



HelpNDoc User Manual

Table of contents

Welcome to HelpNDoc	13
Getting started with HelpNDoc	13
Introduction	14
About HelpNDoc	14
System requirements	14
Getting help	15
HelpNDoc editions and licenses	16
HelpNDoc license agreement	17
What's new in HelpNDoc 9	20
How to buy HelpNDoc	21
Overview of the user interface	22
File menu	24
Ribbon tabs	24
Styles editor	26
Find and replace window	27
Options window	29
Quick start guides	33
Launching HelpNDoc	34
Create a new project	34
Adding topics	34

Setting up topic content	34
Generating documentation	34
Writing documentation	35
Create a new project	36
New project wizard	37
Open an existing project	37
Import other formats	38
Split imported documents	39
Import folders	40
Custom hyperlink types	40
Project options	41
Date and time format settings	42
Managing the table of contents	45
Create topics	47
Delete topics	48
Rename topics	48
Move topics	49
Change topic properties	49
Using the topic editor	52
Topic kind	52
Headers and footers	52
Font properties	53

Paragraph properties	54
Working with styles	56
Working with hyperlinks	57
Link to a specific topic	58
Link to a relative topic	59
Link to an Internet or e-mail address	60
Link to a file	61
Link to a counter	62
Working with tables	62
Working with pictures	63
Working with the image map editor	63
Using the library	64
Import files dialog	66
Folder library item	67
Barcode library item	68
Barcode editor	69
Counter library item	70
Document library item	74
Dynamic content library item	77
Equation library item	79
HTML code library item	81
Image map library item	82

Image map editor	84
Movie library item	85
Picture library item	89
Image editor	94
Snippet library item	95
Snippet editor	97
Variable library item	98
Using the keywords editor	99
Manage keyword association	100
Using the spell checker	101
Publishing documentation	103
Template settings	104
Customize documentation formats	105
CHM documentation settings	106
HTML documentation settings	107
Inline table of contents	110
Setup Google tag manager	111
Word documentation settings	112
PDF documentation settings	112
ePub documentation settings	113
Kindle / Mobi documentation settings	114
Qt help documentation settings	115

Markdown documentation settings	116
Override library items	117
Override styles	119
Sign documents	121
Sign Word documents	121
Sign PDF documents	123
Encrypt documents	125
Encrypt Word documents	125
Encrypt PDF documents	126
Build actions	128
Build actions samples	132
Call an API to display the current date time using build actions	132
Copy files using the run program action	133
Compress output using the run program action	133
Show a warning message using build actions	134
Store last generation date and time using build actions	134
Advanced usages	135
Keyboard shortcuts	135
Keyboard auto-completion	140
Customizing keyboard shortcuts	142
Topic status	142
Conditional content generation	143

Analyzing a project	145
General information	146
Visualize the project's structure	147
Analyzing hyperlinks	148
Analyzing anchors	149
Analyzing library items	150
Analyzing keywords	152
Analyzing conditions	153
Analyzing spelling	154
Vacuuming a project	155
Working with templates	155
Using the template editor	155
HTML based templates	157
General settings	157
Variables	158
Script files	159
Assets	161
Editing assets	163
HTML tags	163
Hooks script	165
Word and PDF templates	166
Text layout	167

Low-level template details	168
Best practices	168
Template configuration file	170
Template inheritance	171
Code templates	172
CHM and HTML templates	172
Handle the generated topic links	173
Methods available in templates	173
Generate multiple files from a single template file	173
Template variables	174
Assets	176
HTML tags	177
Samples	178
Building a single page HTML template	178
Template specifics	182
Default HTML template: JavaScript on page change	183
Usage from the command line	183
Legacy command line syntax for 5.3 and older	189
Customize default project styles	191
Using the Script Editor	192
Object pascal subset	192
Base types	194

Built-in functions	197
HelpNDoc API methods	200
Migrating scripts and templates	244
Migrating scripts from V4 to V5	244
Migrating scripts from V5.0 to V5.1	245
Migrating scripts from V6 to V7	246
Using the integrated web server	247
Backing up projects	249
Documentation formats specifics	249
CHM files and programming languages	249
.NET (C#) integration	250
Delphi integration	250
Older versions of Delphi	250
Java integration	251
Microsoft Access integration	251
Visual Basic integration	251
WinDev integration	251
HTML help URL parameters	252
Context sensitive HTML help	253
License key management	254
Named licenses	254
Grace period	255

Floating licenses	255
Activating the floating license server	258
Check the floating license server's status	259
Check that the floating license server is listening to incoming connections	259
Check the connection to the server from HelpNDoc	260
How to handle dead leases	260
Listen to HTTPS communication	261
Deactivating the floating license server	264
FAQ and troubleshooting	265
Error and warning messages	265
HelpNDoc shows an exception when launched	265
Project already opened by another software	266
Project is too old to open	266
Help compilers	267
What compilers or libraries do I need to install?	267
Installing the Microsoft HTML Help Compiler displays a warning message? ...	267
CHM and HTML help	268
The CHM viewer indicates that the page cannot be displayed	268
CHM content is not displayed after Internet Explorer update	268
Despite modifying the navigation pane's width the CHM file is not updated ..	269
The search feature is not working in the CHM documentation	269
Google Chrome shows an error when searching HTML documentation	270

The HTML help is broken when hosted by CloudFlare	270
Missing files when generating a CHM file in the same directory as HTML	271
The HTML documentation is not loading or behaving incorrectly	271
Table of contents is empty or loading in default HTML template	272
HTML documentation hosted on GitHub are broken	273
Windows reserved file names	273
PDF documentation	273
Adobe Reader won't print with "drawing error" message	274
Microsoft Word documents	274
Table of contents page numbers are wrong in Word documents	274
Qt Help	274
Modifications made to a Qt help file are not updated in assistant.exe	274
Sales and license information	275
What is HelpNDoc's update policy?	275
How much does HelpNDoc costs	275
Do you provide a discounted Educational license ?	275
Do you provide a government license ?	275
I need a special license: site license or global license?	275
What kind of payment devises and currencies do you accept?	276
How can I request a written quote before ordering?	276
Miscellaneous	276
HelpNDoc download problem	276

Doc or DocX files can't be imported	276
Some panels are missing or HelpNDoc's Window is hidden	277
The table of contents is missing topics or behaving strangely	277

Welcome to HelpNDoc



HelpNDoc

HelpNDoc is an **easy to use yet powerful and intuitive** help authoring environment which provides a clear and efficient user interface to build the most amazing **CHM** help files, responsive **WEB** based documentation, **PDF** and **Word** documents, **ePub** and **Kindle** eBooks, **Qt Help** files as well as **Markdown** files from a single source without worrying about the inner working of help file generation.

This help documentation is designed so you can quickly learn HelpNDoc as a new user or enhance your knowledge as a regular user:

- [Introduction](#)
- [Overview of the user interface](#)
- [Quick start guides](#)
- [Writing documentation](#)
- [Publishing documentation](#)
- [Advanced usages](#)
- [FAQ and troubleshooting](#)

Getting started with HelpNDoc

New to HelpNDoc

- Read the [Introduction](#) section to know more about HelpNDoc, its different editions and system requirements.
- Follow the [Quick Start Guides](#) to familiarize yourself with the processes of creating and generating your documentations.

Regular user of older HelpNDoc versions

- Read the [What's new in HelpNDoc 9](#) section to have a quick look at major changes.
- Run through the [Quick Start Guides](#) to familiarize yourself with the new version.

Introduction

- [About HelpNDoc](#)
- [System requirements](#)
- [Getting help](#)
- [HelpNDoc editions](#)
- [HelpNDoc license agreement](#)
- [What's new in HelpNDoc 9](#)
- [How to buy HelpNDoc](#)

About HelpNDoc

HelpNDoc is an easy to use yet powerful and intuitive help authoring environment.

HelpNDoc provides a **clear and efficient** user interface to build the most amazing CHM help files, WEB based documentation, PDF and Word documents, ePub and Kindle eBooks, Qt Help files as well as Markdown files from a single source without worrying about the inner working of help file generation. You just have to enter or import your documentation in the **built-in word processor** and hit the "[Compile](#)" button to obtain a fully functional help file which looks exactly as you designed it.

Forget about bloated user interfaces and incomprehensible tools. HelpNDoc has been engineered to provide the most [advanced functionalities](#) in their simplest form: creating and maintaining HTML help files, Word and PDF documentation is usually a painful process but thanks to HelpNDoc you may surprise yourself enjoying it!

You know how to use your favorite word processor, so you already know how to use HelpNDoc: it's that easy! Add to that many powerful features such as live spell checking in a fully **WYSIWYG** (What You See Is What You Get) environment and you'll begin to imagine how fast and easy it will be for you to create your next help file and how professional it will look like.

System requirements

HelpNDoc's recommended system configuration includes:

- Windows 10 or Windows 11
- 512MB of RAM
- 230MB of free disk space
- 1024x768 screen resolution or higher

- Help Compiler: [Microsoft HTML Help Workshop](#)
- MobiPocket/Kindle Compiler: [Amazon KindleGen](#)
- Qt Help Compiler: [Qt Framework](#)
- Microsoft Edge WebView2 Runtime (already installed on updated versions of Windows):
[WebView2 Runtime](#)

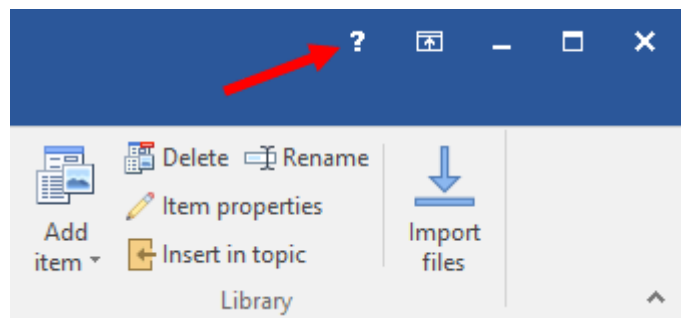
Getting help

This help file can either be viewed on-line or off-line when installed with HelpNDoc. You can obtain the latest version as well as other formats of this help file on line at

<https://www.helpndoc.com/online-help>

Off-line access

The off-line help file is part of the HelpNDoc installation. To launch it, either press the F1 key or click the help button at the top right of HelpNDoc's main windows.



On-line access

To access and view the most recent HelpNDoc's help file on-line, launch a web browser to the following URL: <https://www.helpndoc.com/documentation/html/index.html>



Printing the help file

Alternatively, you can download and print a PDF or Word version of HelpNDoc's documentation from the following URL: <https://www.helpndoc.com/online-help>

HelpNDoc editions and licenses

Three editions of HelpNDoc are available based on your needs:

- [HelpNDoc Ultimate Edition](#): Fully functional licensed edition, which can export banner-free CHM, HTML, Word, PDF documentation, ePub and Kindle eBooks, Qt Help files and Markdown files. Includes all features;
- [HelpNDoc Professional Edition](#): Licensed edition which can export banner-free CHM, HTML, Word, PDF documentation, ePub and Kindle eBooks, Qt Help files and Markdown files. Some features missing;
- [HelpNDoc Standard Edition](#): Licensed edition, which can export banner-free CHM and HTML documentation only. Some features missing;
- [HelpNDoc Personal Edition](#): This edition is completely free for personal use only and adds a small banner at the bottom of all the generated documentation formats. Some features missing;

HelpNDoc Ultimate Edition

- Can be used for commercial purposes;
- Exports to all the formats handled by HelpNDoc without any banner;
- Includes all features
- No spy-ware, viruses or any kind of malware;
- [More information, price and secure order links...](#)

HelpNDoc Professional Edition

- Can be used for commercial purposes;
- Exports to all the formats handled by HelpNDoc without any banner;
- Some features missing
- No spy-ware, viruses or any kind of malware;
- [More information, price and secure order links...](#)

HelpNDoc Standard Edition

- Can be used for commercial purposes;
- Exports to CHM and HTML formats without any banner;
- Exports to PDF, Word, ePub and Kindle eBooks, Qt Help files, Markdown files with a small banner at the bottom of the generated documents;
- Some features missing
- No spy-ware, viruses or any kind of malware;

- [More information, price and secure order links...](#)

HelpNDoc Personal Edition

- Can't be used for commercial purposes or in exchange of any kind of compensation;
- Exports to all the formats handled by HelpNDoc with a small banner at the bottom of the generated documents;
- Some features missing
- No spy-ware, viruses or any kind of malware;
- [Download the free Personal Edition of HelpNDoc...](#)

HelpNDoc Licenses

For commercial use of HelpNDoc, it is possible to choose between:

- A named (per-seat) license: this license can only be used by a single named person and installed on his computer
- A floating license: this license can be shared between multiple people with the limit of one person using it at the same time per purchased license
- [More information, price and secure order links...](#)

See [HelpNDoc's Store page](#) to learn more about available editions and licenses.

IBE SOFTWARE HelpNDoc End User License Agreement

IMPORTANT: THIS SOFTWARE END USER LICENSE AGREEMENT ("EULA") IS A LEGAL AGREEMENT BETWEEN YOU AND IBE SOFTWARE. READ IT CAREFULLY BEFORE COMPLETING THE INSTALLATION PROCESS AND USING THE SOFTWARE. IT PROVIDES A LICENSE TO USE THE SOFTWARE AND CONTAINS WARRANTY INFORMATION AND LIABILITY DISCLAIMERS. BY INSTALLING AND USING THE SOFTWARE, YOU ARE CONFIRMING YOUR ACCEPTANCE OF THE SOFTWARE AND AGREEING TO BECOME BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO BE BOUND BY THESE TERMS, THEN SELECT THE "CANCEL" BUTTON. DO NOT INSTALL THE SOFTWARE AND RETURN THE SOFTWARE TO YOUR PLACE OF PURCHASE FOR A FULL REFUND.

THIS EULA SHALL APPLY ONLY TO THE SOFTWARE SUPPLIED BY IBE SOFTWARE HERewith REGARDLESS OF WHETHER OTHER SOFTWARE IS REFERRED TO OR DESCRIBED HEREIN.

DEFINITIONS

(a) "HelpNDoc" and "Software" refers to IBE Software's HelpNDoc program, in each case, supplied by IBE Software herewith, and corresponding documentation, associated media, and online or

electronic documentation.

(b) "IBE Software" means IBE Software.

(c) "Free Version" or "Freeware Version" or "Freeware Edition" or "Personal Edition" means a free version of the Software for personal use only, so identified, to be used only for non-profit projects. The Free Version is fully functional, without restrictions of any kind but may contain messages in the end product stating that they have been created using the Free Version of the Software.

(d) "Registered Version" means a version which has been bought to IBE Software.

(e) "Educational Version" means a version which has been bought to IBE Software by an educational institution and may only be provided to students and employees of the institution. The Educational Version may have limited functionalities and/or usage restrictions.

LIABILITY DISCLAIMER

THE HELPNDoc PROGRAM IS DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. NEITHER THE AUTHORS NOR IBE SOFTWARE WILL BE LIABLE FOR DATA LOSS, DAMAGES AND LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

RESTRICTIONS

You may not use, copy, emulate, clone, rent, lease, sell, modify, decompile, disassemble, otherwise reverse engineer, or transfer any version of the Software, or any subset of it, except as provided for in this agreement. Any such unauthorized use shall result in immediate and automatic termination of this license and may result in criminal and/or civil prosecution.

FOR HELPNDoc FREE VERSION ONLY

(a) Any Help File or associated intermediate files generated by HelpNDoc Free Version MUST NOT be used for, or in relation with, any commercial or business purpose, whether "for profit" or "not for profit". Any work performed or produced as a result of use of this Software cannot be performed or produced for the benefit of other parties for a fee, compensation or any other reimbursement or remuneration.

(b) The HelpNDoc Free version may be freely distributed, with exceptions noted below, provided the distribution package is not modified in ANY WAY.

(c) The HelpNDoc Free version may not be distributed inside of any other software package without written permission of IBE Software.

(d) The HelpNDoc Free version allows the user to publish its work according to the license agreement, but nor IBE Software nor any member of the company can be held liable for the content of the publication.

FOR HELPNDOC REGISTERED VERSION ONLY

(a) Single-User (per seat) Licenses: You may install and use the Software on a single computer to design, develop, and test the Software's output. Installation on a second computer, such as a laptop and a desktop computer, is permitted if it is guaranteed that you are the exclusive user of both computers.

(b) Multiple-User (floating) Licenses: You may install and use the enclosed Software on a server to design, develop, and test the Software's output. Use of the Software is limited by the number of floating licenses owned. Only one user per floating license owned may use the software at the same time.

(c) The HelpNDoc Registered version allows the registered user to publish its work according to the license agreement, but nor IBE Software nor any member of the company can be held liable for the content of the publication.

(d) The HelpNDoc Registered version guaranties to the registered user free updates for a whole version cycle and for at least 12 (twelve) months.

FOR HELPNDOC EDUCATIONAL VERSION ONLY

(a) You may install and use the Software on a single computer; OR install and store the Software on a storage device, such as a network server, used only to install the Software on your other computers over an internal network, provided you have a license for each separate computer on which the Software is installed and run. A license for the Software may not be shared, installed or used concurrently on different computers.

(b) The Software may be used on a single computer solely for individual and personal "technology enthusiast" purposes, personal education and study (including educational-related research), or administrative use in support of the educational institution. It may not be used for any commercial or business purpose, whether "for profit" or "not for profit." Any work performed or produced as a result of use of this Software cannot be performed or produced for the benefit of other parties for a fee, compensation or any other reimbursement or remuneration.

(c) The HelpNDoc Educational version allows the registered user to publish its work according to the license agreement, but nor IBE Software nor any member of the company can be held liable for the content of the publication.

(d) The HelpNDoc Educational version guaranties to the registered user free updates for a whole version cycle and for at least 12 (twelve) months.

TERMS

This license is effective until terminated. You may terminate it by destroying the program, the documentation and copies thereof. This license will also terminate if you fail to comply with any terms or conditions of this agreement. You agree upon such termination to destroy all copies of the program and of the documentation, or return them to the author.

OTHER RIGHTS AND RESTRICTIONS

All other rights and restrictions not specifically granted in this license are reserved by authors.

What's new in HelpNDoc 9

You can check the detailed release history in [HelpNDoc's what's new](#) page.

New FTP(S) and SFTP build actions

HelpNDoc 9 introduces new FTP-related build actions, allowing users to directly upload their generated documentation files to FTP(S) and SFTP servers. This feature streamlines the process of distributing and publishing documentation by enabling automated, secure file transfers to web hosts or other online platforms supporting these protocols. See: [Build actions](#)

Library items overrides: Customize them for each build individually

HelpNDoc now allows users to override all types of library items (such as pictures, documents, equations, variables, snippets...) for each individual build. This enables the creation of an unlimited number of variations of the main documentation project. See: [Override library items](#)

New library item type: Dynamic content

Create scripts that dynamically generate HTML-based content, which can be seamlessly integrated within the topics of your project. These scripts have access to the full suite of [HelpNDoc's API methods](#) to produce dynamic content. See: [Dynamic content library item](#)

Completely rewritten HTML importer

The enhanced HTML importer in HelpNDoc is now quicker and more dependable, offering better support for modern HTML and CSS features. This improvement ensures a smoother, more accurate import of web content into HelpNDoc projects, facilitating easier integration and better formatting consistency. See: [Import other formats](#)

Greatly improved PDF generator

The updated HelpNDoc PDF generator now produces smaller PDF files with optimized font embedding settings, and handles symbol fonts more effectively. Other PDF improvements include enhanced drawing accuracy for various elements and a fix for over-sized images, ensuring better layout and visual quality.

Improved generation of HTML-based documentation formats

HelpNDoc 9 includes improved HTML/CSS generation for better modern style support and fixes issues with exporting bullets with font symbols to HTML formats. It also enhances HTML code clarity for ePub and Kindle eBooks, reducing validation errors and ensuring smoother compatibility.

Some other improvements and bug fixes

Many bugs are fixed with each release as indicated in the "[What's new](#)" page. Some of them include:

- Added ability to cancel the generation process at any stage
- Improved HTML library item's user interface with ability to load and save content to the hard drive
- Projects styles can be imported from an existing DocX document or HTML web page
- Redesigned script editor's user interface with better logging support; ability to quickly locate info, warning, errors and more...
- Added ability to copy an overridden build style to another build
- Improved Markdown import / export capabilities
- And [much more...](#)

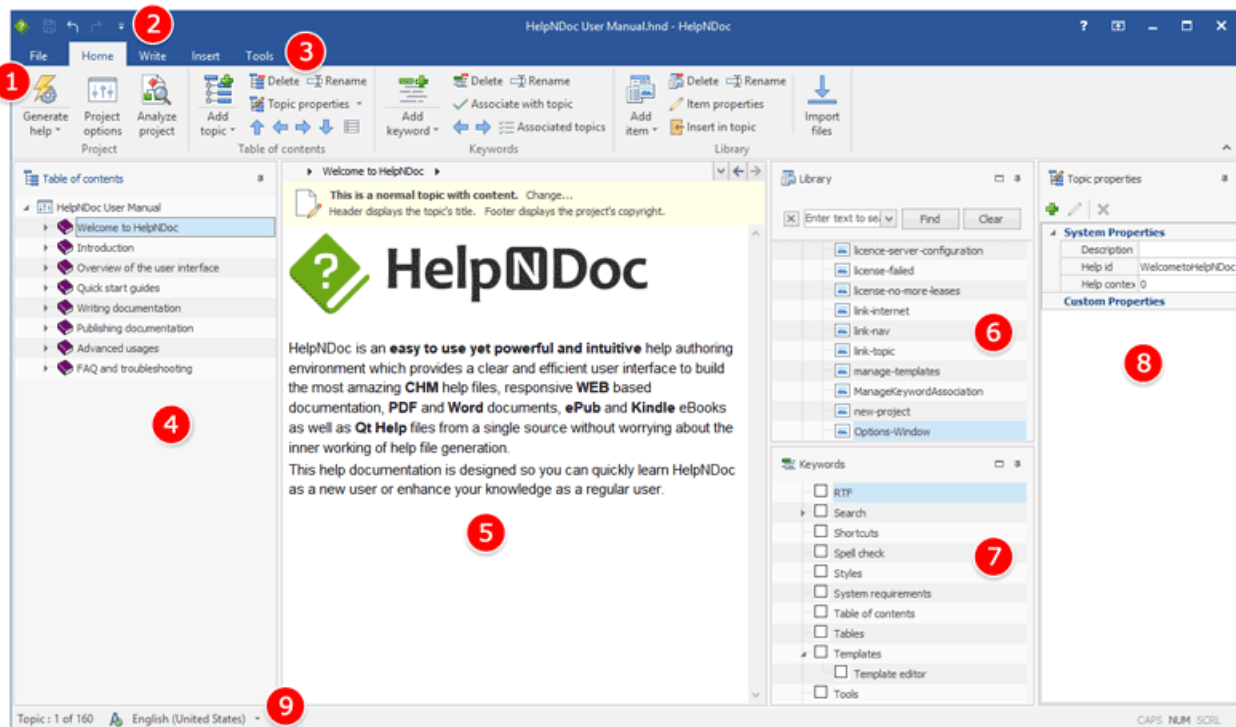
How to buy HelpNDoc

HelpNDoc can be purchased worldwide, either on-line or off-line, and paid using various payment methods (Credit Cards, Check, PayPal...) and currencies (US Dollars, Euros...). As soon as the

transaction is complete, you will receive instructions on how to obtain the full version of HelpNDoc.

