. You can also set the background color. Unfortunately, the Foreground color option is not fully implemented by Microsoft.

**Index Font**: Changes the font used in the index tab. No color options are available for the Index tab.

**Default Font**: Changes the font used for the rest of the tabs, e.g. History tab. The default font is also used for TOC and Index if they aren’t explicitly set.

**TOC Tree View Styles**: For Programmers. The TOC is what is known as a TreeView control, which has several styles that can be changed.

**TVS\_CHECKBOXES**: A checkbox will appear beside each item. Not very useful for a CHM file.

**TVS\_DISABLEDRAGDROP**: Not applicable.

**TVS\_EDITLABELS**: The user will be able to edit the label text, however after editing, the label will change back to the original text.

**TVS\_FULLROWSELECT**: The entire row will be selected when the user clicks on an item. Cannot be used in conjunction with TVS\_HASLINES. The items will take up the entire width and the background color will not be visible.

**TVS\_HASBUTTONS**: Displays (+) and minus (-) buttons next to parent items.

**TVS\_HASLINES**: Shows dotted lines between parents and children.

**TVS\_INFOTIP**: Not applicable.

**TVS\_LINESATROOT**: Draws a dotted line between all items at the root level. Only works if TVS\_HASLINES is also set.

**TVS\_NOHSCROLL**: Disables horizontal scrolling.

**TVS\_NONEVENHEIGHT**: Not applicable.

**TVS\_NOSCROLL**: Disabled both scrollbars.

**TVS\_NOTOOLTIPS**: Not applicable.

**TVS\_RTLREADING**: Possibly only works in conjunction with Right-To-Left fonts.

**TVS\_SHOWSELALWAYS**: The selected item will remain selected when the tree view loses the focus. Otherwise clicking off the TOC will make it hard to identify the selected topic.

**TVS\_SINGLEEXPAND**: Parents only require a single click to expand, however, they automatically collapse if another item is clicked.

**TVS\_TRACKSELECT**: Moving the mouse over an item will cause the item to be highlighted and underlined.

**TOC Window Styles**: For Programmers. The only real style that applies is WS\_BORDER, which draws an addition single pixel border around the TOC. WS\_THICKNESS is neat to try, it provides a resizable TOC. In general, these do not need to be set, but are provided for completeness.

**TOC Window Extended Styles**: For Programmers. In general, these do not need to be set, but are provided for completeness.

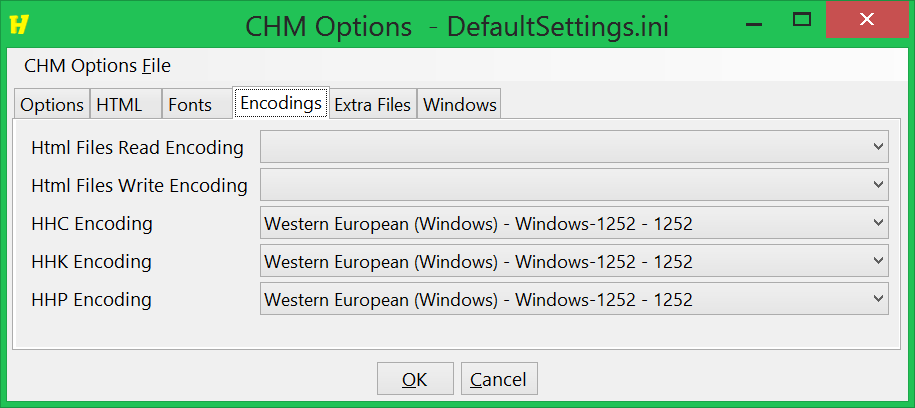
**TOC Default Folder Image**: The image to use for parents (e.g. Books or Folders) if the CHM tag does not explicitly define an ImageNumber.

**TOC Default Item Image**: The image to use for leaf nodes if the CHM tag does not explicitly define an ImageNumber.

The reason the TOC has many more options than the Index tab is because the TOC is a single TreeView control. The index tab is made up of two controls, a textbox for searching and a listbox. It’s likely that the Microsoft developers decided that the level of granularity was getting to be too much, and decided not to provide color and style settings for the other tabs.

## Encodings [CHM InvariantName=”Encodings”]

If you see {gibberish} in the TOC, title, index or pages, then you may need to try to use a different encodings. The effect of each encoding below is explained.



When converting a word document to HTML, Word seems to always use the computer’s default encoding. NüHelp has no control on the encoding that Word uses to convert the doc into HTML. This is why the default **Read Encoding** is blank. If an explicit {**Read Encoding**} is not set, then NüHelp tries to automatically detect the encoding used by parsing the HTML file for the “{charset}” meta attribute. If no charset is found, then NüHelp uses the computer’s default encoding. If you know for sure that Word is using a specific encoding, then you can specify the encoding.

Word outputs one big HTML file. This big HTML file is broken into individual files (each file becomes a page in the CHM file). When writing the individual files to disk, you can specify the **Write Encoding** to use. If no {**Write Encoding**} is specified, then the read encoding is used. If the big file contains a charset meta attribute, then each individual file will have its charset meta attribute updated to the WebName of the write encoding used.