To get more information on the order process and purchase HelpNDoc, launch your web-browser to the HelpNDoc store page at <https://www.helpndoc.com/store>

Overview of the user interface



1. File menu

- Manage projects: create new, open existing, save...
- Access to recent projects and places
- Access the application options
- Access to help and resources on HelpNDoc
- Exit the application

2. Quick access tool-bar

- Access to frequently used actions such as "Save project", "Undo" and "Redo"

3. Ribbon tool-bar

- Contains all actions available within HelpNDoc
- Can be minimized to provide greater documentation editing screen estate

4. Table of contents

- Define and manage the topics hierarchy for the currently opened project
- Root topic is the project topic, used to view and modify project settings
- Selecting a topic will display its associated content for editing

5. Topic editor

- Used to edit the selected topic's content
- Setup the topic's source and behavior

6. Library

- Define and manage the multimedia and reusable items such as images, movies, snippets, included documents...
- Add items to topics

7. Keywords editor

- Define and manage the keywords hierarchy for the currently opened project
- Associate keywords with individual topics

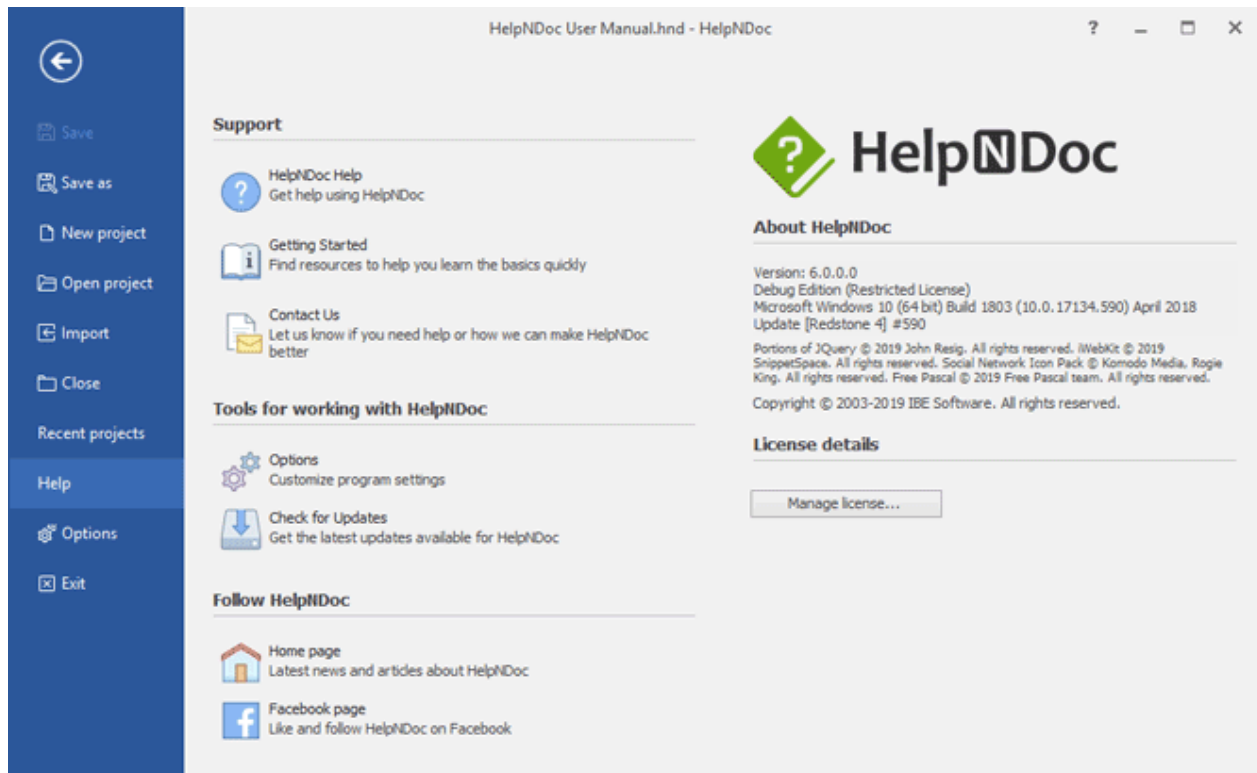
8. Topic properties

- Define topic's system properties
- Manage topic's custom properties

9. Status bar

- Get stats about your documentation
- [Manage dictionaries and spell checker options](#)
- Get information about keyboard status

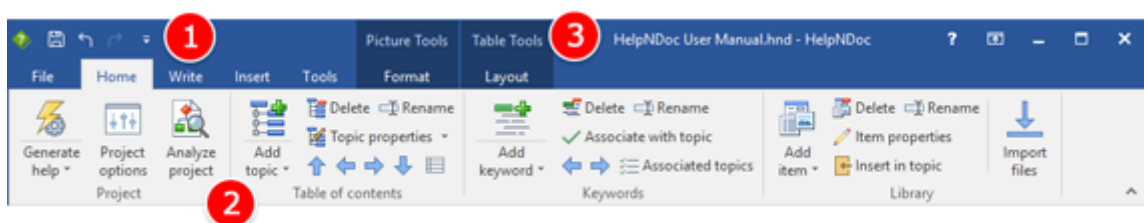
File menu



The HelpNDoc's file menu can be displayed by clicking the "File" button at the top left of the main window. It is used to:

- "Save" or "Save as" a project
- [Create a new project](#) or [open an existing one](#)
- [Import a project in another format](#) such as CHM, HLP, HTML, Word, ePub files
- Close the currently opened project
- Access to recent projects and locations
- [Access help](#) and product information
- Manage the [license](#)
- Access the [options dialog](#) and exit the application

Ribbon tabs



Presentation

The HelpNDoc's ribbon tabs are located at the top of the main window and provide all the features available within HelpNDoc in a categorized fashion. The ribbon tabs parts are:

1. The main tabs - They are always visible and are used for the most important actions
2. The tabs groups - When a tab is selected, it will display actions grouped by similar purpose
3. Contextual tabs - Those tabs are only shown when needed. For example, the "Picture Tools" tab is only visible when a picture is selected

The Home tab

This tab provides access to the basic actions:

- Generate the documentation and change the project options
- Manage the [table of contents](#) and topic properties
- Manage the [keywords hierarchy](#) and association
- Manage the [library](#) and import library items

The Write tab

This tab gives access to actions needed to manage and format the topic editor's content:

- Copy and paste text
- Manage [font](#) and [paragraph](#) properties
- Use and manage [styles](#)
- [Find and replace](#) content throughout the project

The Insert tab

This tab provides access to inserting and importing actions:

- Insert a picture, movie, document, HTML code, variable, snippets...
- Insert a table, symbol, horizontal line, page break, condition...
- Insert or Edit hyperlinks and anchors

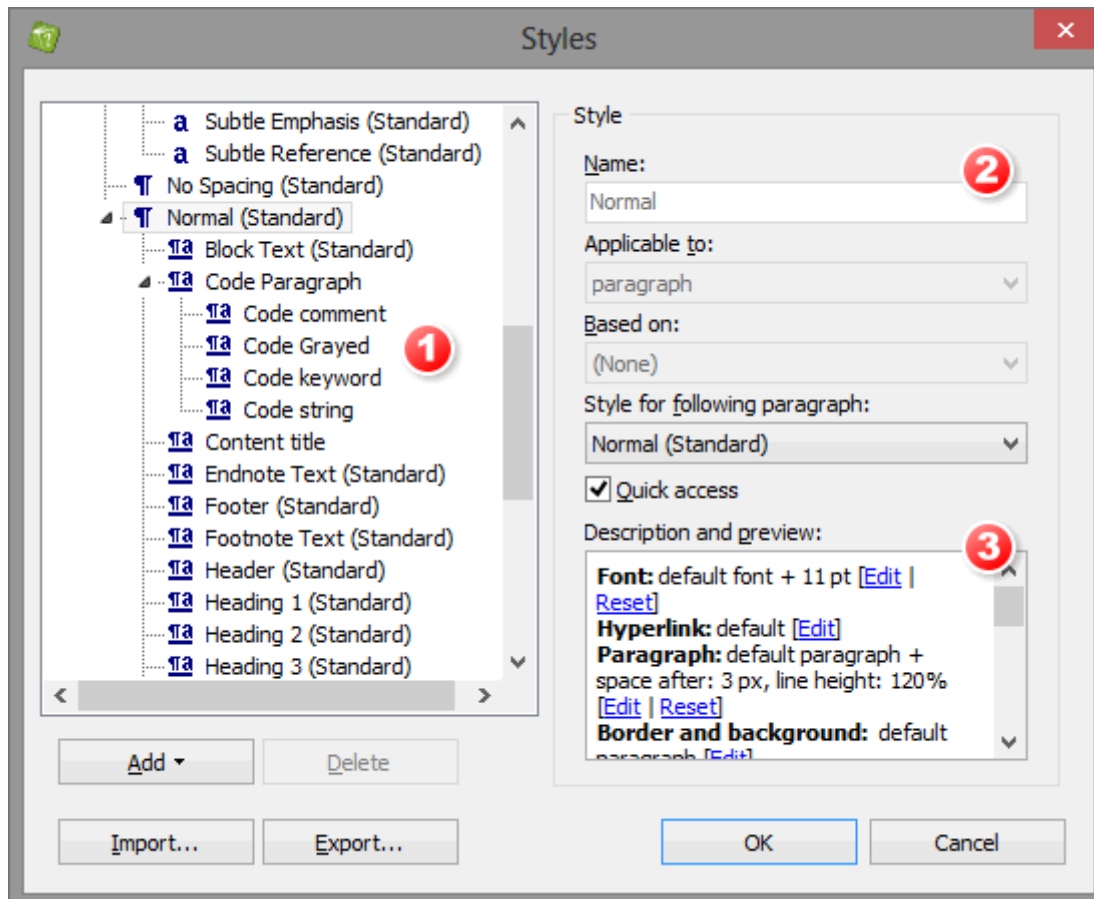
The Tools tab

Access to various tools to manage HelpNDoc or the currently opened project:

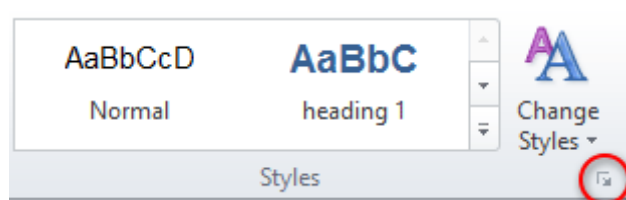
- Transform text
- Syntax highlighter

- [Edit and run scripts](#)
- [Vacuum the project](#)
- [Template Editor](#)
- Launch the [integrated web server](#)

Styles editor



The style editor is where style are created, customized and organized. The styles editor can be accessed via the arrow at the bottom right part of the "Styles" group in the "Write" ribbon tab.



Each style added in the style editor can be used throughout the project to format texts, paragraphs and links. Styles inherit from their parents any properties they do not explicitly define.

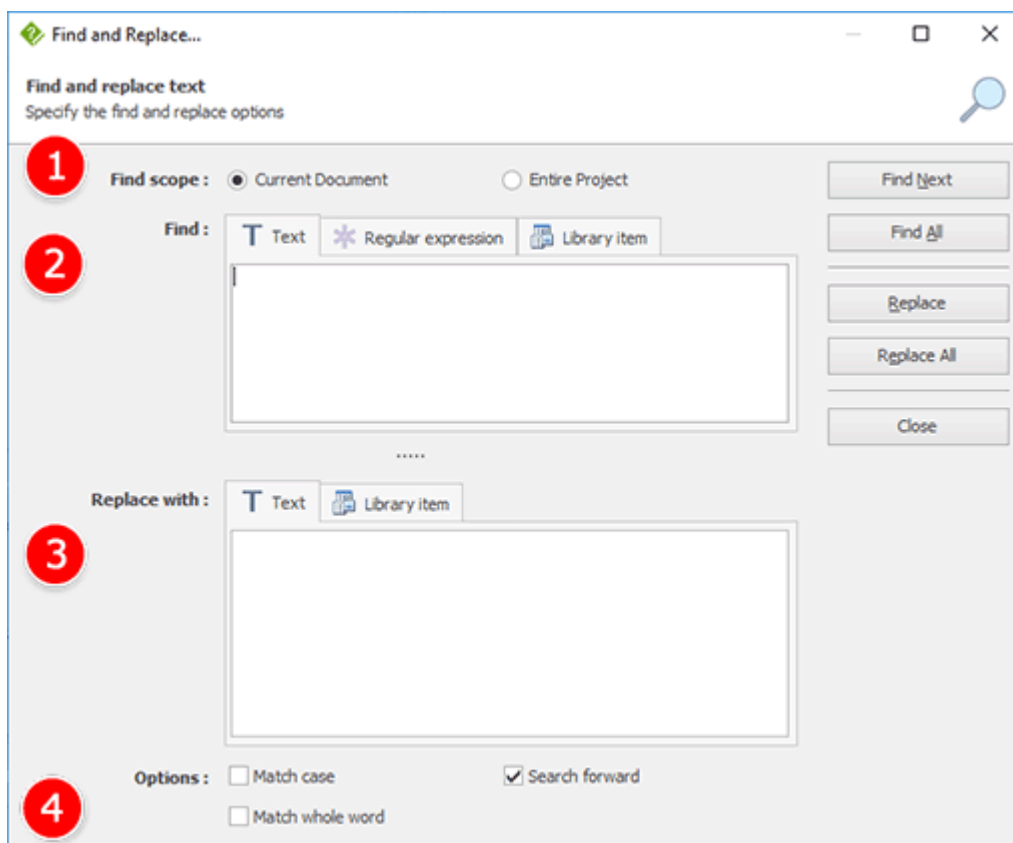
1. The style hierarchy shows a list of all the existing styles and their parents. A style can be selected to be edited. Styles can be added, deleted, imported and exported via the buttons bellow;

2. Common style properties including:

- Name - The name of the style
- Applicable to - Specify if the style is meant for text only, paragraph only or both;
- Based on - Parent style this style refers to. Every parent properties are inherited except the ones edited for that specific style;
- Style for following paragraph - Which style should be automatically applied when creating a paragraph after that style;
- Quick access - If checked, the style will appear in the list in the "Styles" group of the "Write" ribbon tab;

3. Use the "Edit" links to access to customization dialogs for font, hyperlink, paragraph, border and background. Use the "Reset" button to reset to default.

Find and replace window



Access the find and replace window

The find and replace window can be accessed using either:

- The "Find and replace" button in the "Editing" group of HelpNDoc's "Write" ribbon tab
- The CTRL-SHIFT-F keyboard shortcut

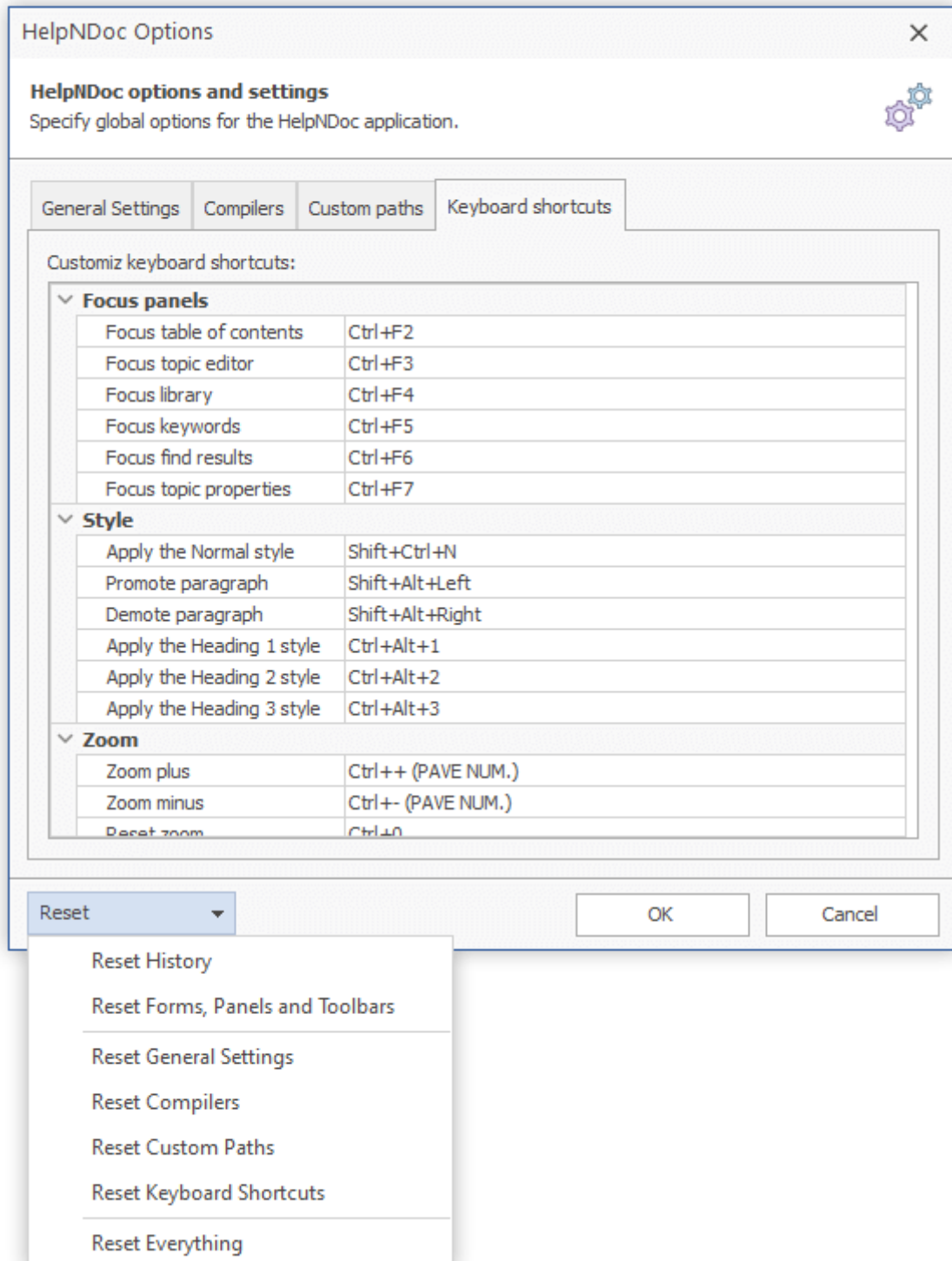
How to use it

Use the find and replace window to look for content (text, regular expression or library items) within the current topic or entire project and optionally replace found occurrences with another text or library item.

The parts of the find and replace dialog are:

1. **Find scope** - Define the scope of the search options. Only in the current document or in the entire project;
2. **Find text** - Content to search: simple text, regular expression, or library item;
3. **Replace with** - Specify the text or library item to use as a replacement for the found content;
4. **Options** - Specify the search options. Match case will find the specified text with the exact same case as it has been written. Match whole word will only search for a complete word. Search forward will specify whether to search forward (from top to bottom) or backward (from bottom to top)

Options window



HelpNDoc's options can be customized by using the "File" menu then "Options" button. This shows the options window with various sections.

General Settings

- Application language: choose HelpNDoc's user interface language. This requires a restart of

HelpNDoc.

- Load and display RSS news: keep up-to-date with [latest news about HelpNDoc](#) as the RSS feed will be displayed in HelpNDoc's welcome page. This may trigger a Windows firewall warning as this requires Internet Access.
- Show ruler: displays or hide a ruler at the top of the topic editor for greater control over paragraph indentation, tabs and table cell sizes.
- Show the navigation bar with breadcrumb editor: displays or hide the navigation bar at the top of the topic editor.
- Automatically hide scrollbars: only show the scroll bars when the mouse is over the control to simplify the UI.
- Default HTTP server port: Specify the default settings for HelpNDoc's [integrated HTTP server](#).

Importers

- Use built-in DocX importer: Built-in DocX importer is faster, uses less memory and better imports DocX documents. If some documents are problematic, un-check it to use the external Microsoft Office Converters instead
- Use built-in CHM decompiler: Built-in CHM decompiler is faster, uses less memory and better imports CHM help files. If some help files are problematic, un-check it to use the Microsoft HTML Help Workshop decompiler instead

Compilers

- To generate some documentation formats, HelpNDoc requires external compilers. This section can be used to setup their path and download them
- Use legacy PDF generator: Starting with HelpNDoc 7.0, the PDF generator has been replaced with a new one which produces better results, faster, yet uses less memory. If some PDF documents are not correctly generated, un-check it to use the old legacy PDF generator instead

Custom paths

Use this section to define the default custom path HelpNDoc will use.

- Default output path: define the default path where project output will be generated when no path is defined. Default is "My Documents\HelpNDoc\Output".
- Backup path: define the path where project backups are stored. Default is "My Documents\HelpNDoc\Backup".
- Dictionaries path: define the path where custom dictionaries are stored. Default is "My Documents\HelpNDoc\Dictionaries".
- Projects path: define the path where projects are opened from or saved to the first time.

Default is "My Documents\HelpNDoc\Projects".

- Scripts path: define the path where custom scripts are stored. Default is "My Documents\HelpNDoc\Scripts".
- Styles path: define the path where the default project styles is placed. See [Customize default project styles](#) to learn more. Default is "My Documents\HelpNDoc\Styles".
- Syntax highlighters path: define the path where customized syntax highlighters are stored. Default is "My Documents\HelpNDoc\Syntax".
- Templates path: define the path where custom templates are located. Default is "My Documents\HelpNDoc\Templates".

Keyboard shortcuts

Some [keyboard shortcuts](#) can be customized. Select a keyboard shortcut in the list and:

- Use the "Edit" button to customize it
- Use the "Reset" button to reset it to its default value

Script Editor

Customize the look and feel of the [script editor](#) used to interact with HelpNDoc API and customize templates:

Setting	Description
Bookmarks / Visible	Show or hide the bookmarks
Bookmarks / Enable	Enable bookmarks keyboard shortcuts
Fonts / Editor	Specify the font name and size of the the editor
Fonts / Gutter	Specify the font name and size of the editor's gutter. Note: Gutter / Use gutter font must be checked too
Gutter / Autosize	Specify if the gutter resizes to fit content
Gutter / Show leading zeros	If checked, leading zeros are added to line numbers

Gutter / Start at zero	Specify if line numbering starts at zero or one
Gutter / Show line numbers	Specify if line numbers are shown in the gutter
Gutter / Use gutter font	If checked, a specific gutter font can be used. If not, it will use the editor's font
Gutter / Visible	Show or hide the gutter
Right Margin / Position	Position of the right margin in number of characters
Spacing / Extra line spacing	Add extra pixels to each line for better clarity
Spacing / Tab width	Specify how many characters are used to display tabs
Options / Auto close after successful run	Automatically close the script editor run from the quick script popup when script successfully runs
Options / Auto indent	Automatically indent based on previous line
Options / Brackets highlight	Highlight matching bracket
Options / Enhanced Home key	Hitting the home key moves to the start of the line or start of code
Options / Enhanced End key	Hitting the end key moves to the end of the line or end of code
Options / Right mouse moves cursor	Editing cursor is moved when right mouse is clicked
Options / Show special chars	Specify if special (invisible) characters are shown or not

Options / Smart tabs	Tabbing checks previous line for faster alignment
Options / Smart tab delete	Delete multiple tabs at once to match previous line
Options / Tab indent	When checked, hit the TAB keyboard shortcut to indent all selected lines, and the SHIFT+TAB keyboard shortcut to decrease indend
Options / Tabs to spaces	Automatically converts tabs to spaces
Options / Trim trailing spaces	Remove any spaces and tabs at the end of lines
Options / Word wrap	Long lines are displayed without horizontal scroll bars when enabled

Reset

Use the "Reset" button to reset parts or all options. **Note:** some reset actions require a restart of HelpNDoc. Available reset options are:

- Reset History: Clear the history of recently opened projects, folders and HTTP server's served folders;
- Reset Forms, Panels and Toolbars: Clear save forms, panels and toolbars size and position and toolbars customization;
- Reset General Settings: Reset the "General Settings" tab to its default values;
- Reset Importers: Reset the "Importers" tab to its default values;
- Reset Compilers: Reset the "Compilers" tab to its default values;
- Reset Custom Paths: Reset the "Custom Paths" tab to its default values;
- Reset Keyboard Shortcuts: Reset all keyboard shortcuts to their default values;
- Reset Everything: Reset all HelpNDoc settings and options to their default values;

Quick start guides

Quickly getting started with HelpNDoc:

- [Launching HelpNDoc](#)

- [Create a new project](#)
- [Adding topics](#)
- [Settings up topic content](#)
- [Generating documentation](#)

Launching HelpNDoc

To launch HelpNDoc, either:

- Locate the HelpNDoc shortcut on the desktop or Windows start menu
- Double click the shortcut in the desktop or single click it in the Windows start menu
- Double click on a HelpNDoc project to open it in HelpNDoc

Create a new project

- Click the "File" menu
- Click the "New project" menu item
- Enter a project title, language
- Select the [Starter kit](#) that you'd like to use
- Click the "OK" button
- Alternatively, click the "Create empty project" button to create a new blank project

See the [Create a new HelpNDoc project](#) step-by-step guide.