The {**HHC Encoding**} is the encoding used to write the hhc file. The hhc file contains the names of the items in the TOC.

The {**HHK Encoding**} is the encoding used to write the hhk file. The hhk file contains the names of the items in the Index.

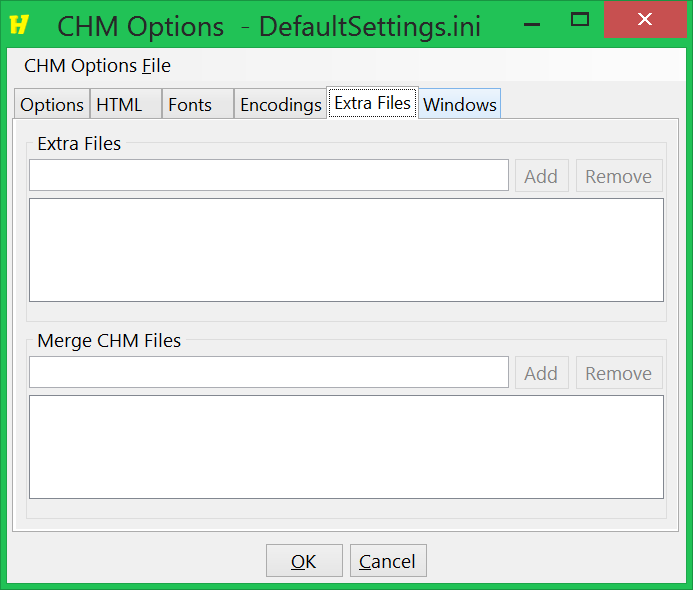
The {**HHP Encoding**} is the encoding used to write the hhp file. The hhp file contains the title that appears at the top of the Help window.

The 1252 encoding will work for Western European alphabets. However for Chinese, you will probably have to use the 936 (Chinese Simplified) encoding. It’s not quite clear on which encodings the hhc.exe compiler supports. You may also have to try setting the Language option on the Options tab.

## Extra Files [CHM InvariantName=”ExtraFiles”]

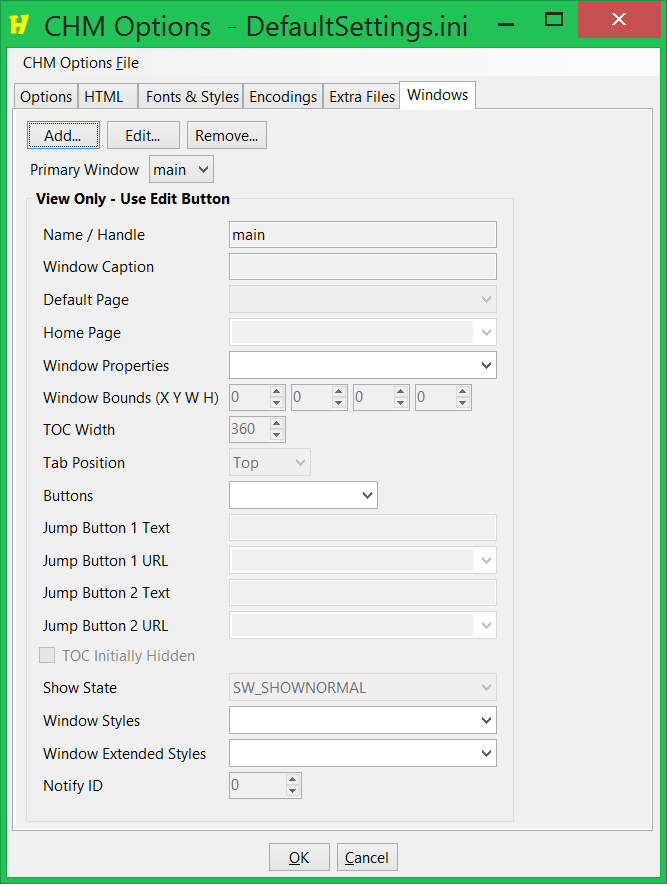
**Extra Files** is a list of files that will be copied to the Output Folder. The files will overwrite any existing filename with the same name in the output folder. Typically this is a list of images files that are referenced by the HTML header or footer or body. The filenames used should be the complete path to the file name. Any file not found is skipped.

**Merge CHM Files** is a list of other already compiled CHM files to merge into the output. At this point, it’s not clear how well this feature works.



## Windows [CHM InvariantName=”Windows”]

A Window provides the most control on the available options in the help file. You can define multiple windows, but only the primary window options are used. If no windows are defined, then default values are used that produce an all-around good result. Use the Add, Edit and Remove buttons to manage the windows. To change the values of an existing window, you must use the Edit button.



**Primary Window:** Optional. The window options that are used. If no windows are defined, then a default window that has good all-purpose settings are used.

**Name / Handle:** Required. Must be Unique. This field matches up with the primary window to determine which window to use.