Adding topics

- Click the "Home" ribbon tab item if it is not already selected
- In the "Topic" section, click the "Add a topic" button
- The new topic's title is made editable, enter a custom title if needed and press enter

See the [How to create a new topic in HelpNDoc](#) step-by-step guide.

Setting up topic content

- Click the topic to be edited in the "table of contents"
- The topic's content is displayed in the topic editor
- Type in the updated content in the topic editor

Generating documentation

- Click the "Home" ribbon tab item if it is not already selected

- Click the "Generate help" button in the "Project" section
- Choose which documentation format you want to generate
- Click the "Generate" button

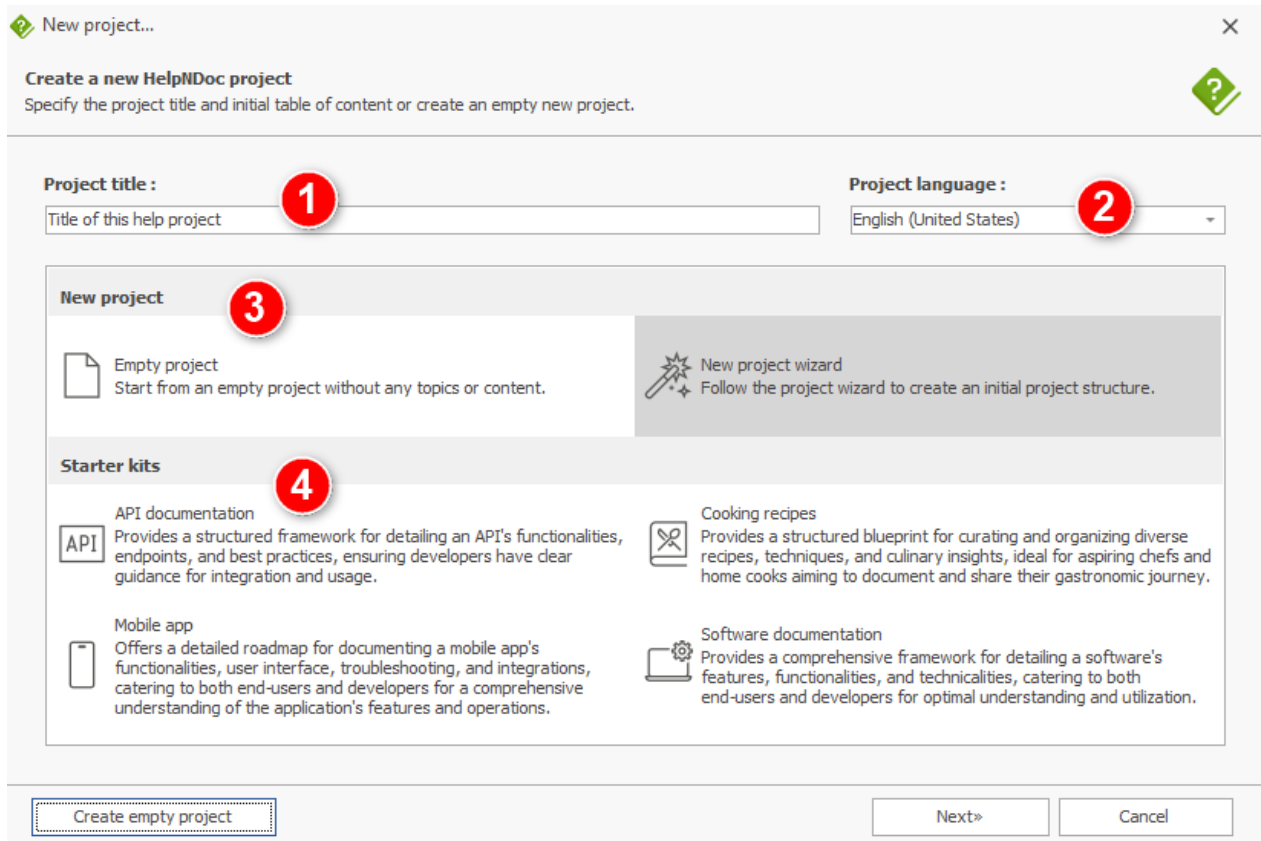
See the [How to publish your documentation](#) step-by-step guide.

Writing documentation

Learn how to use HelpNDoc to write documentation:

- [Create a new project](#)
- [Open an existing project](#)
- [Import other formats](#)
- [Project options](#)
- [Managing the table of contents](#)
- [Using the topic editor](#)
- [Using the library](#)
- [Using the keywords editor](#)
- [Using the spell checker](#)

Create a new project



To create a new project, use the "File" menu and click the "New project" button. This will open the new project wizard dialog. The various parts of this dialog are:

1. Project title: specify the title of the new project
2. Project language: specify the main language of the project
3. Start a new empty project (without any entries in the table of contents) or run the [new project wizard](#)
4. Start a new project using one of the available starter kits

See the [Create a new HelpNDoc project](#) step-by-step guide.

New project wizard

New project...

Create a new HelpNDoc project
Specify the project title and initial table of content or create an empty new project.

Project title :
Title of this help project

Project language :
English (United States)

Table of contents :

Import from existing project...

Introduction
Welcome
What's new
Getting Started
System requirements
Getting help

Create empty project << Back Create project Cancel

The new project wizard can be accessed from the [Create a new project](#) dialog and can be used to quickly define an initial table of contents:

1. Project title: specify the title of the new project
2. Project language: specify the main language of the project
3. Toolbar: Include actions to increase or decrease the current line's indentation and to import the table of contents from another project
4. Table of contents editor

Open an existing project

To open an existing project in HelpNDoc. Either:

- Use the "File" menu, click the "Open project" button and choose the existing project to open
- Use the "File" menu, click the "Recent projects" tab and choose a project which has been opened recently
- Double click on an HelpNDoc project file in the Windows explorer

Import other formats

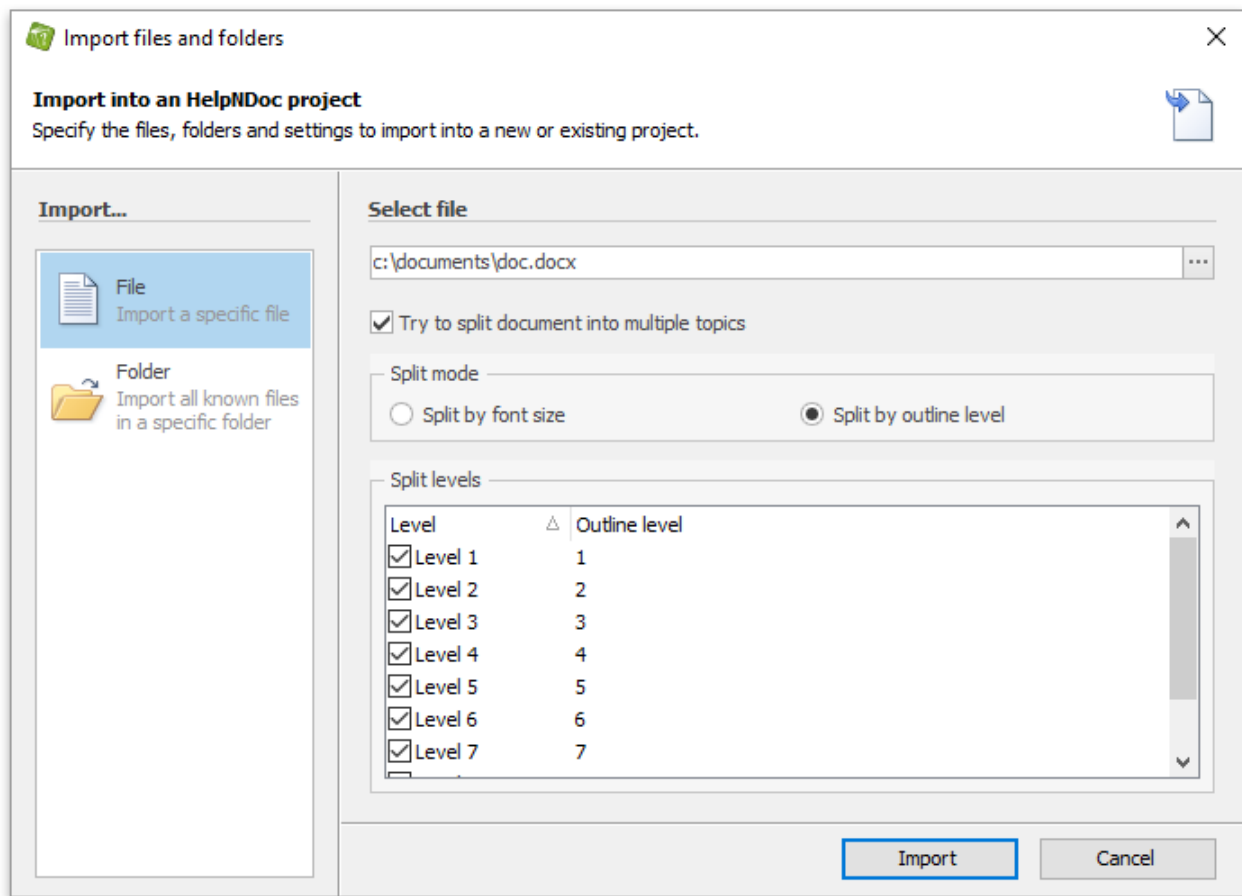
Using the "File" menu, then "Import" action, HelpNDoc can import various existing documentation formats, including:

- Compiled CHM help files as well as source HHP projects;
- WinHelp HLP files;
- ePub eBooks;
- HTML web pages. This format [can be split into multiple topics](#) based on heading level or font size;
- Markdown files. This format [can be split into multiple topics](#) based on heading level;
- Text files;
- Word documents including DocX and RTF file formats. These formats [can be split into multiple topics](#) based on heading level or font size;
- Folders that can contain one or multiple of the previously mentioned formats. Each file will be imported into its own topics. See: [Import folders](#)

Additionally, for some sources (such as HelpNDoc projects, the CHM help format or HHP projects), it is possible to only import the table of contents into a new project:

1. [Create a new project](#)
2. In the new project wizard dialog, click the "import from existing project" link to import the table of contents of the specified file.

Split imported documents

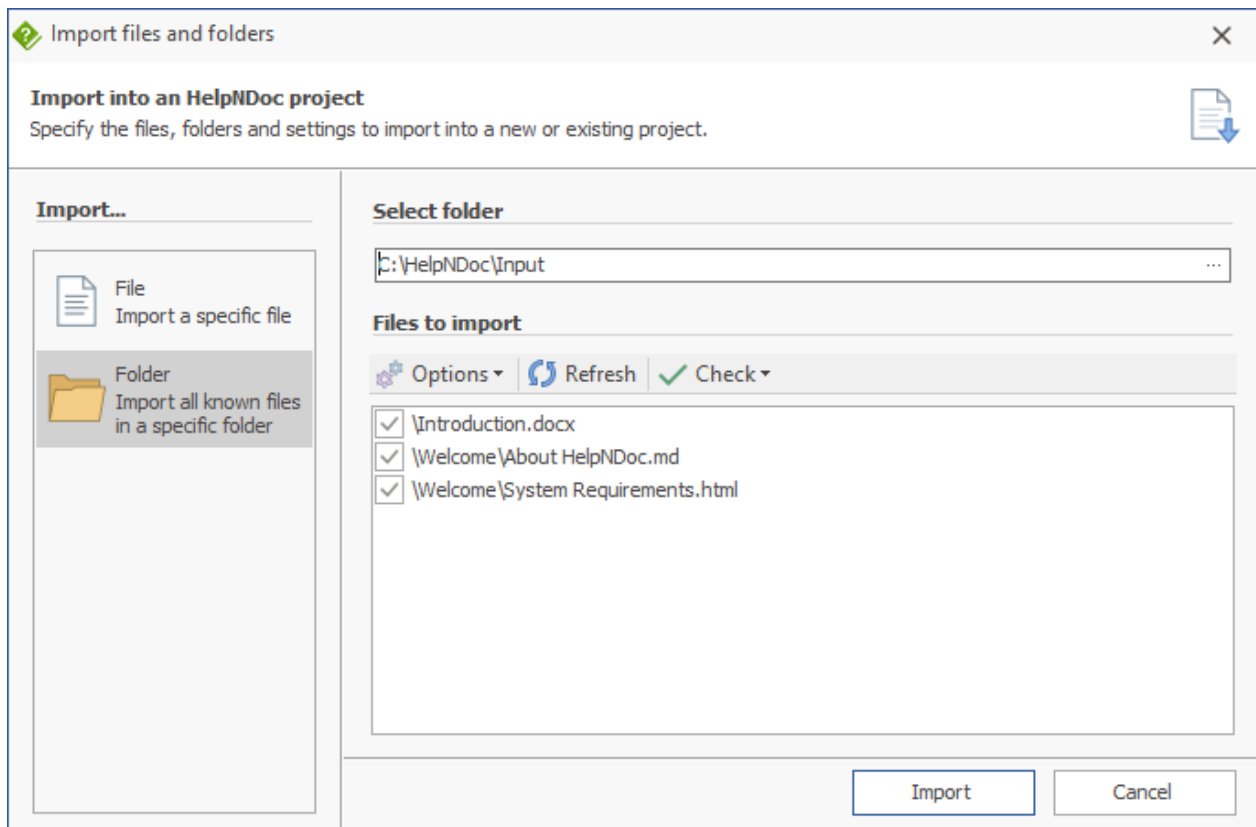


Some documentation formats such as HTML, Markdown and Word documents (DocX and RTF files) can be split into multiple topics. When selecting one of the supported file in the "File" / "Import" dialog, the "Try to split document into multiple topics" checkbox will be enabled. When checked, it is possible to split documents by:

- **Font size** - Use text with the specified font size as topic caption and place subsequent content into the topic;
- **Outline level** - Use text with the specified outline level (heading level) as topic caption and place subsequent content into the topic;

Note: Only checked levels will be imported as topics into HelpNDoc. Other levels will be placed as content within higher level topics.

Import folders



When importing a folder, HelpNDoc will list all [known types of files](#) in the specified folder. Click the "Import" button to import all files with a check mark.

Available actions

Options

- Include sub-folders: Import files which are located in sub-folders of the selected folder
- Recreate hierarchy: Recreate the folder hierarchy in the project's table of contents

Refresh

The "Refresh" button forces checks the selected folder again and updates the list of files to import, based on the selected options.

Check

Quickly check or un-check all files in the list.

Custom hyperlink types

When importing an external document, or when using the scripting API to control the topic editor,

it is possible to define a specific kind of hyperlink to mimic HelpNDoc's hyperlink capabilities:

Scheme	Description
hnd-topic://[TOPIC_ID]	Link to a specific topic using its internal ID. Check the API documentation to learn how to get a topic's internal ID. <i>Ex: hnd-topic://0E16C6CF7A5D42F580EC89876ED08435</i>
hnd-nav:// [RELATIVE_TOPIC]	Link to a relative topic. Possible [RELATIVE_TOPIC] values are: default, first, last, parent, previous, previous-sibling, next, next-sibling <i>Ex: hnd-nav://parent</i>
hnd-counter:// [COUNTER_ID]	Link to a specific counter using its ID. <i>Ex: hnd-counter://figure1</i>

Project options

Each project is saved with its own set of configuration options, which include all the project settings such as copyright, author, and language information.

Access project options

The project options panel can be accessed either by:

- Selecting the "Home" ribbon tab and clicking the "Project options" button in the "Project" group
- Or selecting the project topic in the table of contents, which is the root of all topics and the very first one in the list

Project settings

Various information about the project such as the project title, author information... Some of these options may be exported to the final documentation and overridden for each build. See also [Publishing documentation](#)

Language settings

Specify the project's language and character set. See also [Date and time format settings](#)

Automated settings

Various settings regarding to automation in HelpNDoc.

- **Default properties for new topics:** specify the properties to set to a newly created topic, such as icon, kind, header, footer, status, included in build or visibility
- **Always synchronize Help ID with topic caption:** when this option is checked, a topic's Help ID will be automatically updated when its caption is changed. E.g. changing the topic's caption to "Hello World" will update its Help ID to "HelloWorld". See also [Change topic properties](#)
 - **Synchronize now...** will replace every Help ID based on the current topic's caption
- **Compress un-compressed included library pictures:** when the library contains uncompressed pictures (E.g. Bitmaps BMP files), they will automatically to a non-destructive PNG format in the HND project file to save disk and memory space

Date and time format settings

Some system variables available in an HelpNDoc project can output current date and time information. The following variables are available and will be replaced by the current date and time value at generation time:

Variable name	Meaning
Current Date/Time	Display the current date and time
Date	Display the current date, short format
Date - long	Display the current date, long format
Day	Display the current day, short format
Day - long	Display the current day, long format
Month	Display the current month, short format
Month - long	Display the current day, long format
Time	Display the current time, short format
Time - long	Display the current time, long format

Year	Display the current year, short format
Year - long	Display the current year, long format

Customizing the formats

By default, the date and time variables are displayed based on the project's language settings. It is possible to customize the format for each of those system variables using the "Date / Time format options" dialog.

To show the date and time customization dialog: from the "Home" ribbon tab, click "Project options" then "Customize" next to "Date / Time format" in the "Language settings". The following table explains the various specifiers which can be used when formatting date and time settings:

Specifier	Displays
d	Displays the day as a number without a leading zero (1-31)
dd	Displays the day as a number with a leading zero (01-31)
ddd	Displays the day as an abbreviation (Sun-Sat) using the translated strings according to project language
dddd	Displays the day as a full name (Sunday-Saturday) using the translated strings according to project language
dddddd	Displays the date using the format given by Windows' short date format
ddddddd	Displays the date using the format given by Windows' long date format
e	Displays the year in the current period/era as a number without a leading zero (Japanese, Korean, and Taiwanese locales only)
ee	Displays the year in the current period/era as a number with a leading zero (Japanese, Korean, and Taiwanese locales only)

g	Displays the period/era as an abbreviation (Japanese and Taiwanese locales only)
gg	Displays the period/era as a full name (Japanese and Taiwanese locales only)
m	Displays the month as a number without a leading zero (1-12). If the m specifier immediately follows an h or hh specifier, the minute rather than the month is displayed
mm	Displays the month as a number with a leading zero (01-12). If the mm specifier immediately follows an h or hh specifier, the minute rather than the month is displayed
mmm	Displays the month as an abbreviation (Jan-Dec) using the translated strings according to project language
mmm	Displays the month as a full name (January-December) using the translated strings according to project language
yy	Displays the year as a two-digit number (00-99)
yyyy	Displays the year as a four-digit number (0000-9999)
h	Displays the hour without a leading zero (0-23)
hh	Displays the hour with a leading zero (00-23)
n	Displays the minute without a leading zero (0-59)
nn	Displays the minute with a leading zero (00-59)
s	Displays the second without a leading zero (0-59)
ss	Displays the second with a leading zero (00-59)
z	Displays the millisecond without a leading zero (0-999)

zzz	Displays the millisecond with a leading zero (000-999)
t	Displays the time using the format given by Windows' short time format
tt	Displays the time using the format given by Windows' long time format
am/pm	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hour before noon, and 'pm' for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly
a/p	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly
ampm	Uses the 12-hour clock for the preceding h or hh specifier, and displays the contents of the AM Symbol Windows setting for any hour before noon, and the contents of the PM Symbol Windows setting for any hour after noon
/	Displays the date separator character given by Windows settings
:	Displays the time separator character given by Windows settings
'xx'/"xx"	Characters enclosed in single or double quotation marks are displayed as such, and do not affect formatting

It is possible to reset to the default format for the project's current language by using the "Reset" button. **Warning:** this will discard any previously entered custom date / time format.

Managing the table of contents

The table of contents editor is used to create, manage and organize a hierarchical structure of the documentation projects. Learn more about managing the table of contents:

- [Create topics](#)

- [Delete topics](#)
- [Rename topics](#)
- [Move topics](#)
- [Change topic properties](#)

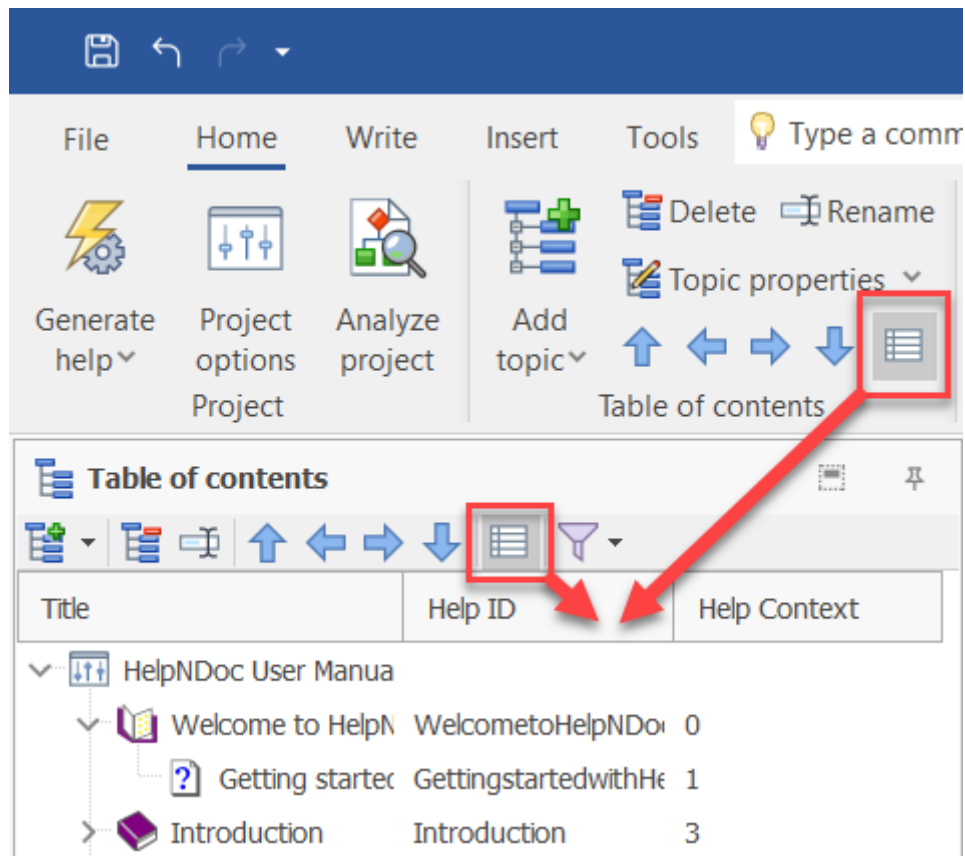
Display the Table of Contents toolbar

To simplify table of contents management, instead of navigating to the global ribbon tab to manage the table of contents, an optional toolbar can be shown at the top of the table of contents panel. To toggle it, click the "Show / Hide table of contents toolbar" in the title bar of the table of contents panel.