**Window Caption:** Optional. The title that appears at the top of the help file window. If left blank, the name of Word document file is used.

**Default Page:** Optional. The page that appears first when the help file is opened. If left blank, the first page in the help file is used. It must be a local file, website URLs do not work here.

**Home Page\*\*:** Optional. The page that is used when the user clicks the Home button. If left blank, the first page in the help file is used. It can also be a URL, in which case it must start with http://

**Window Properties:** Defines which features are available. By default, the search tab and favorites tab are visible. Advanced search is enabled. There is a menu bar. The TOC and button bar are present. The window remembers its last position and size on screen.

**AdvancedSearch:** Display advanced search features. Only applicable if SearchTab is set.

**AutoHideShow:** Automatically hide/show TOC, Buttons and Menu Bar window.

**AutoSync:** If true, the selected item in the TOC will automatically synchronize with the content pane. Good idea to use this.

**ChangeTitle:** Put current HTML title in title bar. Seems to have no effect, the title is always used.

**CustomTab 1 to 9:** Custom tabs are possibly not implemented by Microsoft.

**FavoritesTab:** The CHM file will contain a favorites tab.

**HasMargin:** The window type has a margin. Seems to have no effect.

**HistoryTab:** The CHM file will contain a history tab.

**MenuBar:** Shows the File, Edit, View, Go, Help menu bar.

**NoContentPane:** The content pane will not be visible. Most likely an option for programmers.

**NoButtonText:** Do not display the text on the buttons. Note: In some configurations after 10 buttons, the button text may automatically be hidden.

**NoDefExStyles:** For programmers. If checked, no Window Extended Styles are assumed. If unchecked, the default Window Extended Styles are bitwise OR’ed with the Window Extended Styles below.

**NoDefStyles:** For programmers. If checked, no Window Styles are assumed. If unchecked, the default Window Styles are bitwise OR’ed with the WindowStyles below.

**NoTitleBar:** The window menu and minimize/maximize/close buttons won't appear if this is set.

**NoToolBar:** Don't display a toolbar.

**PostQuit:** For programmers. Post a WM\_QUIT message when window closes. Possibly only for notification of the help window closing.

**SearchTab:** The CHM file will contain a search tab.

**ShowTOCAndButtonBar:** If this isn't set, then only the content pane is visible.

**TopmostWindow:** Sets the window to always be on top. Some end users do not prefer this option.

**Tracking:** For programmers. Send tracking notification messages.

**UserPosition:** Remember the user position of the help window for next time it opens.

**Window Bounds:** The initial location and size of the help window when it is first opened. Leave all zero for the default values.

**TOC Width:** The width of the TOC when the help file is initially opened.

**Tab Position:** The position of the tabs. Top is the most common.

**Buttons:** Defines which buttons appear at the top of the help window. The default buttons are: Back, Forward, Home, Options, Print and Font Size. The Options button adds functionality of buttons not included.

**Back**: Adds a back navigation button. Useful.

BrowseBack: Adds a button that doesn’t seem to do anything.

BrowseForward: Adds a button that doesn’t seem to do anything.

Contents: Adds a button that doesn’t seem to do anything.

Favorites: Adds a button that doesn’t seem to do anything.

**FontSize**: Adds a button that when clicked it cycles through 5 different font sizes, going from smaller to larger. Users can also hold the CTRL key and scroll the mouse wheel.

**Forward**: Adds a forward navigation button.

**HideShow**: Adds a button that when clicked toggles the TOC’s visibility.

**History**: Same functionality as the Options button but the drop down menu only works if at least one Jump button is provided.

**Home**: Adds a home button. You can specify the URL or page displayed when the button is clicked.

Index: Adds a button that doesn’t seem to do anything.

**Jump1**: Adds a button that when clicked can navigate to a defined page but only if a MenuBar isn’t added. If a MenuBar is added, then the button turns into a link under the Options button. If the Options button isn’t added, then no Jump reference is visible. Note: you have to define a Jump1ButtonURL and Jump1ButtonText.

**Jump2**: Adds a button that when clicked can navigate to a defined page but only if a MenuBar isn’t added. If a MenuBar is added, then the button turns into a link under the Options button. If the Options button isn’t added, then no Jump reference is visible. Note: you have to define a Jump2ButtonURL and Jump2ButtonText.

**Locate**: Adds a button that when clicked finds and focuses the Content page for the item selected in the TOC.

Notes: Adds a button that doesn’t seem to do anything.

**Options**: Adds an option button that has drop down options Hide Tabs, Back, Forward, Home, Stop, Refresh, Internet Options..., Print and Search Highlight Off

**Print**: Adds a print button that when clicked brings up a print dialog.

**Refresh**: Adds a refresh button. Should only be used if the chm links to website URLs.

**Search**: Same functionality as the Options button but the drop down menu only works if at least one Jump button is provided.

**Stop**: Adds a stop button. Should only be used if the chm links to website URLs.

**TOCNext**: Jumps to the next topic in the TOC. Unfortunately it only works if binary TOC is on.

**TOCPrev**: Jumps to the previous topic in the TOC. Unfortunately it only works if binary TOC in on.

**Jump Button 1 Text\***: The text that appears on the jump button 1. Only applicable if Jump1 in buttons is checked.

**Jump Button 1 URL\*\***: The URL or page to navigate to when the button is clicked. Only applicable if Jump1 in buttons is checked.

**Jump Button 2 Text\***: The text that appears on the jump button 2. Only applicable if Jump2 in buttons is checked.

**Jump Button 2 URL\*\***: The URL or page to navigate to when the button is clicked. Only applicable if Jump2 in buttons is checked.

**TOC Initially Hidden**: This option only applies the very first time the help file is displayed. After the first time, it remembers the previous state.

**Show State**: For programmers. Specify the window state when the help window is first opened.

**Window Styles**: For programmers. Specifies low-level operating system attributes of the help window.

**Window Extended Styles**: For programmers. Specifies low-level operating system attributes of the help window.

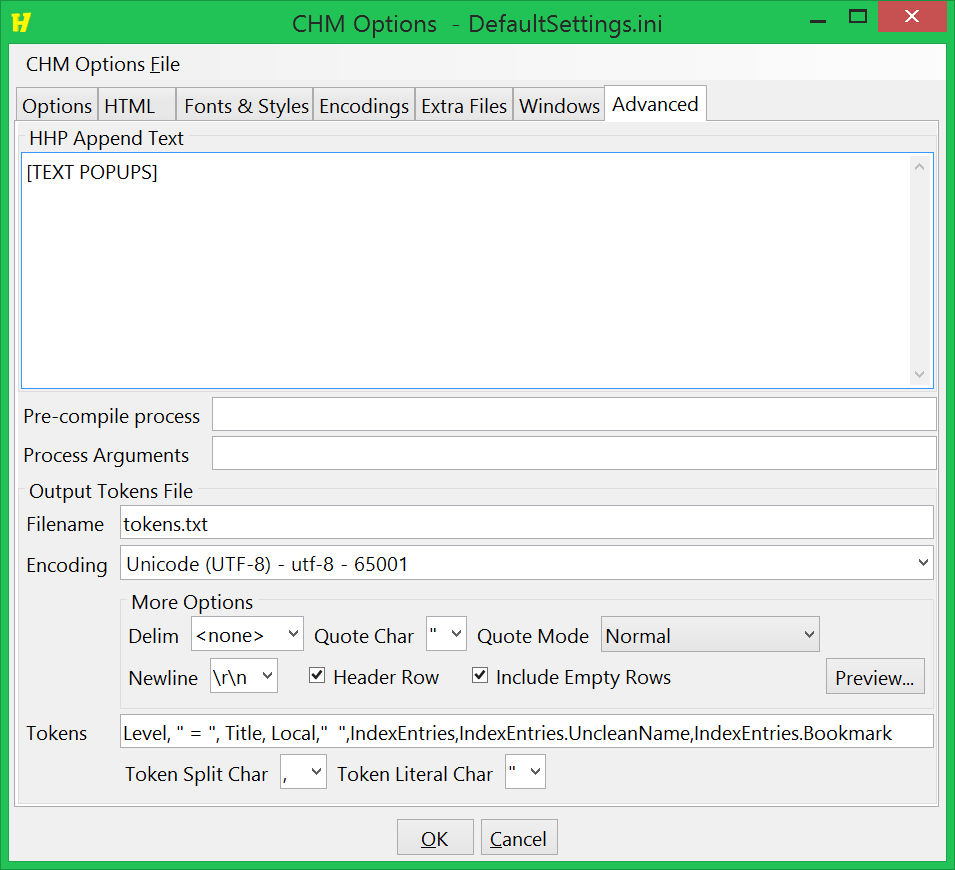
**Notify ID:** For programmers. The ID to send in WM\_NOTIFY messages.

\*The jump buttons will only be separate buttons if there is no menu-bar. As soon as there is a menu-bar then instead of buttons, menu-items are added to the drop down buttons Options and.

\*\*The Home Page and Jump button URLs can be links to websites. If so, they must start with http://. Even https:// does not work. Unfortunately the hhc.exe compiler thinks the URL is a local file and outputs a false error message. This error message can be ignored.

## Advanced Options [CHM InvariantName=”AdvancedOptions”]

The Advanced tab provides the ability to have a block of text appended to the HHP file, which is used by the htmlhelp hhc.exe compiler. This is typically for users who want to provide their own custom [MAP], [ALIAS] or [TEXT POPUPS] sections.



At the bottom, there are two more features available.