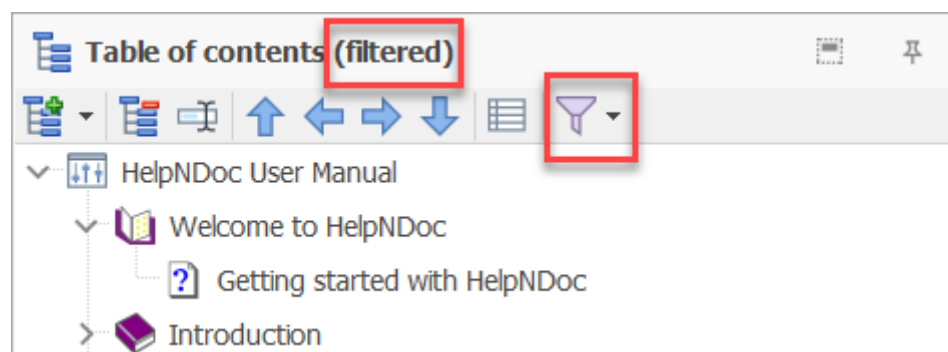
Toggle from compact to extended view

By default, the table of contents only shows a subset of topic properties, such as the status, icon and caption. It is possible to display the topics' Help ID and Help Context numbers in the table of contents by clicking the "Switch to compact / extended view button", which can be found in the "Table of contents" group of the "Home" ribbon tab, or in the table of content's toolbar.



Filtering topics

The table of contents toolbar includes a Filter popup menu to filter topics based on various properties such as their visibility or status. The table of contents panel's title is changed to indicate when a filter is currently applied.



Create topics

Creating a new topic in an opened project can be achieved via two ways:

- Select the "Home" ribbon tab then click the upper part of the "Add topic" button
- Right click on any existing topic in the table of contents (including the project topic) and click the left part of the "Add topic" menu item

By default, a new topic is added at the bottom of the table of contents and becomes the last topic overall. HelpNDoc can optionally create a new topic at the following positions:

- Before the currently selected topic
- After the currently selected topic
- As a child of the currently selected topic
- As the last topic overall (Default behavior)

These actions are available from a sub-menu which can be accessed via the following ways:

- Select the "Home" ribbon tab then click the arrow on the bottom part of the "Add topic" button
- Right click on any existing topic in the table of contents (including the project topic) and hover the arrow on the right part of the "Add topic" menu item

When a new topic is created, its title will become selected and editable for easier modification: it becomes faster to create multiple topics and rename them.

See the [How to create a new topic in HelpNDoc](#) step-by-step guide.

Delete topics

Deleting an existing topic can be achieved via two ways:

- Select the "Home" ribbon tab then click the "Delete topic" button after a topic has been selected in the table of contents
- Right click on any existing topic in the table of contents to open the topic management menu then choose "Delete topic"

A word of caution: Deleting a topic containing children will also delete its children. When a topic is deleted, its associated content is also deleted. Library items used by the topics and keywords linked with the topic are not deleted.

See the [How to delete topics in HelpNDoc](#) step-by-step guide.

Rename topics

The topic's title as displayed in the table of contents can be changed using one of the following ways:

- Right click on the topic and select "Rename"
- Select the topic then hit the "F2" keyboard shortcut
- Select the topic then go to the "Home" tab then click the "Rename" button in the "Table of

contents" section

See the [How to rename a topic in HelpNDoc](#) step-by-step guide

Move topics

Topics are managed as a tree-structure from the table of contents. A topic can contain any number of children topics which itself can contain any number of children topics and so on. Also topics are not sorted in any way in the table of contents so they can freely be positioned. To move a topic in the table of contents:

- Select the topic then from the "Home" ribbon tab, choose one of the move topic action: move up, move down, move left, move right
- Right click on the topic to move then from "Move topic" menu item, choose one of the action
- Drag and drop the topic to the desired position by clicking and holding it, moving the mouse, then release the mouse button where needed

Note: The project topic can't be move. It is always the root of all the topics available in the project.

See the [How to move topics in HelpNDoc](#) step-by-step guide.

Change topic properties

Topic icon

The topic icon is displayed before the topic title in the table of contents. By default, topics containing children will be given a book icon, whereas topic without child will be given a note icon. To change the icon for each individual topic:

- Select the "Home" ribbon tab, then click the "Topic properties" item and choose the new topic icon
- Right click on any existing topic in the table of contents to open the topic management menu then choose the new topic icon

See the [How to select a new icon for a topic in HelpNDoc](#) step-by-step guide.

Topic kind

Each individual topic in HelpNDoc can be either:

- A normal topic - This is a standard topic where new content can be entered in the topic editor
- An empty topic - No content will be entered in that topic and will make it a chapter topic

- An URL topic - This topic will show an external URL instead of the content
- An external included file - The file specified will be included at compilation time in the content of the topic

To change the topic's kind:

- Select the "Home" ribbon tab, then click the "Topic properties" then "Topic kind" item and choose the topic kind
- Right click on any existing topic in the table of contents to open the topic management menu then go to the "Topic kind" item to choose the topic kind
- At the top of the topic editor, click the "Change..." link to choose the topic kind

See the [How to assign topic kind in HelpNDoc step-by-step guide](#).

Description

This field can be used to describe or summarize the current topic. It is used by the default HTML template to generate the HTML description tag for better Search Engine Optimization (SEO).

Help ID

This is one of the most important part of a topic when using the generated documentation. The help ID is a unique alpha-numeric identifier used to locate the topic. This ID will be used to name individual HTML files in the generated HTML documentation, and can be used to open that specific topics from any programming language in the CHM documentation.

To change a topic's Help ID:

- Select the "Home" ribbon tab, then click the "Topic properties" item and enter the new Help ID
- Right click on any existing topic in the table of contents to open the topic management menu then enter the new Help ID

Note: If the "Always synchronize Help ID with topic caption" [project option](#) is checked, HelpNDoc will automatically replace a custom Help ID when the topic's caption is changed.

Note: The Help ID can only contain alpha-numeric and "-" characters. HelpNDoc will ensure this rule by automatically removing any unwanted characters (such as spaces) from the input.

See the [How to manage your topic identifiers in HelpNDoc step-by-step guide](#).

Help Context

The help context is a numeric value which is unique to each topic. It can be used to uniquely identify a topic, or open a specific CHM topic using Windows APIs.

To change a topic's Help context:

- Select the "Home" ribbon tab, then click the "Topic properties" item and enter the new Help context
- Right click on any existing topic in the table of contents to open the topic management menu then enter the new Help context

See the [How to manage your topic identifiers in HelpNDoc](#) step-by-step guide.

Topic Header

The topic header is a simple text which is usually displayed as the title of the topic. By default, HelpNDoc will use the topic title as the header, but it can be configured to either:

- Hide the header - No header will be displayed for that topic
- Display a custom text - A custom text can be specified as the header of that topic.

To change a topic's header:

- Select the "Home" ribbon tab, then click the "Topic properties" then "Topic header" item and choose the topic header
- Right click on any existing topic in the table of contents to open the topic management menu then go to the "Topic header" item to choose the topic header
- At the top of the topic editor, click the topic header link to choose the topic header

See the [How to define a header for a topic in HelpNDoc](#) step-by-step guide.

Topic Footer

The topic footer is a simple text which is usually displayed at the bottom of the topic. By default, HelpNDoc will use the project copyright as the footer, but it can be configured to either:

- Hide the footer - No footer will be displayed for that topic
- Display a custom text - A custom text can be specified as the footer of that topic.

To change a topic's footer:

- Select the "Home" ribbon tab, then click the "Topic properties" then "Topic footer" item and

choose the topic footer

- Right click on any existing topic in the table of contents to open the topic management menu then go to the "Topic footer" item to choose the topic footer
- At the top of the topic editor, click the topic footer link to choose the topic footer

See the [How to define a footer for a topic in HelpNDoc](#) step-by-step guide.

Custom Properties

Each topic can have a set of custom properties. The "Topic properties" panel can be used to manage them: add, rename, delete them as well as edit their values. Custom properties can have any name and value and can be used by [templates](#) or [scripts](#) for additional custom processing.

Using the topic editor

The topic editor is where each topic's content and properties are defined.

Topic kind

Kinds of topics

A topic can be of different kind. When first created in HelpNDoc, the topic is a normal topic with content. The different topic kinds are:

- Normal topic - This is the default topic kind where text, tables, library items... can be added in the topic's content
- Empty topic - This is a topic without any content of any kind attached to it
- Show external URL - The topic will show the specified URL when shown in supported documentation formats
- Include external file - The specified file will be included as the topic's content when the documentation is generated

Changing a topic's kind

Changing the kind of a specific topic can be done by either:

- Right-click the topic in the table of contents and choose a new topic kind
- Change the current topic's kind by clicking the "Change" link at the topic of the topic editor

Headers and footers

Each topic can have a specific header and footer. By default, the topic header is set to display the

topic's title as defined in the table of contents and the topic footer is set to display the project copyright. This can be changed to:

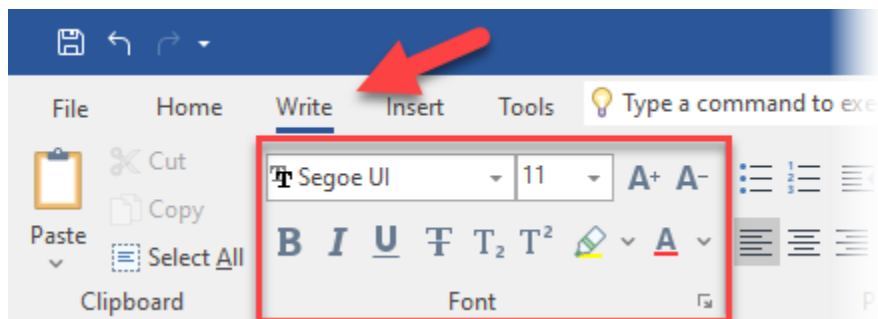
- Hide the header / footer - Do not display anything for that topic
- Display custom header / footer - Specify the text to use for that header / footer

To change the header and footer for a specific topic, either:

- Click in the header and footer links at the top of the topic editor and choose the new option
- Right click on the topic and change its header and footer options.

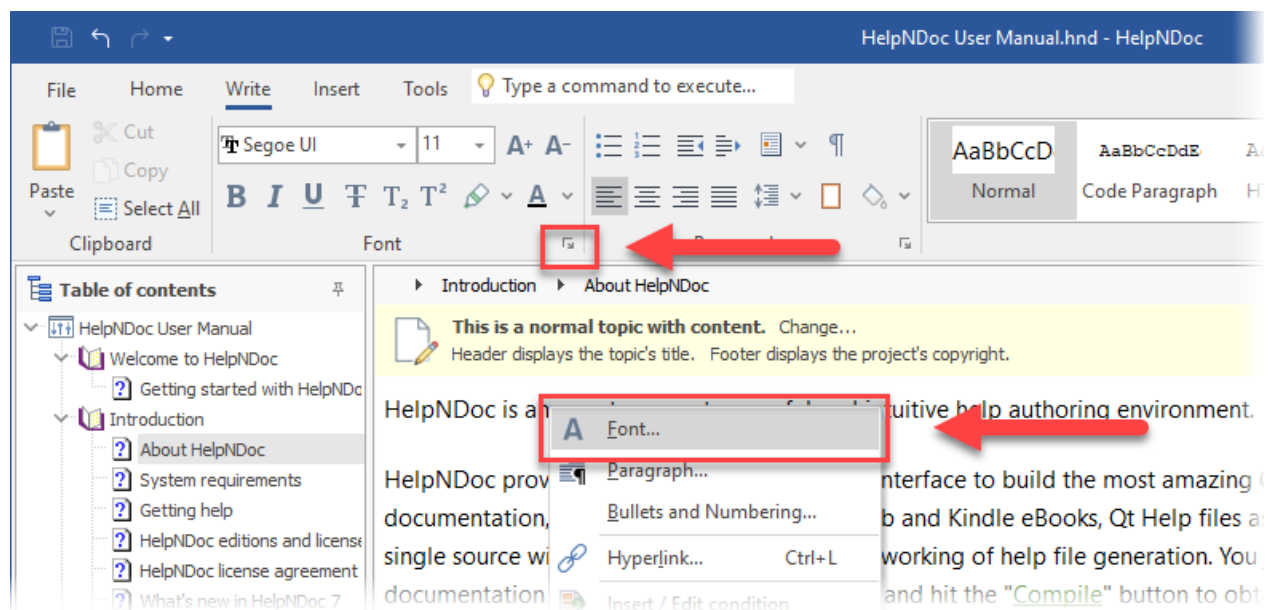
Font properties

Basic font properties



The most often used font properties are easily accessible from HelpNDoc's "Write" [ribbon tab](#). These include: font name and size control, styling options (e.g. bold, italic, underline...), and color options.

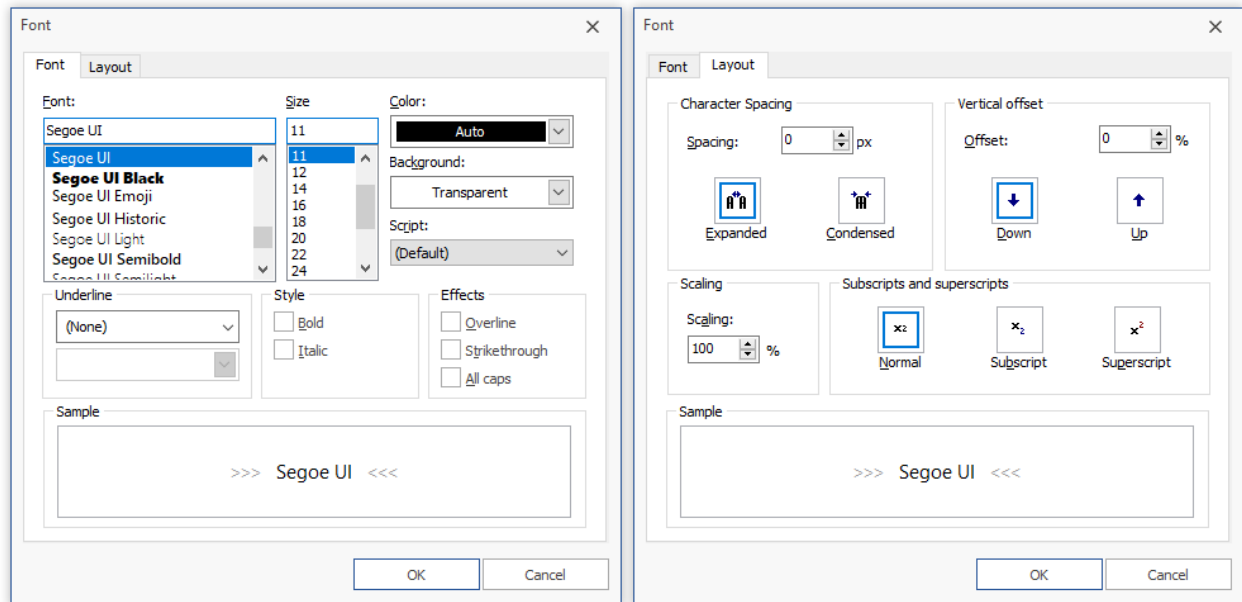
Advanced font properties



For greater control over font properties, the "Font" window can be accessed either:

- From the "Write" ribbon tab, click the arrow at the bottom-right of the "Font" group
- Right-click in the topic editor to show the popup menu, then click "Font..."

Font window

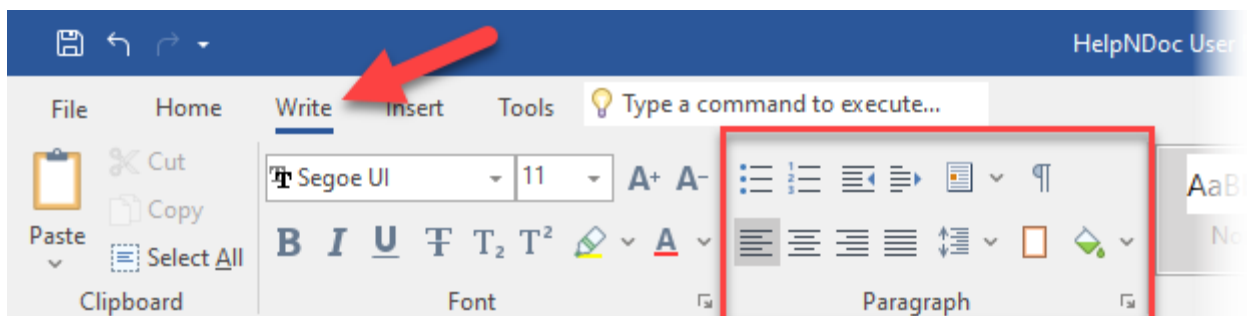


The Font window includes:

- A "Font" tab where common font settings can be defined, such as font name, size, color, background color, style and effects;
- A "Layout" tab where advanced spacing, offset and scaling settings can be defined
- A "Sample" preview of the currently defined font settings

Paragraph properties

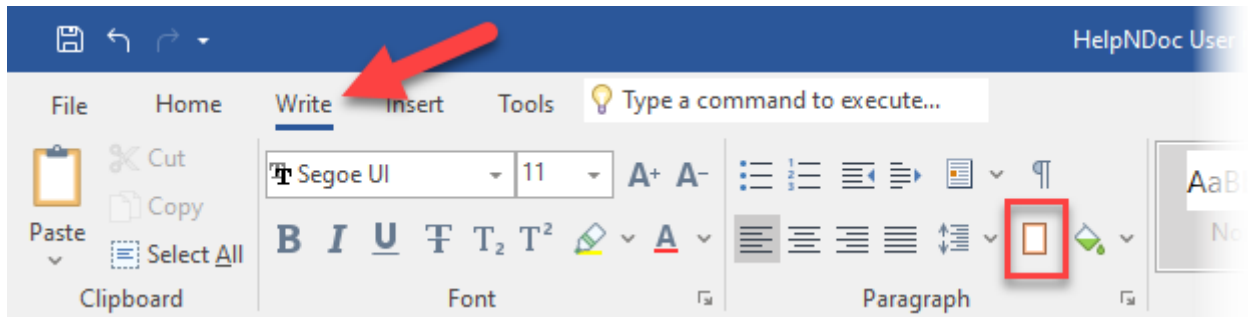
Basic paragraph properties



The most used paragraph properties are easily accessible from HelpNDoc's "Write" [ribbon tab](#).

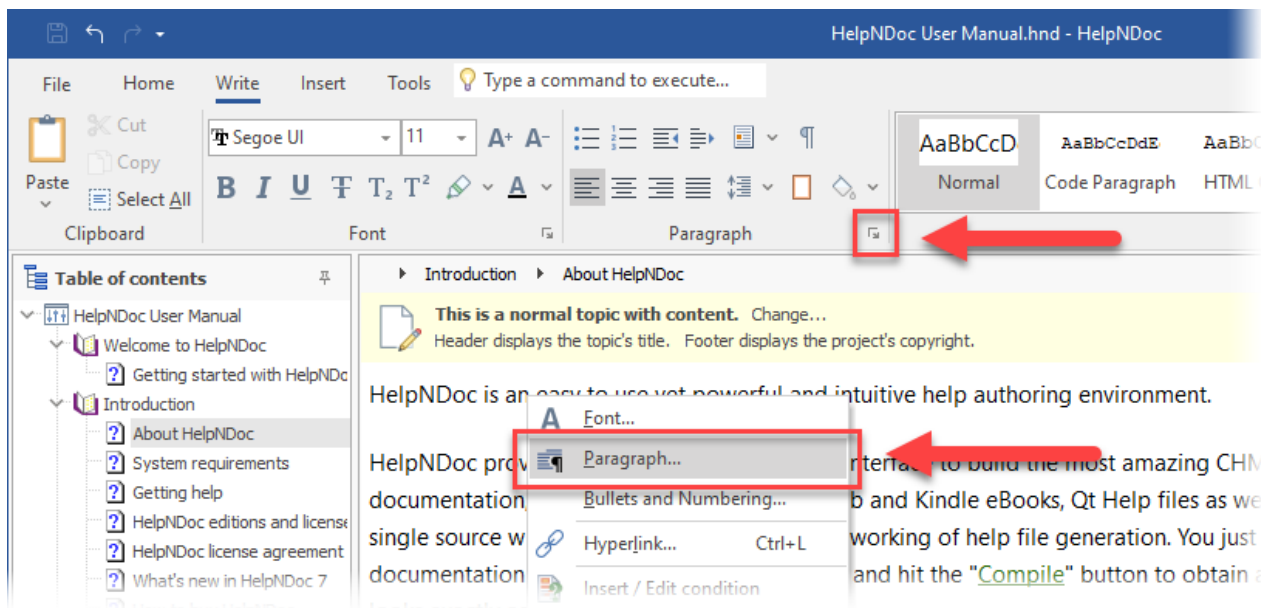
These include: paragraph alignment, spacing, indentation, color options.

Paragraph borders and background



From HelpNDoc's "Write" ribbon tab, in the "Paragraph" group, click the "Paragraph Borders and Background" button to access borders and background settings for the currently selected paragraph.

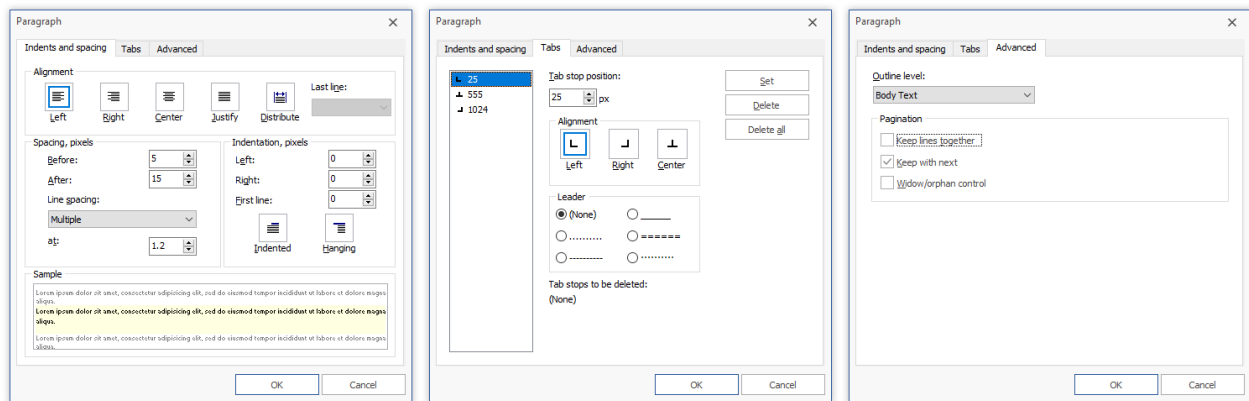
Advanced paragraph properties



For greater control over paragraph properties, the "Paragraph" window can be accessed either:

- From the "Write" ribbon tab, click the arrow at the bottom-right of the "Paragraph" group
- Right-click in the topic editor to show the popup menu, then click "Paragraph..."