1. **Process Hook:** The ability to call a process (and pass in arguments) before the htmlhelp hhc.exe compiler is called.
2. **Output Tokens File:** The ability to extract the tokens that are defined in the Word document [CHM … ] header sections into a text file. After the compile button is clicked, first the token output file is created, and then the Process Hook is called.
   * **Filename:** The full path of the token text file.
   * **Encoding:** The encoding to use to save the file (UTF-8 will be good for most).
   * **Delim:** The delimiter character used, typically a comma, tab or pipe character.
   * **Newline:** Windows uses two characters for the newline. However, developers may find it easier to split the file if it uses \n only.
   * **Quote Char:** The character used to encapsulate a value when that value contains a special character.
   * **Quote Mode:** Select the rule to use to decide whether a value is surrounded in the quote char.
     1. Normal: Values and literals are only quoted if they contain a special character (delim char, quote char or newline).
     2. Always: Values are always quoted. Having this consistency may make it easier for some situations.
     3. Never: Never quote no matter what.
     4. NormalNeverLiterals: Same as normal mode, except that literals are never quoted.
   * **Include Empty Rows:** Option to always output a row of text for each node in the project tree, even if that node doesn’t have any tokens with a non-zero length.
   * **Header Row:** Option to have the text defined in the Tokens field appear at the top of the text file.
3. **Tokens:** The tokens to output to the specified file.
   * **Token Split Char:** The character used to split the token field into individual tokens.
   * **Token Literal Char:** The character used to indicate a value is not a token, E.g. in the screenshot above, a quotation mark character is used, and the token field contains “ = “.

We recommend users not to rely on the HHP Append text feature too heavily, and to let us know about feature requests that could apply to a broad range of users.

# Image Quality

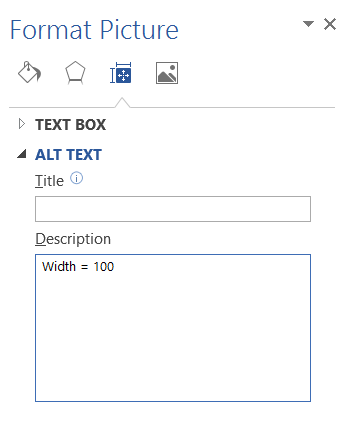
One pitfall of MS-Word is that it resizes the image data, instead of keeping the original image data and scaling it. In your help file, you may want to have smaller images (e.g. thumbnails) that when clicked become full size. This section describes how to accomplish that.

The first step is to insert the full size image as you normally would. Do not resize it. For this example, I’ve added the NüHelp logo:



Now, right click on the image, Format Picture 🡪 Layout & Properties 🡪 Alt Text

In the Alt Text box, specify the desired width of the image, for example:

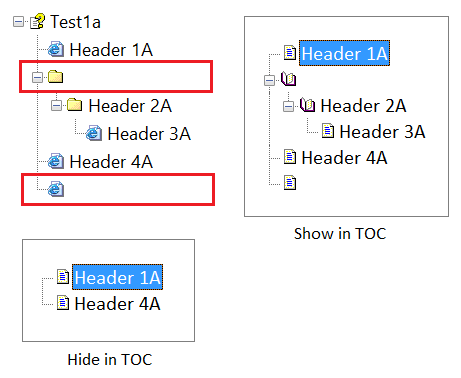


The width is separated by an equal sign and then the desired width. The width specified here is the initial width that the image will appear. When the user clicks the image, it will expand to be full size.

Any line in the Alt Text that has an equal sign will be turned into an attribute value and placed in the <img> tag attributes.

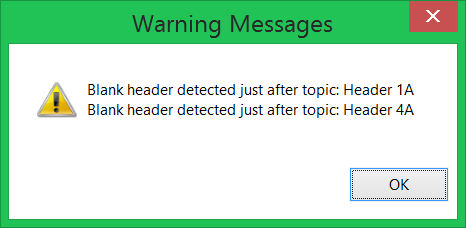
# Blank Headers

A blank header is any header that does not have any visible text. For example, <h1><b>&nbsp;&nbsp;</b></h1> is a blank header. In the following screenshot, there are two blank headers in the top left project tree, which are highlighted with a red rectangle. The two other boxes show how the TOC looks in the chm file depending on the option chosen:



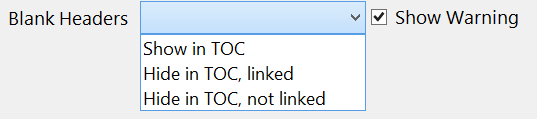
As you can see, “Show in TOC” has a blank book and a blank page, whereas “Hide in TOC” removes the blank entries. Since Header 2A and Header 3A are children of a blank header, they are also not shown.

Most of the time, a blank header is an accident, and a warning message is displayed if the “Show Warning” checkbox is checked in the Options screen.



If it is an accident, then simply edit the Word document and delete the blank header. Sometimes they are hard to find, so as the text caret moves, looks for when the Style changes to one of the Heading styles.

However, sometimes having a blank header is intentional. In this case, there are three options provided to determine how to treat them:



1. **Show in TOC:** The blank headers are displayed in the TOC. The headers and footers that contain [next] and [prev] links will link to them.
2. **Hide in TOC, linked:** The blank headers are hidden in the TOC. The headers and footers that contain [next] and [prev] links will link to them. The only way to get to these pages is either by a [next] or [prev] link, by a hyperlink from another page, or by an index entry.
3. **Hide in TOC, not linked:** The blank headers are hidden in the TOC. Pages that have a blank header, and pages that are a child of a blank header, are considered isolated pages. The header and footer used for these pages will be the first page header and the last page footer. Any [next] or [prev] links will be substituted with the pound sign character #, and thus will not navigate anywhere. The only way to navigate to these pages is by hyperlink from another page, or by an index entry. Warning: If no hyperlink or index link exists to a blank header page or a child of a blank header page, then the page will not be included in the help file.

# Temporary Files

A {temporary file} is any HTML file that was generated from converting a Word file to HTML files. Since these files are generated from a source file, they are defined as temporary.

However, if you add an HTML file to the project tree, that file is **not** considered a temporary file.

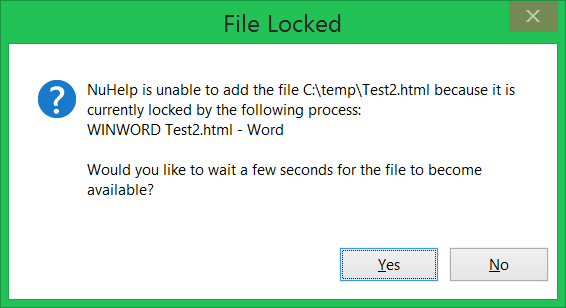
Temporary files are also the hhp, hhc, and hhk text files that are required by the compiler and are automatically generated.

# Troubleshooting

This section contains a list of problems that might occur using NüHelp and how to resolve them.

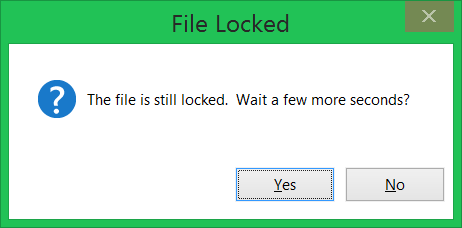
## File Locked

Some applications will exclusively lock a file, which prevents other applications from reading the contents of the file. When adding a Word document to NüHelp, it’s possible that Word may not release the lock on the file. When this happens, the following dialog box will appear:



The dialog box shows the application that is locking the file, “WINWORD” and the title of the application’s main window, which in this case is "Test2.html - Word".

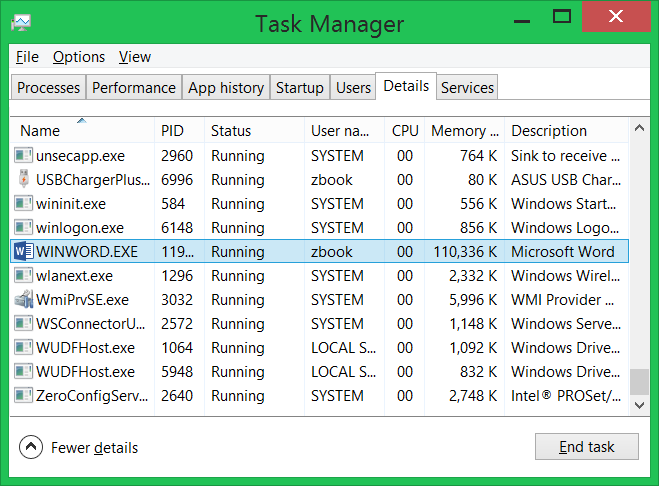
Press the Yes button and NüHelp will wait 3 seconds and try to read the file again. If the file is still locked, the following dialog box will appear:



If the file is still locked at this point, then you may have to close MS-Word. If you do not see MS-Word open, then check Task Manager by either right clicking on the taskbar and then clicking Task Manager, or by pressing Ctrl+Alt+Delete once and click on Task Manager.

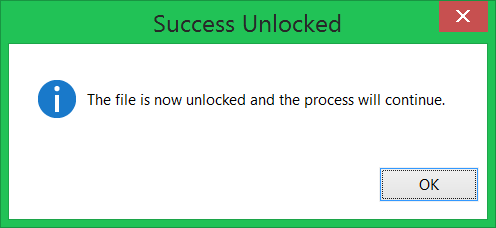


Once task manager is open, look for WINWORD.exe in the “Details” tab. If you do not see a “Details” tab, then you may have to click the “More details” button in the lower left hand corner of the Task Manager window:



Then click the “End task” button to close Word. **Please make sure all your other documents are saved first before doing this.** If you end task the wrong process then you risk losing your work.

If you successfully unlock the file, then you will see the following dialog box:

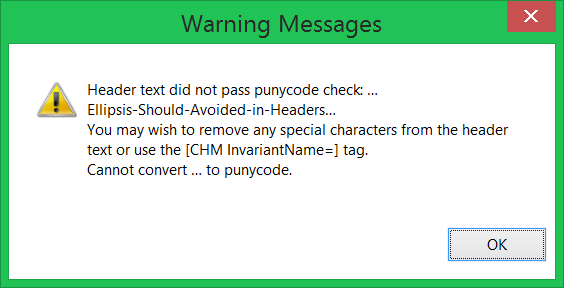


However, if you still see a “File Locked” dialog, then email [info@opulos.com](mailto:info@opulos.com) and we will try to help resolve this issue.