Paragraph window



Indents and spacing

Specify paragraph alignment, spacing and indentation properties.

Tabs

Define tab stop position, alignments and leader characters for that paragraph.

Advanced

Define outline level (heading level) and pagination settings:

- Outline level: Define the heading level of that paragraph (e.g. heading 1, heading 2...). This is used to define the topic's [inline table of contents](#).
- Keep lines together: Paragraph is exported on one page, if possible
- Keep with next: Paragraph is exported on the same page as the next paragraph, if possible
- Widow/orphan control: Prevents page breaks after the first line and before the last line of the paragraph

Working with styles

Styles are an important part of HelpNDoc as they provide a way to keep an uniform look throughout the documentation's topics. A style is applied to a piece of text which then becomes linked to it: when the style changes, the format of the text changes too.

HelpNDoc comes with a set of predefined styles. Styles can be added and managed using the [styles editor](#).

To apply a style to a piece of text:

- Select the text to apply the style to

- From the "Write" ribbon tab, choose the style to apply from the "Styles" group and click it

Note: By default, any custom font attribute applied to the selected content won't be overwritten when a style is applied. To reset custom first attributes before applying a new style:

- Selecting the content to style in HelpNDoc's topic editor
- Press the CTRL keyboard shortcut and keep it pressed
- Apply the desired style

See the following step-by-step guides:

[How to customize the default styles for new projects](#)

[How to create a style to display a note or a warning message](#)

Working with hyperlinks

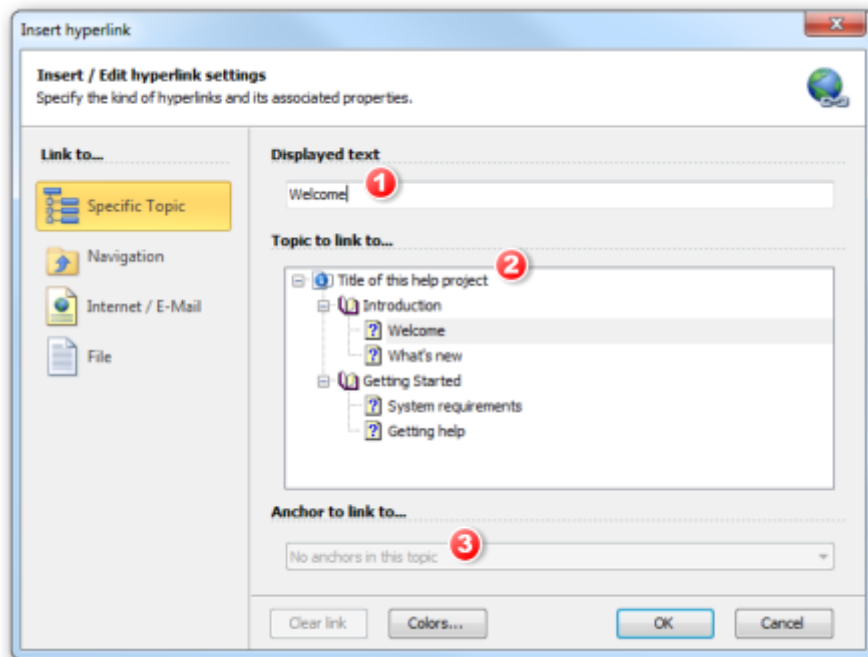
The text contained in a normal topic with content can contain hyperlinks. Those links will redirect the reader to the specific element they link to. To create an hyperlink:

- Select the text to transform to an hyperlink
- Select the "Insert" tab and click the "Insert / Edit hyperlink" in the "Links" panel (Keyboard shortcut: CTRL+L)
- Specify the links' attributes

An hyperlink can link to:

- [A specific topic](#): the specified topic will be shown
- [A relative topic](#): the relative topic, based on the currently viewed one, will be shown
- [An Internet or e-mail address](#): the Internet page will be shown or a new e-mail will be created
- [A file](#): the specified file will be shown or downloaded
- [A counter](#): the specified counter instance will be shown

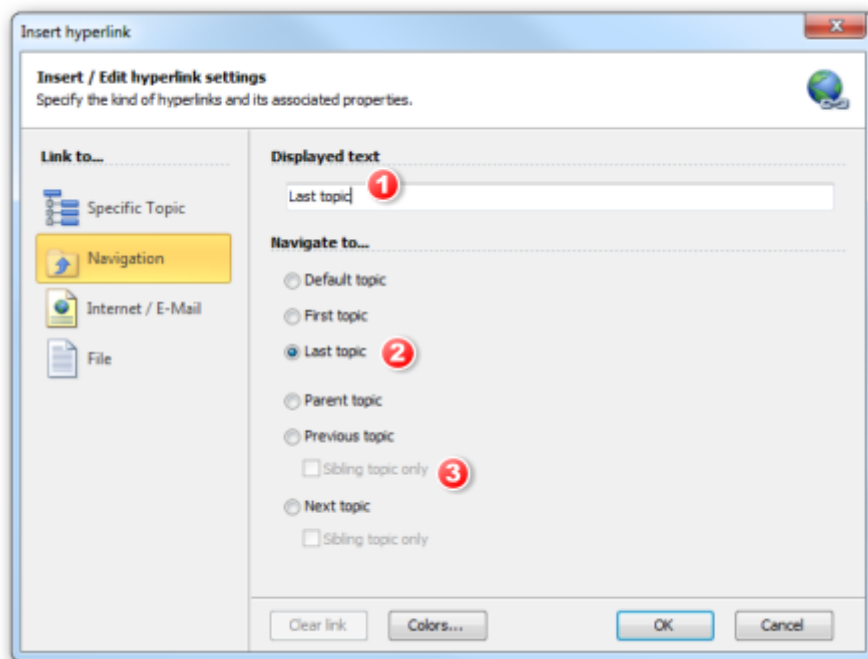
Link to a specific topic



Linking to a specific topic will allow the end-user to navigate to that particular topic by clicking the link. To create a link to a specific topic:

1. Provide the link text. This field is not enabled if you have already selected the text in the topic editor
2. Choose which topic to link to by selecting it in the hierarchy
3. Optionally choose the topic's anchor to link to

Link to a relative topic



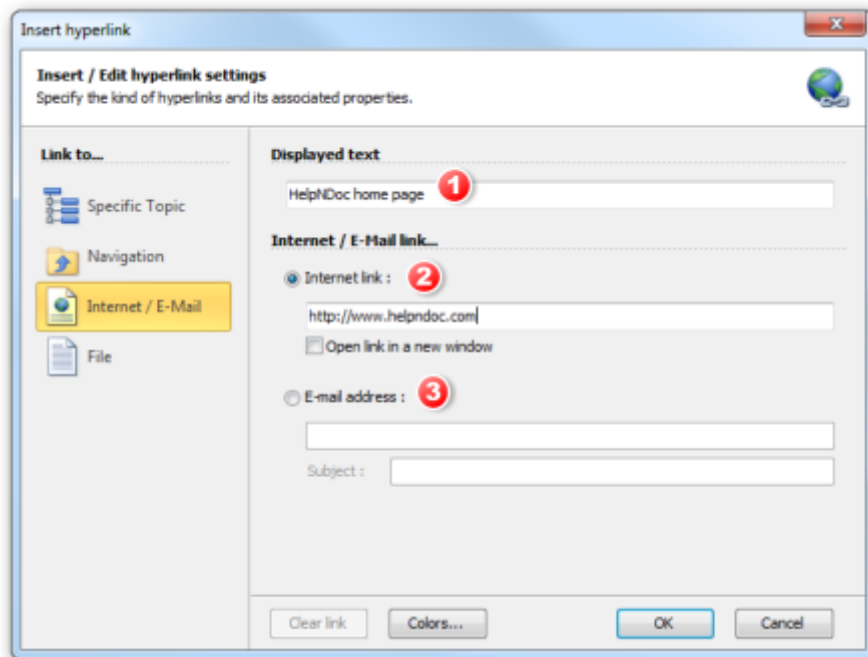
Creating a relative link, or navigation link, will provide a way to link to a topic relative to the current one. HelpNDoc can create navigation links to:

- Default topic - The topic which has been set as the default one in the [project options](#)
- First topic - The very first topic in the table of contents
- Last topic - The very last topic in the table of contents
- Parent topic - The parent topic of the topic containing the link
- Previous topic - The topic just before the one containing the link. If "Sibling topic only" is checked, it will link to the previous topic at the exact same hierarchy level
- Next topic - The topic just after the one containing the link. If "Sibling topic only" is checked, it will link to the previous topic at the exact same hierarchy level

To create a navigation link:

1. Provide the link text. This field is not enabled if you have already selected the text in the topic editor
2. Choose what kind of navigation link to create
3. For previous and next topics, specify whether to link to a sibling topic or not

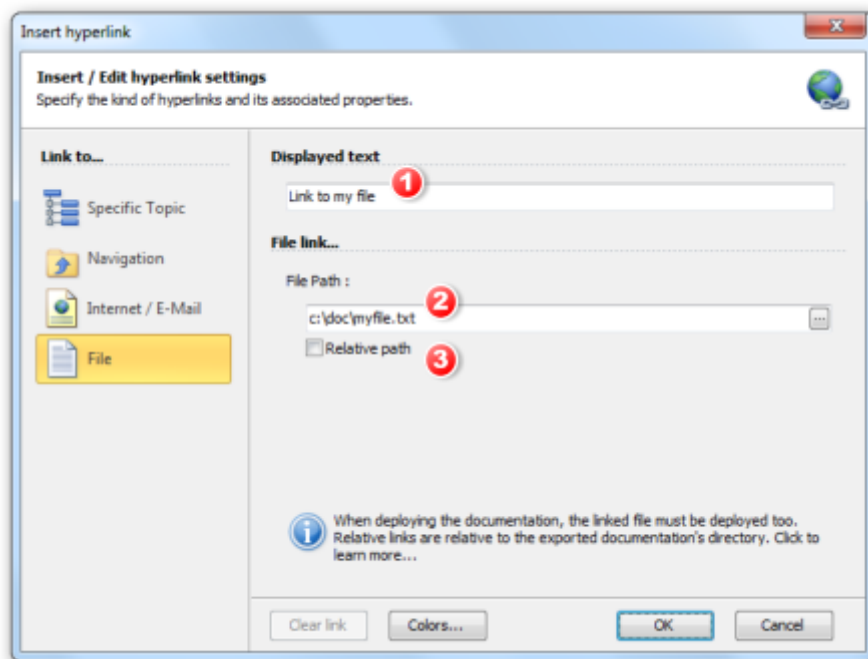
Link to an Internet or e-mail address



Create a link to an Internet or e-mail address by:

1. Provide the link text. This field is not enabled if you have already selected the text in the topic editor
2. For an Internet link, specify the URL and whether this link should open in a new window or not
3. For an e-mail address, specify the e-mail address and optionally a subject for the e-mail to send

Link to a file




Based on the end-user Windows configuration and documentation format, an hyperlink to a file will either:

- Show the file directly in the help viewer
- Show the file in an external application if the file format is registered to that application
- Provide a download file box to let the user download it locally

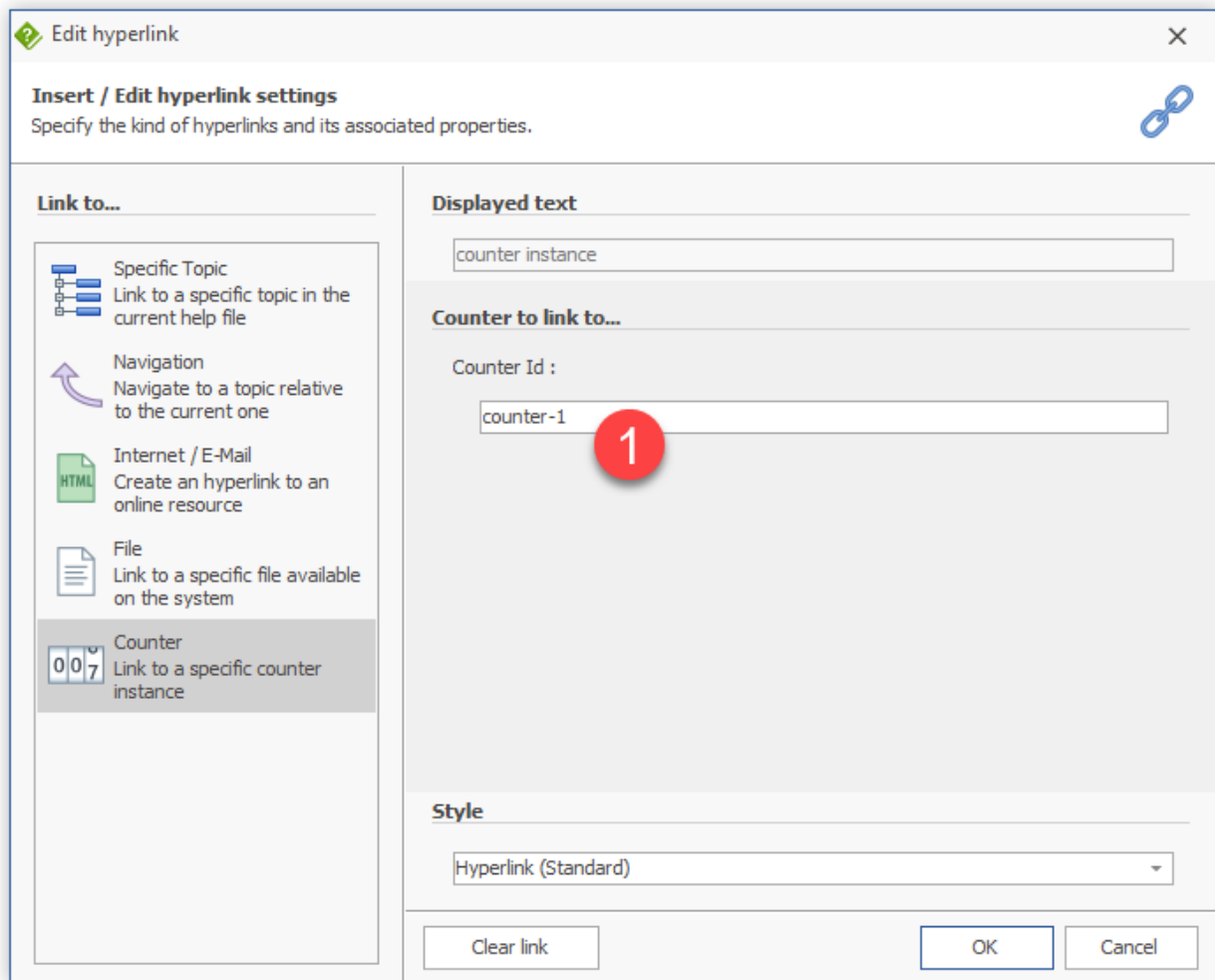
To create an hyperlink to a file:

1. Provide the link text. This field is not enabled if you have already selected the text in the topic editor
2. Indicate the file path and name
3. Indicate whether the provided path is an absolute or relative path

 The help file won't be included with the generated documentation. This means that the help file must be deployed with the final documentation and placed in the correct folder when installed on the end-user computer:

- For a non-relative file: the file must be placed in the exact same folder and have the same name as the one defined in the file path field. Example: c:\doc\myfile.txt
- For a relative file: the file must be placed in a relative folder based on the main documentation file. Example: the relative path is set to "file\myfile.txt" so the file must be placed in the "file" sub-folder of the documentation output folder

Link to a counter



Create a link to a specific counter instance by specifying its unique identifier:

1. Specify the unique identifier of the counter to link to

Working with tables

Tables are used to either display tabular data or create complex layout in the final documentation. To create a new table in HelpNDoc, from the "Insert" ribbon tab, click the "Insert table" button in the "Items" group and either:

- Choose the number of rows by columns to add by clicking the desired table size
- Click the "Insert table" button to specify the size and some properties for the new table

Once a table is present in a topic, clicking it will display the "Table tools, Layout" ribbon tab. This can be used to create, delete, change properties for the cells and the table.

Working with pictures

Pictures are inserted in the library then in the topic editor. This provides a way to use the same picture multiple times and modify it from the library without the need to find it in the topics. To insert a picture, either:

- From the "Home" ribbon tab, click the "Add item" button in the "Library" group and choose "Add picture". Then drag the picture from the library in the topic editor
- From the "Insert" ribbon tab, click the "Insert picture" button then "Insert another picture". This will add it to the library prior to inserting it in the topic editor

When a picture is clicked in the topic editor, the "Picture tools, format" contextual ribbon tab is shown to modify the picture's properties. From there, it is possible to:

- Replace the picture with another one
- Reset the picture properties such as size and alignment
- Align the picture in the text flow
- Adding the picture's alternative text: this is used in HTML based documentation as a placeholder text while the picture is being loaded
- Specify the picture's width and height

See the [How to add an item to the library](#) step-by-step guide.

Working with the image map editor

[Image maps](#) can contain one or multiple interactive shapes. A shape can be a rectangle, circle or polygon and can link to any kind of links handled by HelpNDoc: topics, relative links, URLs, EMails, or file links. See how to use the [image map editor](#).

Note: Image maps are only compatible with CHM and HTML documentation formats as well as some ePub readers.

Manage shapes

To create a shape in the image map editor, choose the kind of shapes in the "Create Shapes" section and draw it over the image.

To select a shape, click on it in the editor, or choose it in the "Shape" list of the "Properties" section.

When one or more shapes are selected, click "Delete" in the "Manage Shapes" section to delete them.

Move and resize shapes

Once a shape is added, it can be moved and resized by dragging its content or its handles in the editor. Rectangle and Circle shapes can also be moved and resized using the editors in the "Properties" section.

Assign title and link to shapes

When a shape is selected, change its title in the "Properties" group. Title are used as hints in web-browser or as indication for accessibility settings.

To update the link of a shape, click the link next the "Link To" in the "Properties" group to use the [hyperlink editor](#).

See the [How to create an image map](#) step-by-step guide.

Using the library

The library is a central storage place for media and third-party elements such as:

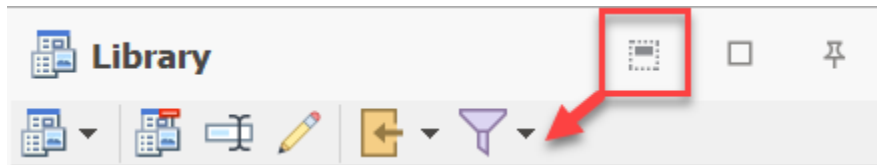
- [Folders](#) - Containers for other library items
- [Barcodes](#) - Barcode and QR code images
- [Counters](#) - Auto-incrementing fields used to count items such as figures, tables, equations...
- [Documents](#) - DOCX, RTF, HTML, TXT...
- [Dynamic content](#) - Scripts producing dynamic HTML-based content
- [Equations](#) - Mathematical equations and expressions
- [HTML Code](#) - This is raw HTML code which will be exported as-is in the final HTML based documentation
- [Image maps](#) - Pictures with interactive click-able zones
- [Movies](#) - MOV, AVI...
- [Pictures](#) - PNG, JPEG...
- [Snippets](#) - Place-holders for rich text content which can contain formatted texts, pictures...
- [Variables](#) - Place-holders for textual content

All those elements are stored within the library and **can be re-used in any number of topics** within the current project. Once an item is placed in the topic editor, it is linked to the corresponding library item and therefore any modification made to the library item will also be propagated to the all linked items. As an example, changing a picture in the library which has been placed in hundreds of topics, will automatically update all those topics to display the updated

picture.

Display the Library toolbar

To simplify library management, instead of navigating to the global ribbon tab to manage library, an optional toolbar can be shown at the top of the library panel. To toggle it, click the "Show / Hide library toolbar" in the title bar of the library panel.



Adding a library item

There are multiple ways to add items in the library:

- From the "Home" ribbon tab, in the "Library" group, use the "Add item" button;
- From the "Home" ribbon tab, in the "Library" group, use the "Import files" button to [import multiple files at once](#);
- Drag and drop files from the Windows Explorer on the library to show the ["Import files" dialog](#) and import them;
- Drag and drop files from the Windows Explorer on the topic editor to show the ["Import files" dialog](#), import them, and place them within that topic.

See the [How to add an item to the library](#) step-by-step guide.

Inserting a library item

To insert an item from the library to the topic editor, either:

- Drag and drop the item from the Library panel into the topic editor
- Select the element in the library panel then click the "Insert in topic" button from the "Home" ribbon bar, in the "Library" group
- Right click the element in the library panel and choose "Insert in topic"

Inserting the library item's content into a topic

Some library items (documents, HTML, variables and snippets) can have their content directly inserted within a topic. This is useful to transform library items as "templates" for topic content. If the selected library items supports it, either:

- Click the arrow at the right of the "Insert in topic" button from the "Home" ribbon bar, in the

"Library" group, then click "Insert content in topic"

- Right click the element in the library panel, hover over the "Insert in topic" item, then click "Insert content in topic"

Removing a library item

When an item is not needed anymore, it can be removed from the library:

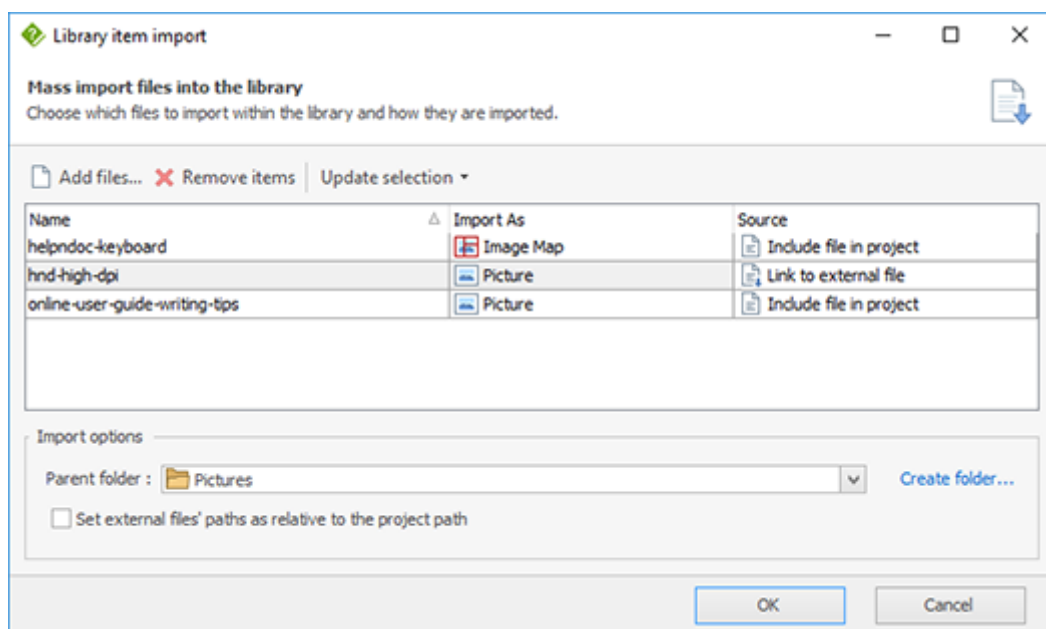
- Select the unwanted item in the "Library" panel
- From the "Home" tab, in the "Library" group, click "Delete"

Note: Deleting a library item from the library will not delete any instance of that item in the project's topics. This providing a way to review each topic individually and decide if that instance needs to be deleted or replaced. The [project analyzer](#) can be used to quickly spot and fix delete library items.

See the [How to delete an item to the library](#) step-by-step guide.