As a temporary workaround, you can Save As 🡪 **Web Page Filtered (\*.htm, \*.html)** in Word and then in NüHelp, instead of adding the docx file, add the html file. This will skip the conversion process and avert any related timing issues.

## Punycode Warning Message

After a file has been added using the green plus button, you may see a Warning Message dialog box, such as:



This message is saying that the header text for a particular section cannot be converted to a valid Internet filename. This usually happens when the filename contains special characters, like the three dots in a row. You should see if you can modify the header text to remove any special characters. Alternatively, you can directly specify the filename by using the [CHM InvariantName=] tag, for example:

How to … [CHM InvariantName=”HowTo”]

So then the filename used to save the html file for that section would be HowTo.html instead of How-to-….html.

## Gibberish in TOC

If you are using international characters in the header text, then make sure the following options are set properly, for example, using Chinese text:

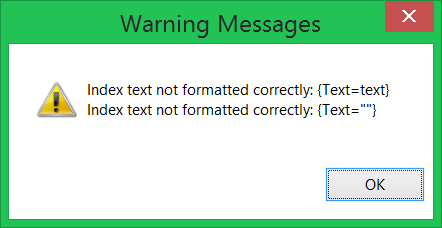
Options 🡪 Language (e.g. 0x0804 Chinese – People’s Republic of China)

Fonts & Styles 🡪 TOC Font 🡪 Charset 🡪 GB2312

Encoding 🡪 HHC Encoding 🡪 Chinese Simplified (GB2312) - gb2312 - 936

## Index Text Not Formatted Correctly

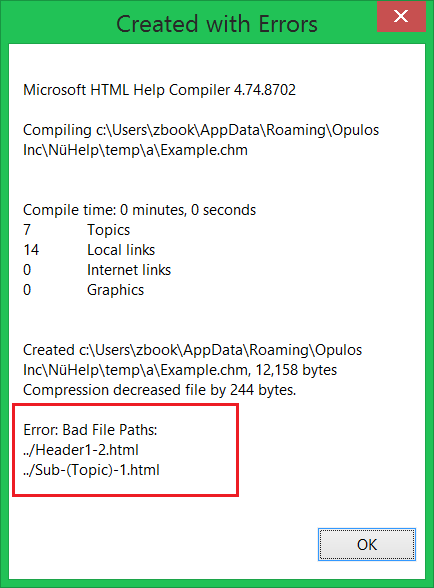
In the curly brackets, if you use the Text attribute, then the text must be non-empty and surrounded in quotation marks. For example, Text=text is not valid, but Text=”text” is valid.



## Bad File Paths

All file paths in the CHM must share the same Output Folder as their root folder. If a path begins with two dots, it means that path does not have the Output Folder in common.

Once compiled, a CHM file is like a ZIP file, there is nothing below the root. If you try to link to file paths below the folder containing the .HHP project file, then the files will be dumped into the root of the CHM. (Which can also cause filenames to clash if the same filename exists in the Output Folder as well.)



# Command Line [CHM InvariantName=”CommandLine”]

This section is for people who work with a lot of documents and want to setup an automatic way to convert them into help files.

It’s possible to use the command line to create a CHM file. These parameters can be used interchangeable and multiple times each. Square brackets denote optional items. Angle brackets indicate a value to be inputted by the user.

1. -File:<Filename>[\*<Topic>]
   1. <Filename> is the name of the html/doc/docx file to include.
   2. <Topic> is optional. Topic is the invariant name of the node to append it to. If Topic is omitted, then the file is appended at the root level.
   3. The <Filename>[\*<Topic>] can be repeated multiple times, separated by a pipe character |.
   4. If an asterisk \* is provided without a Topic, then the Filename without extension is used as the Topic.
   5. Examples:
      * -File:Doc1.doc
      * -File:Doc1.doc -File:Doc2.doc (this example will output two different CHM files, one for each document.)
      * -File:Doc1.doc|Doc2.doc (this example will add both documents to a single CHM file.)
      * -File:Doc1.doc|Doc2.doc\*TopicA|Doc3.doc (Doc1 is added, then Doc2 is added to node TopicA in Doc1. Doc3 is added to the root.)
2. -Ini:<Filename>
   1. <Filename> is the name of the compile options ini file to use.
   2. All -File: arguments use the most immediate preceding -Ini: specified.
   3. If omitted, then the DefaultSettings.ini is used.
   4. Examples:
      * NuHelp.exe -File:Doc1.doc (No ini specified, DefaultSettings.ini is used.)
      * NuHelp.exe -Ini:Options1.ini -File:Doc1.doc -File:Doc2.doc (Both documents use the Options1.ini settings.)
      * NuHelp.exe -File:Doc1.doc -Ini:Options1.ini -File:Doc2.doc (Doc1 uses the default settings, Doc2 uses the Option1.ini settings.)
3. -Path:<Path>
   1. <Path> is the absolute path to a directory. It’s used to avoid typing the same directory name again and again.
   2. Instead of typing: -File:C:\SomeLongPath\Doc1.doc|C:\SomeLongPath\Doc2.doc write: -Path:C:\SomeLongPath -File:Doc1.doc|Doc2.doc
   3. Path can be reset by using a blank argument, i.e. -Path:
4. -SuccessBox:<millis> and -ErrorBox:<millis>
   1. The number of milliseconds to display the final message box depending on if the command line was successful or had an error.
   2. Use a negative value to not display.
   3. Use a 0 value to display until the OK button is clicked.
   4. The default values are 5000 for SuccessBox and 10000 for ErrorBox.