Import files dialog

The import file dialog presents a quick and easy way to import multiple media elements in the project's library. It can import images, image maps, documents and videos.



The import file dialog can be accessed:

- From the "Home" ribbon tab, in the "Library" group, use the "Import files" button;
- By drag and dropping files from the Windows Explorer on the library;

- By drag and dropping files from the Windows Explorer on the topic editor;

Adding files

To import additional files, either:

- Drag and drop files from the Windows Explorer on the file list;
- Use the "Add files" button to select files to import

Managing files

The file list can be used to manage files before they are imported:

- Delete an item using the "Remove item" button;
- Rename the item by editing the "Name" column;
- Change the import type from [Picture](#) to [Image map](#) and vice versa;
- Change the source of the library items for images, documents and movies:
 - Include file in project will save the file within the HND project file;
 - Link to external file will create a link to that file and only store that link. Use the "Set external files' paths as relative to the project path" check box to store a relative path instead of the full path. **Note:** this check box is grayed if the project hasn't been saved yet.

Update selection

"Import As" and "Source" columns can be rapidly changed for multiple items by selecting the desired items in the list, then use the "Update selection" drop down menu to update them.

Folder selection

Select the parent folder to import all the items in the list in that folder. The "Create folder" link can be used to create a new folder in the library if needed.

Folder library item

Folders are containers for other library items or sub-folders. They are useful for organization purposes such as:

- Infrequent use of library item: place all system variables in a specific folder and collapse it to hide them from the list
- Filtering library items: place each kind of library items in its specific folder (e.g. all pictures in the "pictures" folder...)

- Important library items: create a folder where important library items must never be updated / deleted by co-workers
- And so on...

Note: A folder library item can't be placed in a topic, only its children library items (which are not folders) can.

Warning: Deleting a folder will also delete all library items contained in that folder, including sub-folders. Proceed with care.

Placing a library item in a folder

After [adding a folder to the library](#), drag and drop an item over that folder to move it there. Alternatively, it can be dropped on any non-folder item in that folder to perform the same move.

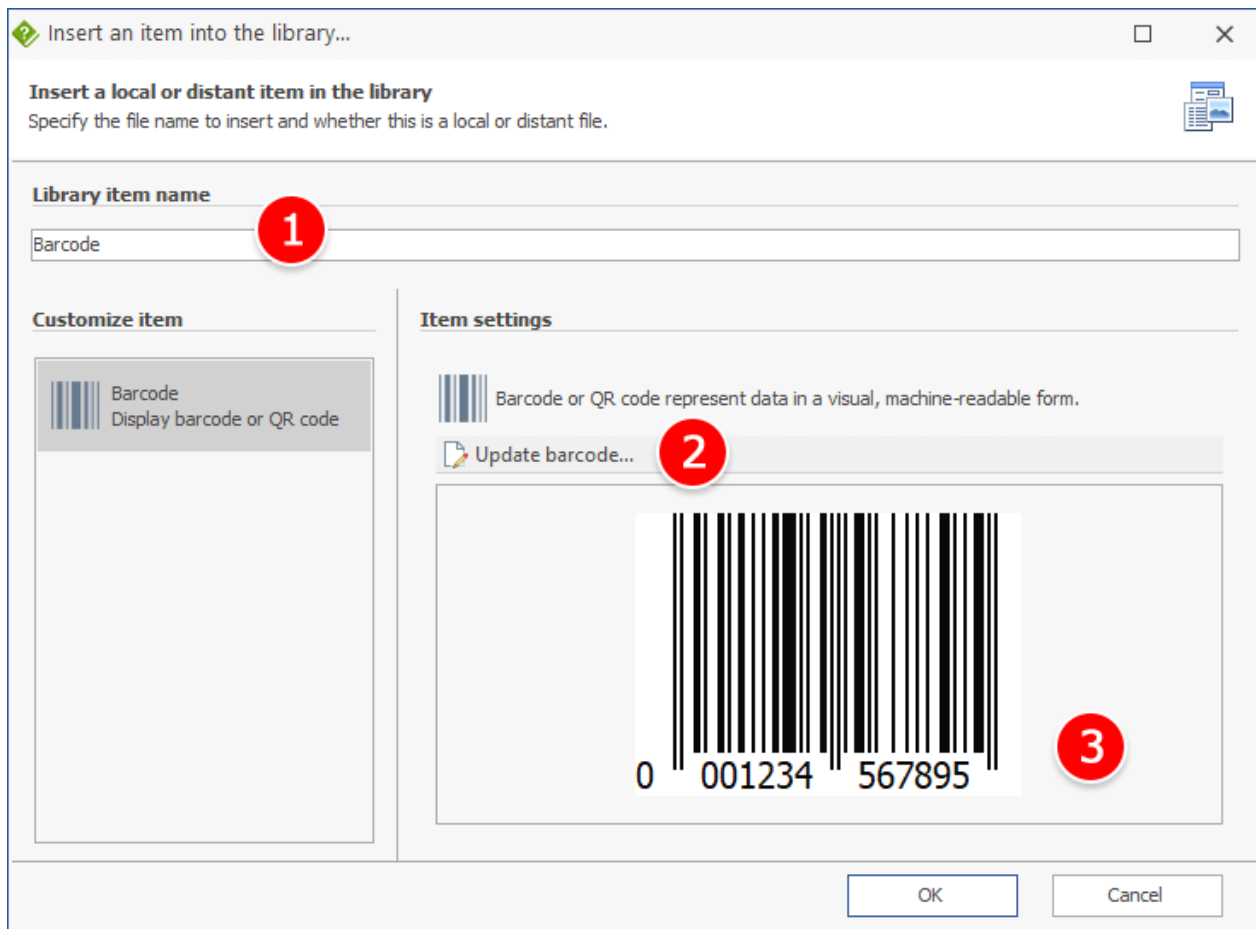
Export content

Use the "Export content" button to export every "included" items contained in that folder.**Note:** only items included in the project will be exported.

Barcode library item

Barcodes and QR codes are extremely useful to produce a machine-readable visualizations: a bar code reader or smartphone camera can be used to quickly and safely interpret them, making it easier to share complex data with end-users. HelpNDoc supports the following barcode formats: Code 11, Code 128, Code 39, Code 39 Extended, Code 93, Code 93 Extended, EAN-8, EAN-13, Interleaved 2 of 5, MSI, QR Code, UPC-A and UPC-E.

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Commands

Commands available for the image map library item:

- Update barcode: show the [Barcode editor](#) to edit the current content of the barcode

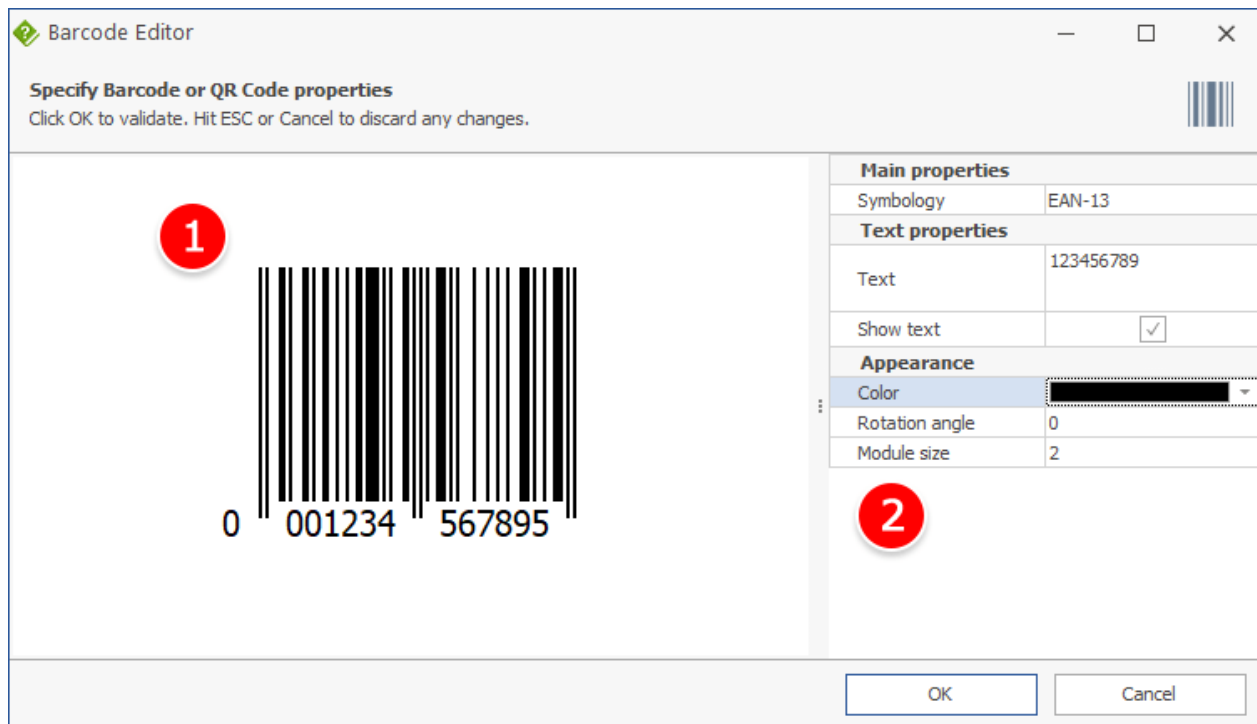
3. Preview

Current content of the barcode

[Barcode editor](#)

The Barcode editor can be used to create and modify barcodes in the documentation project.

Overview of the user interface



1. Preview

Visual representation of the currently edited barcode.

2. Properties

Choose the type of barcode to create and set its properties:

- **Symbology:** Type of barcode (e.g. EAN-13, MSI, QR Code...)
- **Text:** Content to encode in the barcode
- **Show text:** Some barcodes can show the encoded text
- **Color:** Specify the color of the barcode
- **Rotation angle:** Rotate the visual representation of the barcode 90°, -90° or 180°
- **Module size:** Size of the module

Counter library item

Counters are auto-incrementing fields. They can be used to count items such as figures, tables, equations... throughout the project. Counter instances can have an optional caption and identifier. Counters have both global properties (such as caption format) and instance properties (such as caption and unique ID).

Create a counter: overview of the user interface

When creating a new counter to the library, the following dialog is shown.

1. Library item name

Choose a unique name for that library item.

2. Format

The final exported counter's value is based on the format specified:

- Caption format: Specify the format of the exported counter if the counter instance's caption is not empty
- Empty caption format: Specify the format of the exported counter if the counter instance's caption is empty

Some examples of exported counters:

Caption format	Empty caption	Instance's	Exported value

	format	caption	
Figure {A}: {caption}	Figure {n}	<i>[Empty]</i>	Figure 1
Figure {A}: {caption}	Figure {n}	HelpNDoc	Figure A: HelpNDoc
{caption} - Table {R}	Table {r}	<i>[Empty]</i>	Table i
{caption} - Table {R}	Table {r}	HelpNDoc	HelpNDoc - Table I

3. Placeholders

Field	Description
{caption}	Replaced by the instance's caption
{n}	Counter number as an integer value (e.g. 1, 2, 3, 4...)
{r}	Counter number as lowercase roman value (e.g. i, ii, iii, iv...)
[R]	Counter number as uppercase roman value (e.g. I, II, III, IV...)
{a}	Counter number as lowercase alpha value (e.g. a, b, c, d...)
{A}	Counter number as uppercase alpha value (e.g. A, B, C, D...)

Counter instance: overview of the user interface

By double-clicking a counter in HelpNDoc's topic editor, the counter instance's editor is shown.

Edit library item Figures

Edit the specified library item
Specify the properties of the library item

Library item name

Figures **1**

Customize item

Counter
Manage a documentation wide counter

Item settings

Counters are used throughout the documentation to count and list elements such as tables, figures ...

Instance settings **2**

Caption: HelpNDoc

Id: hnd

Global settings **3**

Caption format: Figure {n}: {caption}

Empty caption format: Figure {n}

{caption} Replaced by the instance's caption

{n} Counter number as an integer value

{r} or {R} Counter number as lowercase or uppercase Roman value

{a} or {A} Counter number as lowercase or uppercase alpha value

OK Cancel

1. Library item name

Choose a unique name for that library item. **Warning:** This is shared between all instances as it is applied to the associated library item.

2. Instance settings

Settings which are specific to this counter's instance.

Setting	Description
Caption	Caption of this instance which will be used to replace the {caption} placeholder when exported. If specified, it will be exported using the "Caption format" value. If empty, it will be exported using the "Empty caption format".

Id	Unique identifier of the instance. See: Link to a counter
----	---

3. Global settings

Those settings are shared between all instances as they are applied to the associated library item.
See: [Create a counter - overview of the user interface](#).

Document library item

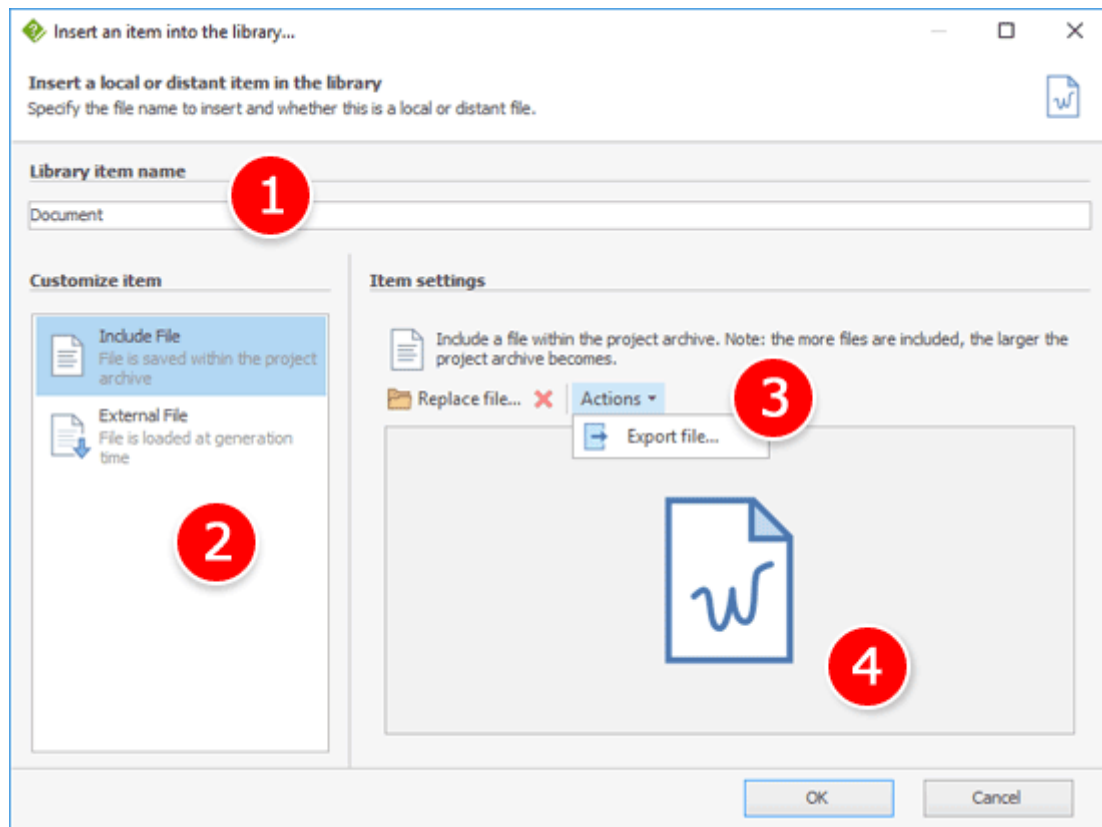
A document library item will be included where it has been placed at generation time. It can be useful in multiple situations:

- A piece of content needs to be repeated multiple times within the project
- The document is managed by someone else without access to the HND project file

Note: HelpNDoc can import the following documentation formats:

- TXT text files;
- RTF rich text documents;
- HTML web pages;
- Markdown files;
- DOC and DOCX Word documents. This might require as specific Microsoft Office Compatibility Pack: see [Doc or DocX files can't be imported](#)

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Customize item

A document can either be stored within the project or linked from an external location based on the project's requirements. Choosing how it is stored can be decided individually for each item based on pros and cons for that specific item and / or overall documentation project.

Storage	Description	Pros	Cons
Include File	Item file is stored within the project archive	<ul style="list-style-type: none"> Item is always available even when the project is moved to a different location Sharing project is easier as the HND 	<ul style="list-style-type: none"> HND project file becomes larger with each included item Replacing an item involves locating it in the

		<p>project file contains the item's content</p> <ul style="list-style-type: none"> The item is always available to the end-user as it is stored with or within the generated documentation file 	<p>library and updating it</p> <ul style="list-style-type: none"> Items can't be shared between multiple projects Generation time is slower as the file needs to be copied / included in the generated documentation
External File	<p>Item file is stored anywhere on the hard drive or a network location, and included at generation time. Note: The external file path can be absolute, or relative to the HND project file location</p>	<ul style="list-style-type: none"> HND project file is smaller as only the path to the external file is stored Updating the item on the hard drive or network location will update it the next time the documentation is built Items can be shared between multiple projects: each project will include it when needed The item is always available to the end-user as it is stored with or within the generated documentation file 	<ul style="list-style-type: none"> Item location must be updated in the library when the HND project file is moved, or the item must be moved with the project Sharing a project requires all external items to be shared too Sharing a project might require an update of all external items' paths Generation time is slower as the file needs to be copied / included in the generated

			documentation
--	--	--	---------------

3. Commands and fields

Include File

- Insert / Replace file: Locate a file on the hard drive or a network location to
- Remove file from project: Remove the content of the file from the project
- Export file: Export the content of the file to the hard drive

External File

- File path: Absolute or relative path of the file to include at generation time

4. Preview

For included files, an icon of the library item is displayed. It is possible to drag and drop a file from a third party application such as the Windows Explorer into the preview to include it or replace the existing file.

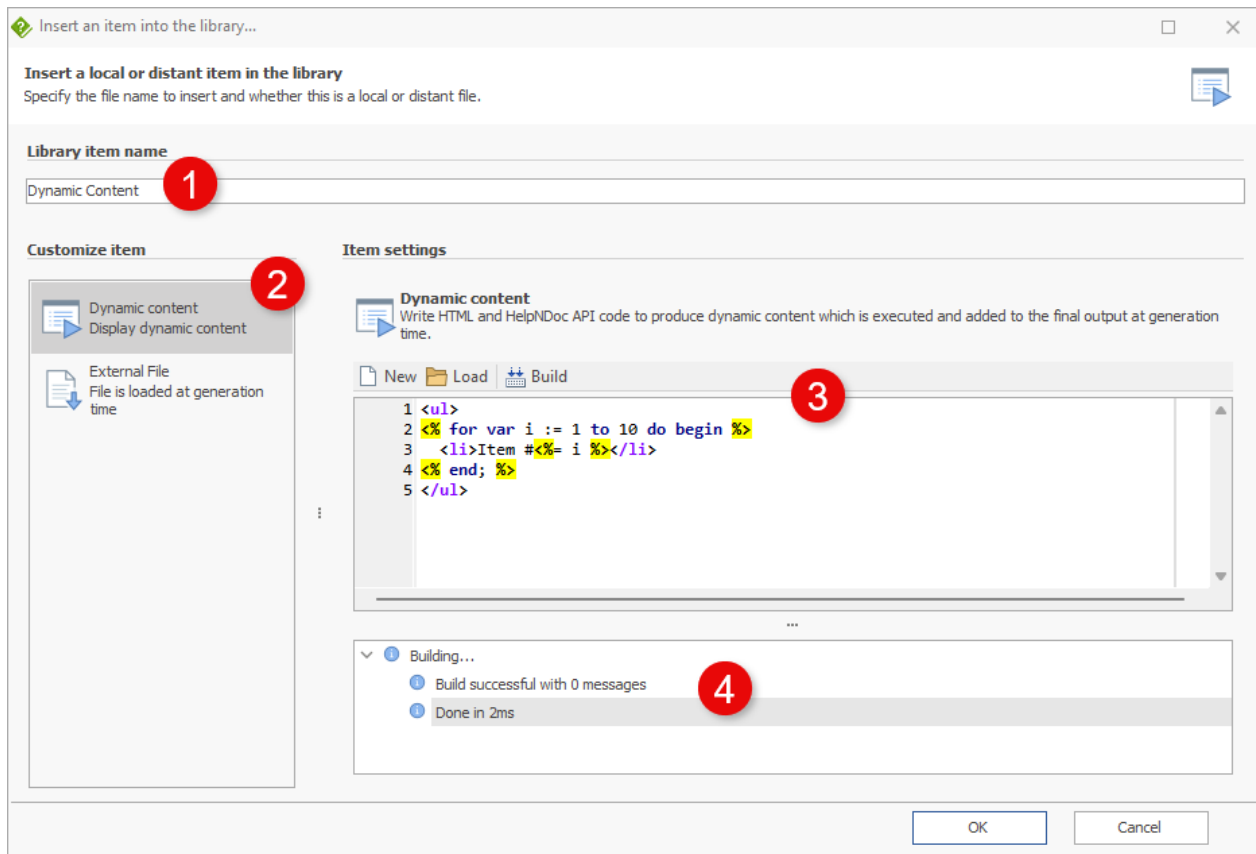
Dynamic content library item

The dynamic content library item provides a way to produce HTML content using HelpNDoc's built-in [scripting methods](#) and [API](#). Dynamic content placed within topics are interpreted at generation time to produce content which is then integrated in the final documentation file. It can be useful in multiple situations:

- Produce personalized documentation based on some conditions, such as topic properties, build settings...
- Topic-specific content display, such as listing children topics, or associated keywords
- Time-sensitive updates, which are only displayed based on the current generation date and time
- Content formatting, such as converting external JSON or XML files to human-readable tables
- And more...

Note: Even though they are based on HTML and scripts, dynamic content are compatible with every documentation formats supported by HelpNDoc, including Word and PDF formats.

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Source

Choose if the content is stored within the project or loaded from an external file at generation time

3. Dynamic content editor

Specify the dynamic content's HTML-based script using the built-in editor. Use the "Build" button to check that the syntax is valid.

4. Build output

Once built, any information, warning or error messages produced during the build process are displayed in the build output log.