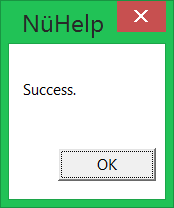
In the examples above, the filenames did not contain paths. Typically the filename will be an absolute path, e.g. C:\Some Folder\Doc1.doc. If the path contains spaces then the argument must be surrounded in quotation marks, e.g. NuHelp.exe “-File:C:\Some Folder\Doc1.doc”

You can also use relative paths, but the paths are relative to the user’s roaming folder, under \Opulos Inc\NüH­elp\. For example, -File:\test\Doc1.doc would translate to: C:\Users\**<Username>**\AppData\Roaming\Opulos Inc\NüHelp\test\Doc1.doc

The entire process will abort on error if:

1. A blank -Ini argument is used.
2. The ini file cannot be found.
3. A blank -File argument is used.
4. A specified Topic does not exist.
5. A CHM file could not be created.
6. Any unknown argument is used.

After the command line is finished, a brief message box is displayed which is automatically closed after 5 seconds. For example:



# Help File Integration

This section is intended for {programmers}.

The framework for {integrating} the help file is fairly straightforward. What you need is:

1. In your Form class, override the {OnHelpRequested} as illustrated in the next section. Even better, put the code in a general class that extends Form and have your Form extend that class. Your form also needs to implement the two interfaces, IHelpURL & IHelpFile.
2. Add the C# files ControlHelpEx.cs and HelpWindowUtil.cs to your project. These are located in the Opulos folder in the zip file.
3. After the Help file is compiled, a C# (.cs) constants file is generated in the output folder. It contains all the invariant names and a reference to the CHM file name. For every UI control that maps to a help file header, have that UI class implement the IHelpURL interface, and then have the HelpURL property return the appropriate constant from the C# generated file.
4. The HelpFile property can return the CHMFile constant from the C# generated file. Most of the time, you will distribute the CHM file in the same folder as your program’s EXE file, so it does not require an absolute path.

## Invariant Names

The reason you should use the [CHM InvariantName=””] is if you ever intend to translate your program to different languages and have the help file still navigate to that section. The {InvariantName} will stay the same, but the Header Name will be translated.

## How It Works

Suppose you coded a UI control and you want to write help documentation for it. Your UI control must implement the IHelpURL interface and return the constant that refers to that section.

When the {F1 key is pressed}, the Opulos code provided in the zip file does the work of determining the focused control, and scanning up the control’s parents until the first parent that implements the IHelpURL interface, which will be your UI control. The URL is then passed to the .Net Help.ShowHelp(…) static method, and voila, the help file will open directly to the page talking about the UI control!

The following code is a template on how to structure your code. As you can see, a general class Form2 extends Form. Then your form would extend Form2, and implement IHelpURL & IHelpFile. More commonly, you will have classes that extend Panel that implement the IHelpURL interface.

IndexingOff

public static class HelpConstants {

public static readonly String CHMFile = "QuickGuide.chm";

public static readonly String MainScreen = "MainScreen.html";

public static readonly String Options = "Options.html";

}

public class Form2 : Form {

DateTime LastHelpEvent;

protected override void OnHelpRequested(HelpEventArgs hevent) {

// pressing F1 when a WebBrowser control is focused causes

// the OnHelpRequested to be fired twice

TimeSpan ts = DateTime.Now - LastHelpEvent;

if (ts.TotalSeconds < 1)

return;

LastHelpEvent = DateTime.Now;

if (hevent != null) {

if (hevent.Handled)

return;

hevent.Handled = true;

}

this.BeginInvoke((Action) delegate {

// a corrupt ieframe.dll can cause a memory access violation exception.

// thus, wrap the call to ShowHelp() in a try-catch so the entire program

// doesn't crash.

try {

Control c = ControlHelpEx.FindFocusedControl(Form.ActiveForm) ?? this;

c.ShowHelp();

} catch (Exception ex) {

MessageBox.Show(this, ex.Message, "Help Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

});

}

}

public class MyProgramForm : Form2, IHelpURL, IHelpFile {

public String HelpURL {

get {

return HelpConstants.MainScreen;

}

}

public String HelpFile {

get {

return HelpConstants.CHMFile;

}

}

}

IndexingOn