Sample dynamic content scripts

List of children topics

When placed within a topic, the following script produces a lists of links to access all its generated direct children topics:

```
<%
  var aChildrenTopicList :=
HndTopicsEx.GetTopicDirectChildrenListGenerated(HndTopics.GetCurrentTopic());
  if (aChildrenTopicList.Length > 0) then
  begin
%>
<ul>
  <% for var nTopic := 0 to aChildrenTopicList.Length - 1 do begin %>
    <li><a href="hnd-topic://<%= aChildrenTopicList[nTopic].id %>"><%=
aChildrenTopicList[nTopic].Caption %></a></li>
    <% end; %>
  </ul>
<% end; %>
```

List of associated keywords

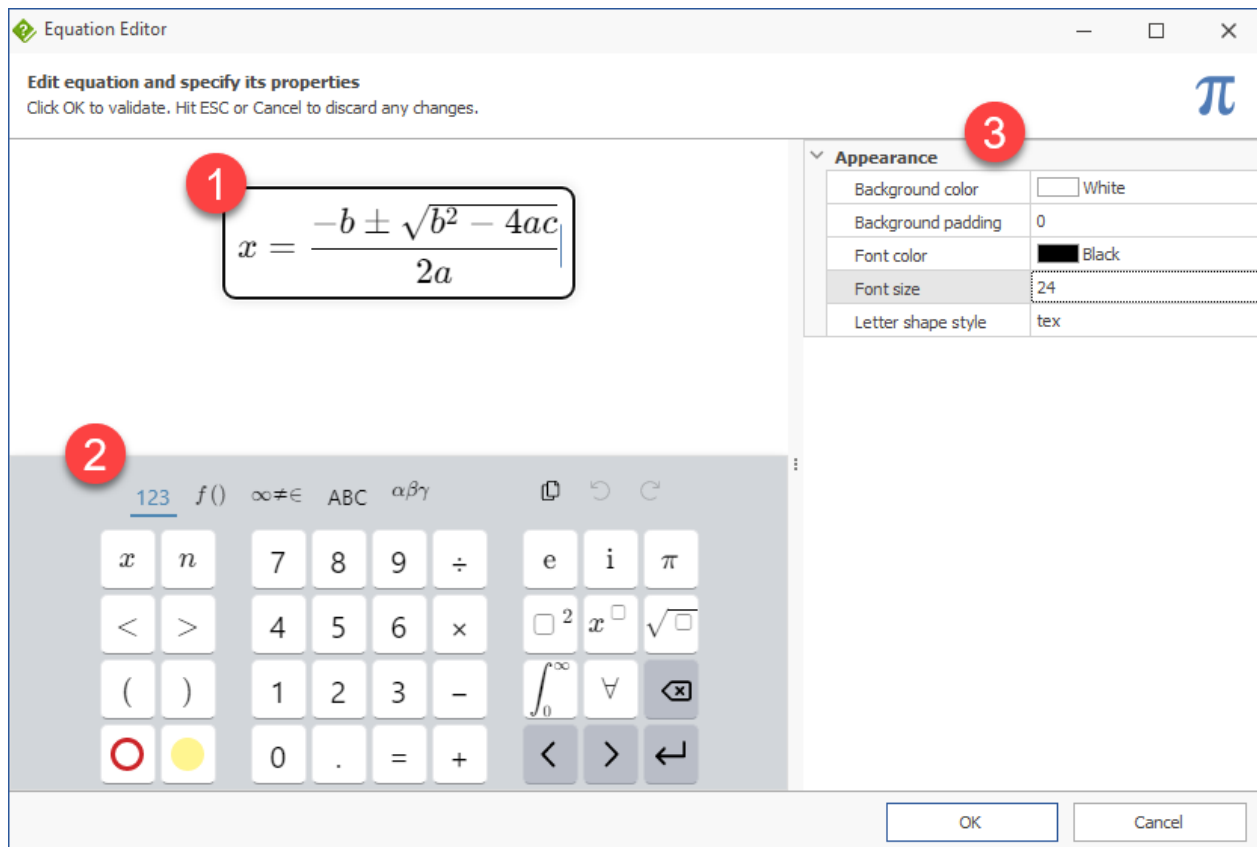
Place this dynamic content in a topic to produce a list of all keywords attached with this topic:

```
<%
  var aKeywordsList :=
HndTopicsKeywords.GetKeywordsAssociatedWithTopic(HndTopics.GetCurrentTopic());
  for var nKeyword := 0 to aKeywordsList.Length - 1 do
  begin
%>
  <span style="background-color: #eee"><%=
HndKeywords.GetKeywordCaption(aKeywordsList[nKeyword]) %></span>
<%
  end;
%>
```

Equation library item

HelpNDoc's built-in equation editor can be used to define mathematical expressions. The editor generates images of the defined expression, which is displayed in all supported documentation formats.

Overview of the user interface



1. Equation editor

Input equation content using either the physical or virtual keyboard.

2. Virtual keyboard

Displayed when the equation is focused, the virtual keyboard can be used to input various notations and features of the equation editor. See also the physical [keyboard shortcuts](#) available to manage the equation.

3. Appearance and properties

Define various settings about the currently edited equation:

Property	Description
Background color	Define the background color of the equation
Background padding	Define a padding around the equation content which is filed by the

	background color																
Font color	Define the color of the equation content																
Font size	Define the size of the font																
Letter shape style	Control which letters are automatically italicized: <div><table><tr><td><i>xyz</i></td><td>ABC</td><td><i>αβγ</i></td><td><i>ΓΔΘ</i></td></tr><tr><td><i>xyz</i></td><td>ABC</td><td><i>αβγ</i></td><td><i>ΓΔΘ</i></td></tr><tr><td><i>xyz</i></td><td>ABC</td><td><i>αβγ</i></td><td><i>ΓΔΘ</i></td></tr><tr><td><i>xyz</i></td><td>ABC</td><td><i>αβγ</i></td><td><i>ΓΔΘ</i></td></tr></table></div>	<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>	<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>	<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>	<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>
<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>														
<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>														
<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>														
<i>xyz</i>	ABC	<i>αβγ</i>	<i>ΓΔΘ</i>														

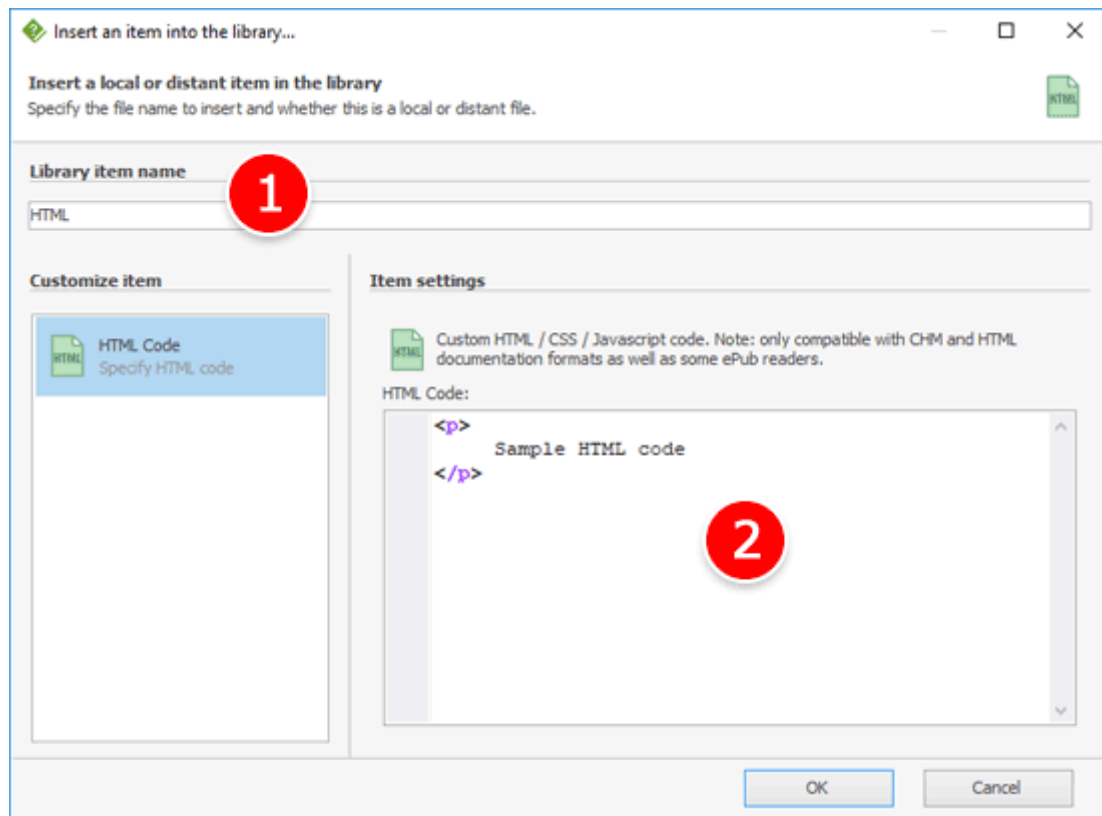
HTML code library item

HTML code library items provides a way to add custom HTML, CSS, JavaScript code to the generated documentation. It can be useful in multiple situations:

- Insert advanced HTML widgets such as accordions, tabs, [YouTube / Vimeo](#) movies...
- Add custom code to customize the behavior or look and feel of the generated documentation

Note: HTML code library items are only compatible with CHM and HTML documentation formats as well as some ePub readers. Extra care must be given to cross-browser compatibility with the included HTML code.

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Code editor

Insert / Edit the custom HTML, CSS and JavaScript code.

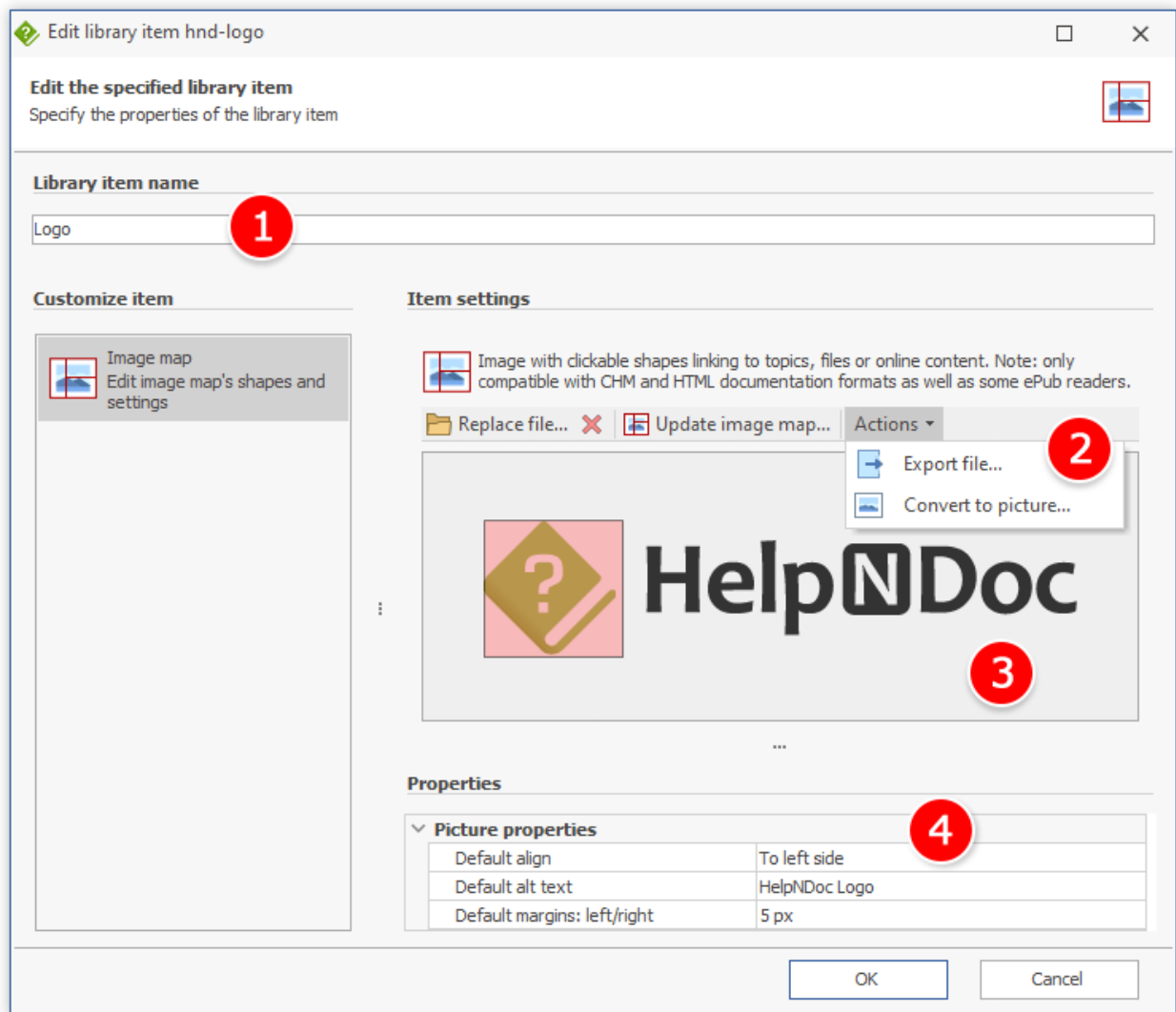
Image map library item

An image map is a special image with [click-able shapes](#) linking to topics, files or on-line content. It can be useful in multiple situations:

- Display a screenshot linking to additional information based on where the user clicks;
- Display a map linking to additional information based on coordinates;

Note: Image maps are only compatible with CHM and HTML documentation formats as well as some ePub readers.

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Commands and fields

Commands available for the image map library item:

- Insert / Replace file: Locate a file on the hard drive or a network location to
- Remove file from project: Remove the content of the file from the project
- Update image map: Show the [image map editor](#) to edit its click-able shapes
- Export file: Export the content of the file to the hard drive
- Convert to picture: Convert an image map to a [picture](#). **Note:** This action can take time on large projects as it needs to analyze the whole project and convert every instances of this

image map. This action can't be undone.

3. Preview

A preview of the library item is displayed. It is possible to drag and drop a file from a third party application such as the Windows Explorer into the preview to include it or replace the existing file.

4. Properties

Custom properties for this library item.

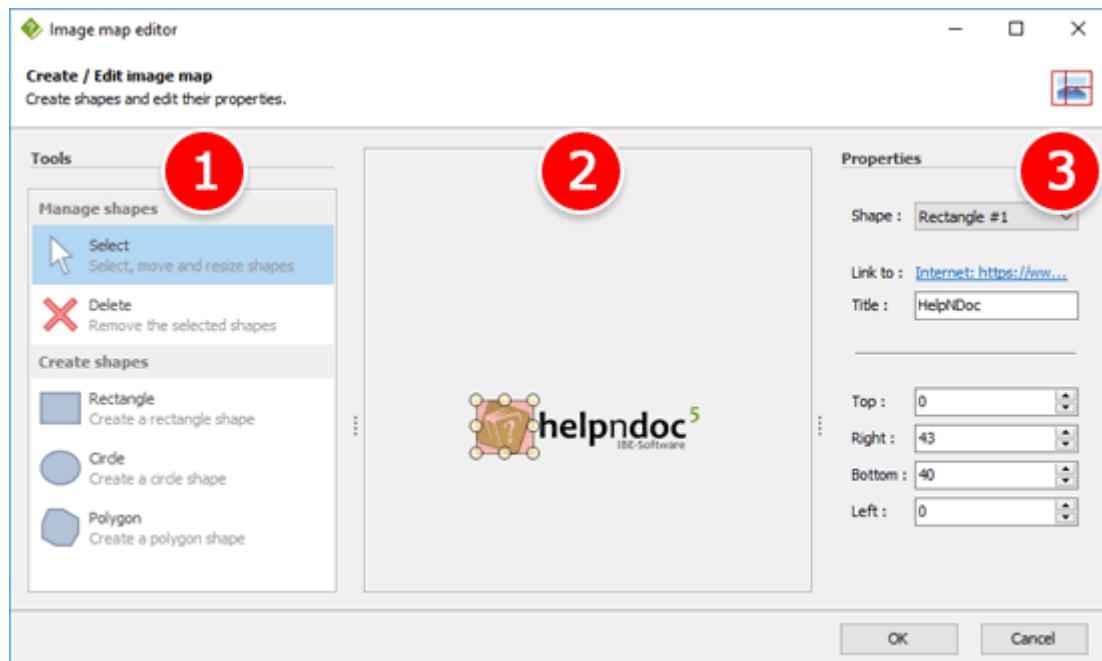
Property	Description
Default align	Specify the default alignment of the picture when added to a topic.
Default alt text	Specify the default alt text of the picture when added to a topic.
Default margins	Specify the default margins for the picture when added to a topic.
Default padding	Specify the default padding for the picture when added to a topic.
Default width	Specify the default width for the picture when added to a topic. Note: Clear this value to use the picture's default width.
Default height	Specify the default for the picture when added to a topic. Note: Clear this value to use the picture's default height.

Image map editor

The image map editor is used to draw and update shapes on an image, and set their properties such as where they are linking to. The image map editor can be accessed as follows:

- [Create or edit an image map library item](#) to show the library item edit window
- Click the "Update image map" button

Overview of the user interface



1. Tools

The tools can be used to create, select and manipulate shapes on the image map:

- **Select**: click any shape on the editor to select it. Use its resizing handles to resize it
- **Delete**: delete the currently selected shape
- **Rectangle**: create a rectangle shape
- **Circle**: create a circle shape
- **Polygon**: create a polygon shape. Click once to add a point, twice to close the polygon

2. Editor

Shows a preview of the image map with the various shapes. Select a shape to display the resizing handles and edit its properties.

3. Properties

Define the currently selected shape's properties such as link, title, and position.

Movie library item

Any movie required by the documentation project is first placed in the library. It can then be included in any number of topics while being centrally managed from the library: updating the movie in the library will automatically update all instances of that movie in any topic where it has

been placed.

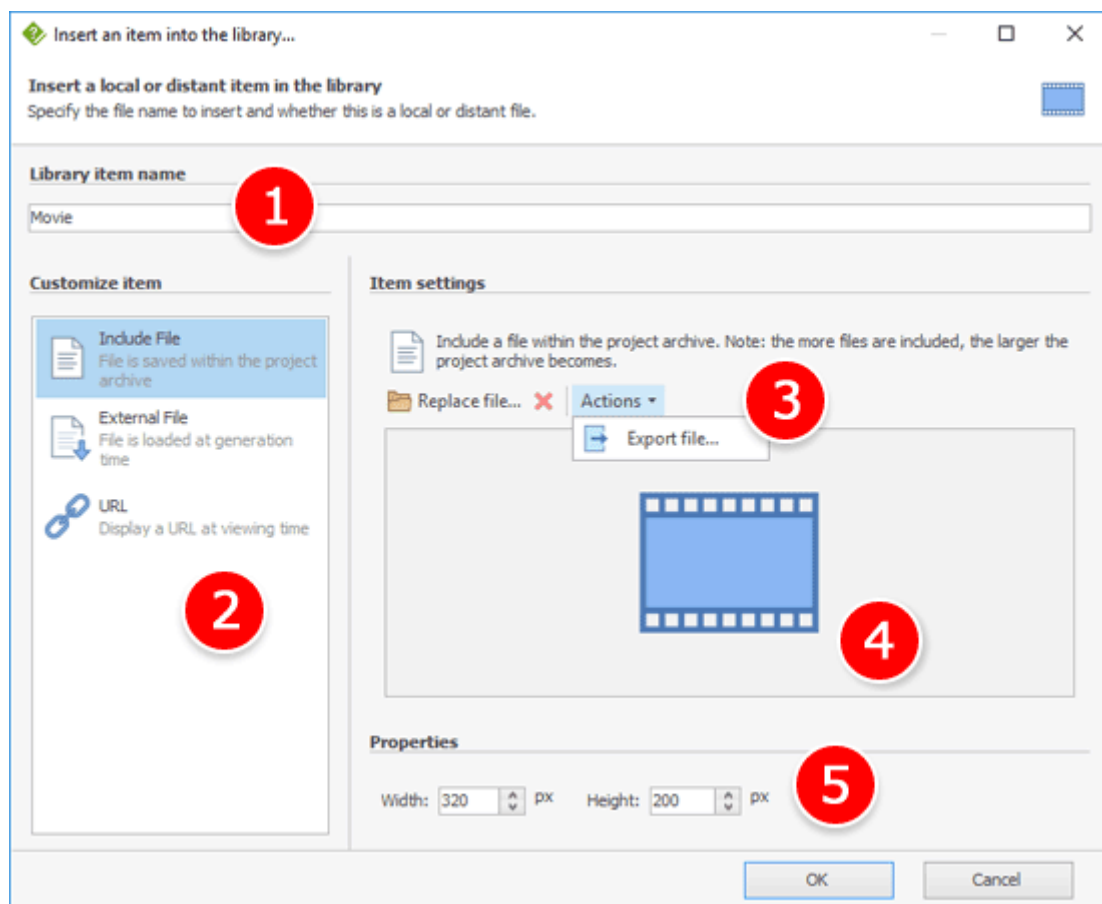
Notes:

- Movies can be huge files and including them in the project can lead to significant saving, loading and generation time;
- Movies are only compatible with CHM and HTML documentation formats as well as some ePub readers. Also, the movie format has to be considered carefully: it might not render on a device where that specific format's codec hasn't been installed.

For those reasons, we highly recommend the use of online movie hosts such as [Youtube](#) or [Vimeo](#):

- The project and generated documentation will be smaller and faster to generate;
- The movie host takes care about transcoding the movie to the correct format so that it is visible by anyone.

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Customize item

A movie can either be stored within the project or linked from an external location based on the project's requirements. Choosing how it is stored can be decided individually for each item based on pros and cons for that specific item and / or overall documentation project.

Storage	Description	Pros	Cons
Include File	Item file is stored within the project archive	<ul style="list-style-type: none"> Item is always available even when the project is moved to a different location Sharing project is easier as the HND project file contains the item's content The item is always available to the end-user as it is stored with or within the generated documentation file 	<ul style="list-style-type: none"> HND project file becomes larger with each included item Replacing an item involves locating it in the library and updating it Items can't be shared between multiple projects Generation time is slower as the file needs to be copied / included in the generated documentation
External File	Item file is stored anywhere on the hard drive or a network location, and included at generation time. Note: The external file path can be absolute, or relative to the HND project file location	<ul style="list-style-type: none"> HND project file is smaller as only the path to the external file is stored Updating the item on the hard drive or network location will update it the next time the documentation is built 	<ul style="list-style-type: none"> Item location must be updated in the library when the HND project file is moved, or the item must be moved with the project Sharing a project requires

		<ul style="list-style-type: none"> • Items can be shared between multiple projects: each project will include it when needed • The item is always available to the end-user as it is stored with or within the generated documentation file 	<p>all external items to be shared too</p> <ul style="list-style-type: none"> • Sharing a project might require an update of all external items' paths • Generation time is slower as the file needs to be copied / included in the generated documentation
URL	Item file is stored on a web server, and displayed at viewing time	<ul style="list-style-type: none"> • Item is always available even when the project is moved to a different location • HND project file is smaller as only the URL is stored • Updating the item on the web server automatically updates any client application requesting that item from now on • Items can be shared between multiple projects: each project will display it when needed • Generation time is 	<ul style="list-style-type: none"> • A working Internet connection is required at viewing time to request and display the item • Only compatible with CHM, HTML and Markdown documentation formats as well as some ePub readers

		faster as the file is not copied / included in the generated documentation	
--	--	--	--

3. Commands and fields

Include File

- Insert / Replace file: Locate a file on the hard drive or a network location to
- Remove file from project: Remove the content of the file from the project
- Export file: Export the content of the file to the hard drive

External File

- File path: Absolute or relative path of the file to include at generation time

URL

- URL: Internet link to the file to display at viewing time

Specifying a URL with a YouTube or Vimeo domain name will automatically generate the proper embed code.

4. Preview

For included files, an icon of the library item is displayed. It is possible to drag and drop a file from a third party application such as the Windows Explorer into the preview to include it or replace the existing file.

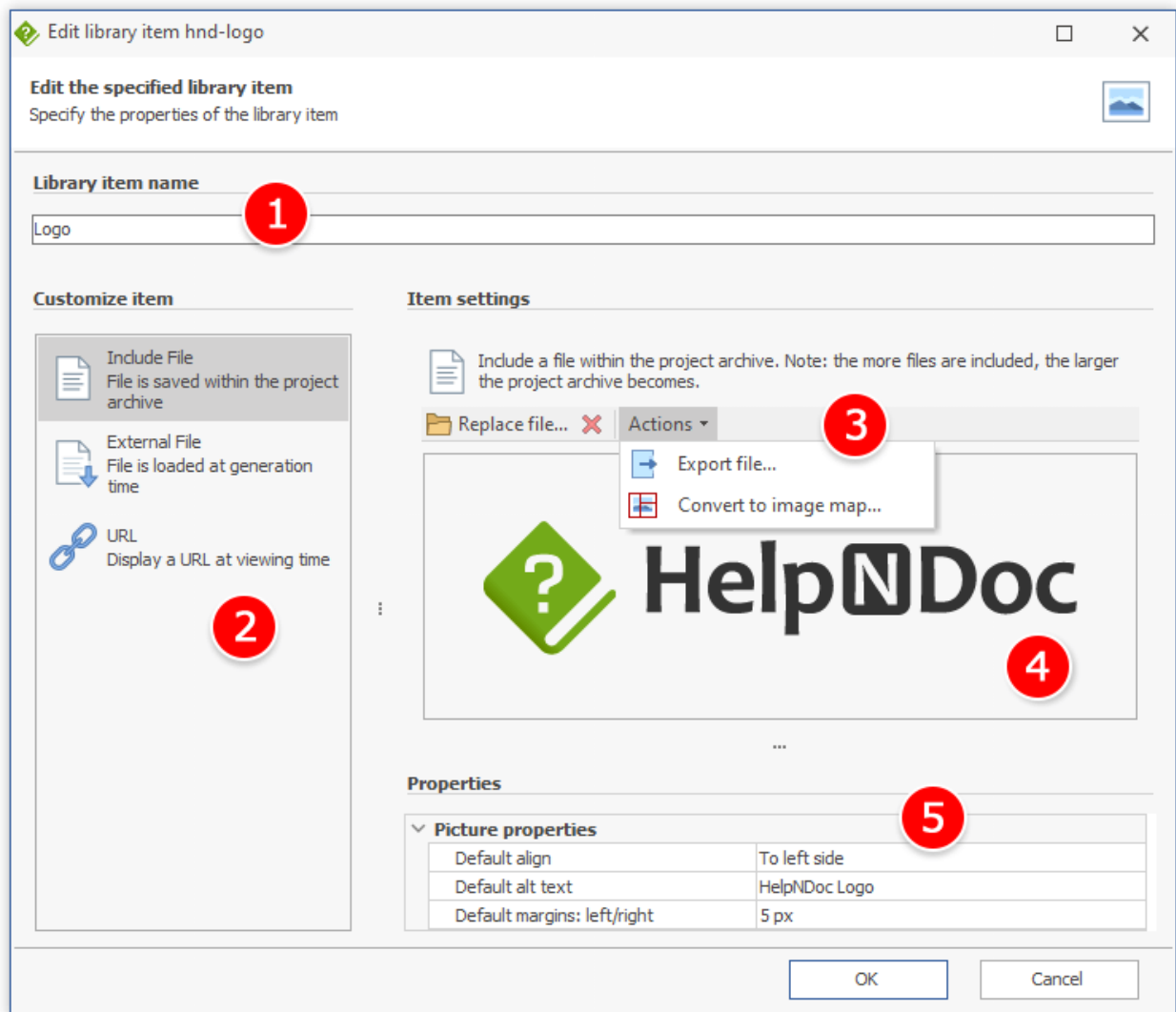
5. Properties

Specific properties about the movies such as its width and height.

Picture library item

Any picture (photo, screenshot...) required by the documentation project is first placed in the library. It can then be included in any number of topics while being centrally managed from the library: updating the picture in the library will automatically update all instances of that picture in any topic where it has been placed.

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Customize item

A picture can either be stored within the project or linked from an external location based on the project's requirements. Choosing how it is stored can be decided individually for each item based on pros and cons for that specific item and / or overall documentation project.

Storage	Description	Pros	Cons
Include File	Item file is stored within the project	<ul style="list-style-type: none"> Item is always available even when 	<ul style="list-style-type: none"> HND project file becomes larger

	archive	<p>the project is moved to a different location</p> <ul style="list-style-type: none"> • Sharing project is easier as the HND project file contains the item's content • The item is always available to the end-user as it is stored with or within the generated documentation file 	<p>with each included item</p> <ul style="list-style-type: none"> • Replacing an item involves locating it in the library and updating it • Items can't be shared between multiple projects • Generation time is slower as the file needs to be copied / included in the generated documentation
External File	Item file is stored anywhere on the hard drive or a network location, and included at generation time. Note: The external file path can be absolute, or relative to the HND project file location	<ul style="list-style-type: none"> • HND project file is smaller as only the path to the external file is stored • Updating the item on the hard drive or network location will update it the next time the documentation is built • Items can be shared between multiple projects: each project will include it when needed • The item is always available to the end-user as it is 	<ul style="list-style-type: none"> • Item location must be updated in the library when the HND project file is moved, or the item must be moved with the project • Sharing a project requires all external items to be shared too • Sharing a project might require an update of all external items' paths • Generation time

		stored with or within the generated documentation file	is slower as the file needs to be copied / included in the generated documentation
URL	Item file is stored on a web server, and displayed at viewing time	<ul style="list-style-type: none"> • Item is always available even when the project is moved to a different location • HND project file is smaller as only the URL is stored • Updating the item on the web server automatically updates any client application requesting that item from now on • Items can be shared between multiple projects: each project will display it when needed • Generation time is faster as the file is not copied / included in the generated documentation 	<ul style="list-style-type: none"> • A working Internet connection is required at viewing time to request and display the item • Only compatible with CHM, HTML and Markdown documentation formats as well as some ePub readers

3. Commands and fields

Include File

- Insert / Replace file: Locate a file on the hard drive or a network location to

- Remove file from project: Remove the content of the file from the project
- Edit image: Open the built-in [image editor](#) to edit the image.
- Convert to image map: Convert a picture to an [image map](#). **Note:** This action can take time on large projects as it needs to analyze the whole project and convert every instances of this picture. This action can't be undone.
- Export file: Export the content of the file to the hard drive

External File

- File path: Absolute or relative path of the file to include at generation time

URL

- URL: Internet link to the file to display at viewing time

4. Preview

For included files, a preview of the library item is displayed. It is possible to drag and drop a file from a third party application such as the Windows Explorer into the preview to include it or replace the existing file.

5. Properties

Custom properties for this library item. Use the "Reset" button to reset to the default value. Default properties can be applied to every instances within the project by clicking the "Apply to every instances throughout the project" button.

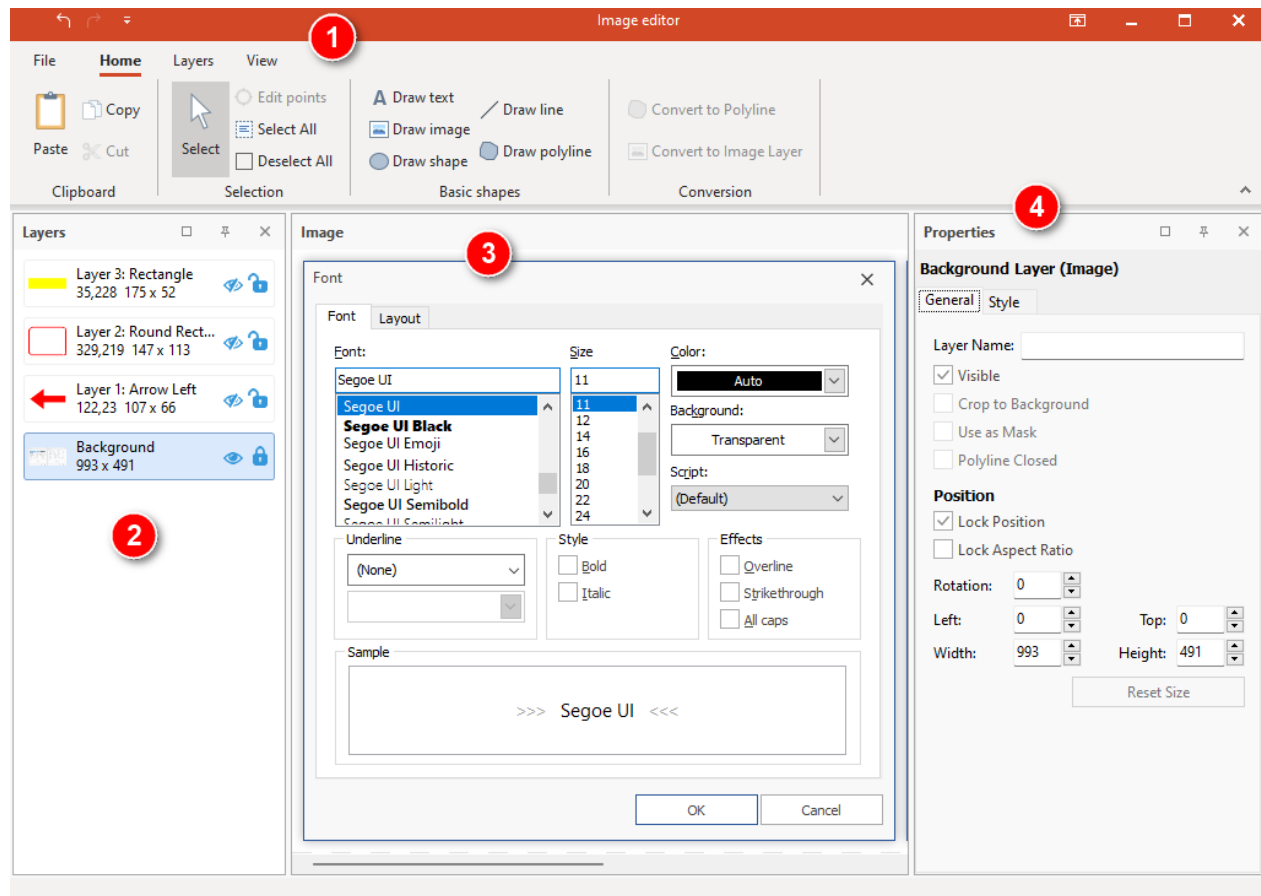
Note: this will override any manual settings defined in any instances.

Property	Description
Default align	Specify the default alignment of the picture when added to a topic.
Default alt text	Specify the default alt text of the picture when added to a topic.
Default margins	Specify the default margins for the picture when added to a topic.
Default padding	Specify the default padding for the picture when added to a topic.
Default width	Specify the default width for the picture when added to a topic. Note: Clear this value to use the picture's default width.

Default height	Specify the default for the picture when added to a topic. Note: Clear this value to use the picture's default height.
----------------	---

Image editor

Pictures which are included within the library can be edited using the built-in image editor.



1. Ribbon bar

Use the ribbon toolbars to access the various actions to create and manage layers, view, as well as contextual tools.

2. Layers editor

The image is made of multiple layers which can be edited from the layers editor. The top layer is the one in the front while the bottom layer is the one at the back, or background. Layers can be reorganized by dragging them to another position. The "Eye" button can be used to toggle the layer's visibility while the "Lock" button can be used to lock the layer.

3. Image editor

Use the image editor to preview the final image, and manage layers by moving, resizing, rotating... them.

4. Properties

Displays the contextual properties of the selected layer(s). Use the editors to customize the current selection.

Stamps editor

Insert any of the pre-made stamps in the image by dragging them and dropping them into the image canvas.

Select any numbers of layers, then click "Add stamp" to convert them to a reusable stamp.

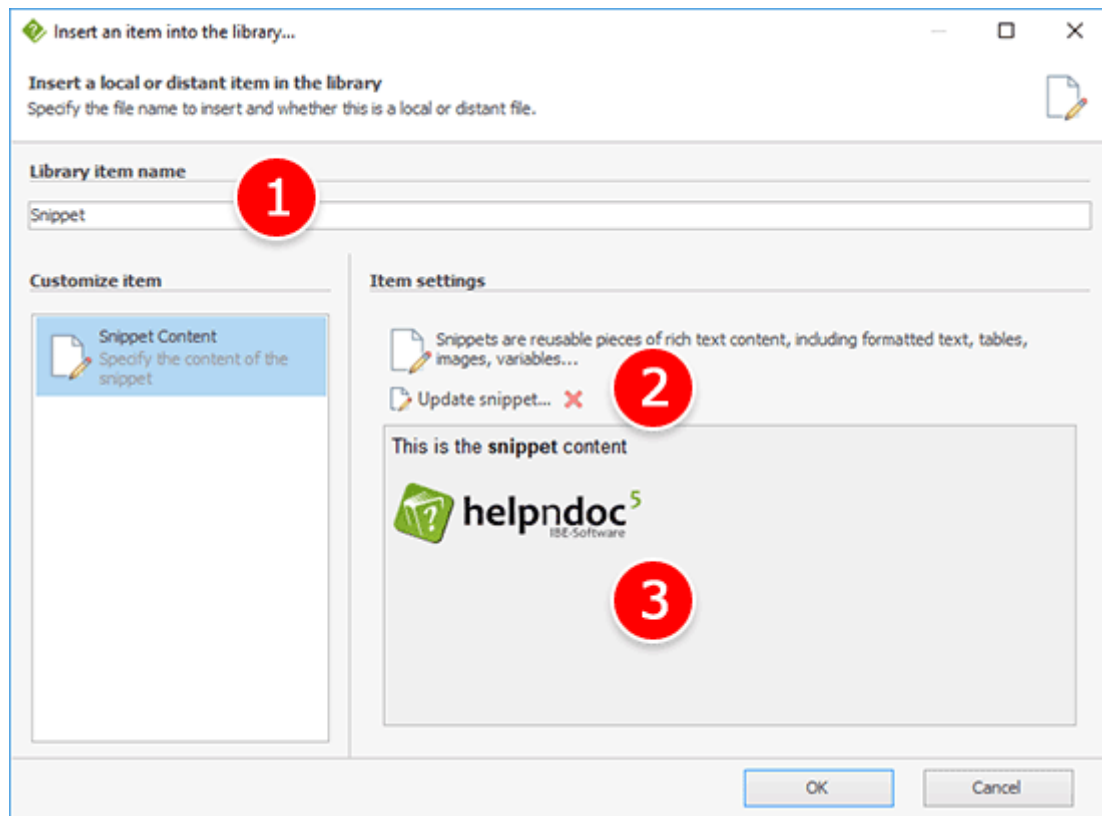
Select a custom stamp then click "Delete stamp" to delete it. **Note:** System stamps can't be deleted.

Snippet library item

Snippets are very similar to variables: the snippet is placed within topics and will be replaced by its current value at generation time. They can be useful in multiple situations:

- A piece of content needs to be repeated multiple times within the project;
- A work in progress can first be written as a snippet included in the topic, and later copy / pasted as the topic content;

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Commands

Commands available for the image map library item:

- Update snippet: show the [Snippet editor](#) to edit the current content of the snippet
- Clear snippet content: clear the content of the snippet

3. Preview

Current content of the snippet

Convert the topic's content to a snippet

It is possible to quickly convert the selected topic's content to a snippet, to be able to reuse it elsewhere:

- Select the desired content in the topic editor
- Right click on it

- Click "Convert to snippet..."

HelpNDoc will ask if the selected content should be replaced by the newly created snippet. Choose either:

- Yes: the selected content is replaced by the newly created snippet
- No: the selected content remains as-is and a new snippet is created in the library
- Cancel: cancel the operation without creating a new snippet

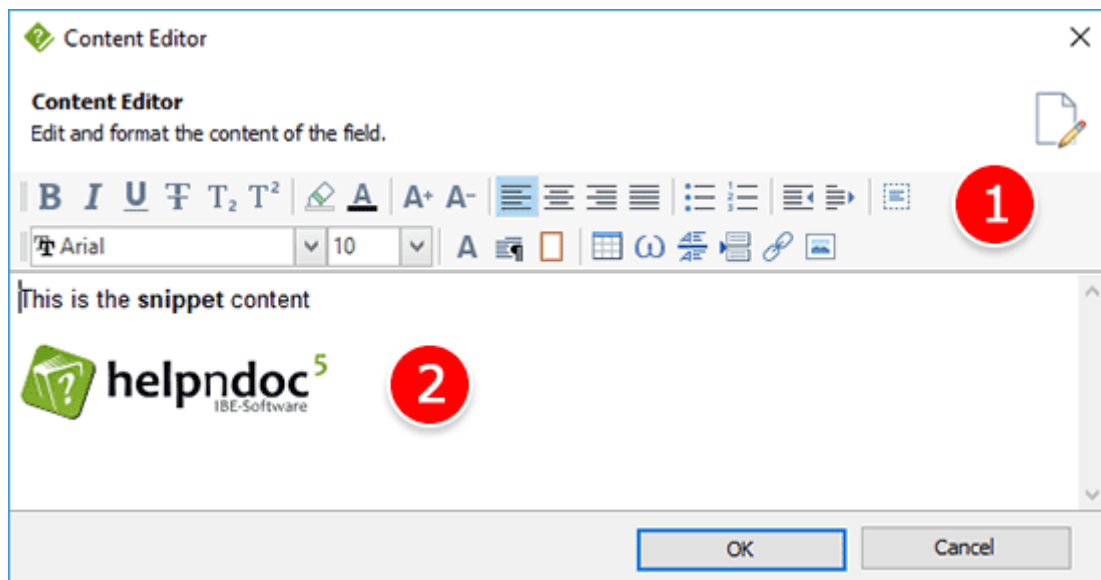
Snippet editor

The snippet editor is used to edit the rich text content of the snippet. Snippets can contain:

- Formatted text (font, color, bold, italic, alignment...)
- Bullets and numbering
- Tables
- Symbols
- Links
- Images

Note: Images are contained within the snippet and not within the library. The same images added to multiple snippets will increase the project and generated documentation size.

Overview of the user interface



1. Actions

Use the toolbar actions to format the content of the snippet and insert content such as tables, images...

2. Editor

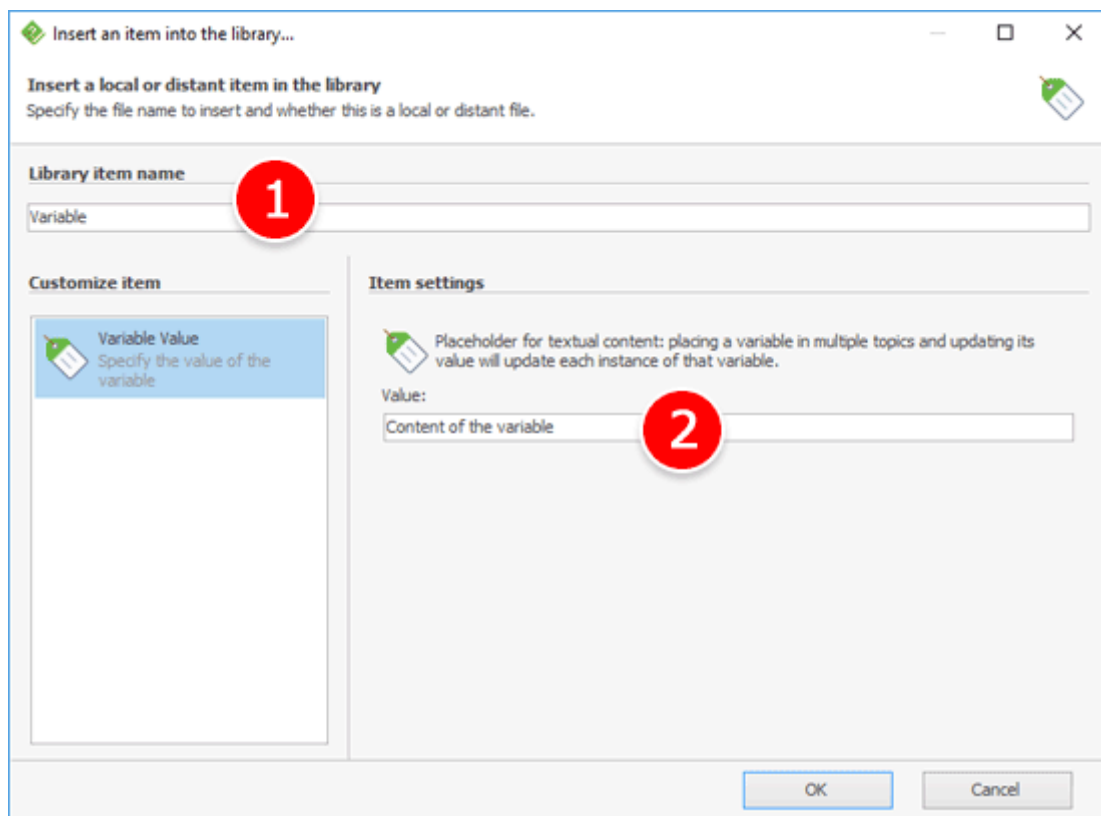
Use the editor to customize the content of the snippet.

Variable library item

Variables are placeholders for textual content. The variable is placed within topics and will be replaced by its current value at generation time. They can be useful in multiple situations:

- When an information is not yet known or is subject to change, such as a product which is in the development phase and whose name is not yet known;
- When the generated documentation has multiple targets: it is possible to override variables for each documentation build generated by HelpNDoc thus generating multiple variations of a documentation;

Overview of the user interface



1. Library item name

Choose a unique name for that library item.

2. Variable value

Current value of the variable.

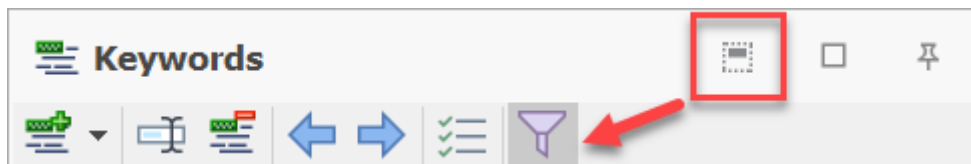
Using the keywords editor

Keywords are words or short sentences used to tag or index one or multiple topics. HelpNDoc offers the possibility to define a keyword hierarchy where each keyword can be associated with one or multiple topics. Keywords are alphabetically and hierarchically ordered: a keyword can contain one or more children keywords.

See the [How to access the keywords panel](#) step-by-step guide.

Display the keywords toolbar

To simplify keywords management, instead of navigating to the global ribbon tab to manage keywords, an optional toolbar can be shown at the top of the keywords panel. To toggle it, click the "Show / Hide keywords toolbar" in the title bar of the keywords panel.



Create keywords

To create a first-level keyword:

- Create the top part of the "Add keyword" button in the "Keywords" group of the "Home" ribbon tab
- The keyword is added in the list ready to be named: enter a name and validate using the Enter keyboard shortcut

To create a child keyword, first select the parent keyword then:

- Click the bottom part of the "Add keyword" button in the "Keywords" group of the "Home" ribbon tab
- Click "Add child keyword"
- The keyword is added in the list ready to be named: enter a name and validate using the Enter keyboard shortcut

See the [How to create a keyword](#) step-by-step guide.

Rename keywords

To rename a keyword:

- Select the keyword in the keywords panel
- Click the "Rename" button in the "Keywords" group of the "Home" ribbon tab
- Enter a name and validate using the Enter keyboard shortcut

See the [How to rename a keyword](#) step-by-step guide.

Delete keywords

To delete obsolete keywords:

- Select the keyword in the keywords panel
- Click the "Delete" button in the "Keywords" group of the "Home" ribbon tab

See the [How to delete a keyword](#) step-by-step guide.

Associate with topics and remove association

Keywords with checked boxes are associated with the currently selected topic. Keywords with unchecked boxes are not associated with it. They might be associated with other topics or not used anymore: you can use the [project analyzer](#) to check for unused keywords.

To associate / remove association between a keyword with a topic:

- Select the topic in the table of contents
- Click the check-box in front of the desired keyword to check or un-check it
- Alternatively, you can use the "Associate with topic" check button in the "Keywords" group of the "Home" ribbon tab

To manage the topics associated with a specific keyword, you can use the [Manage keyword association](#) window.

See the [How to associate keywords with topics](#) step-by-step guide.

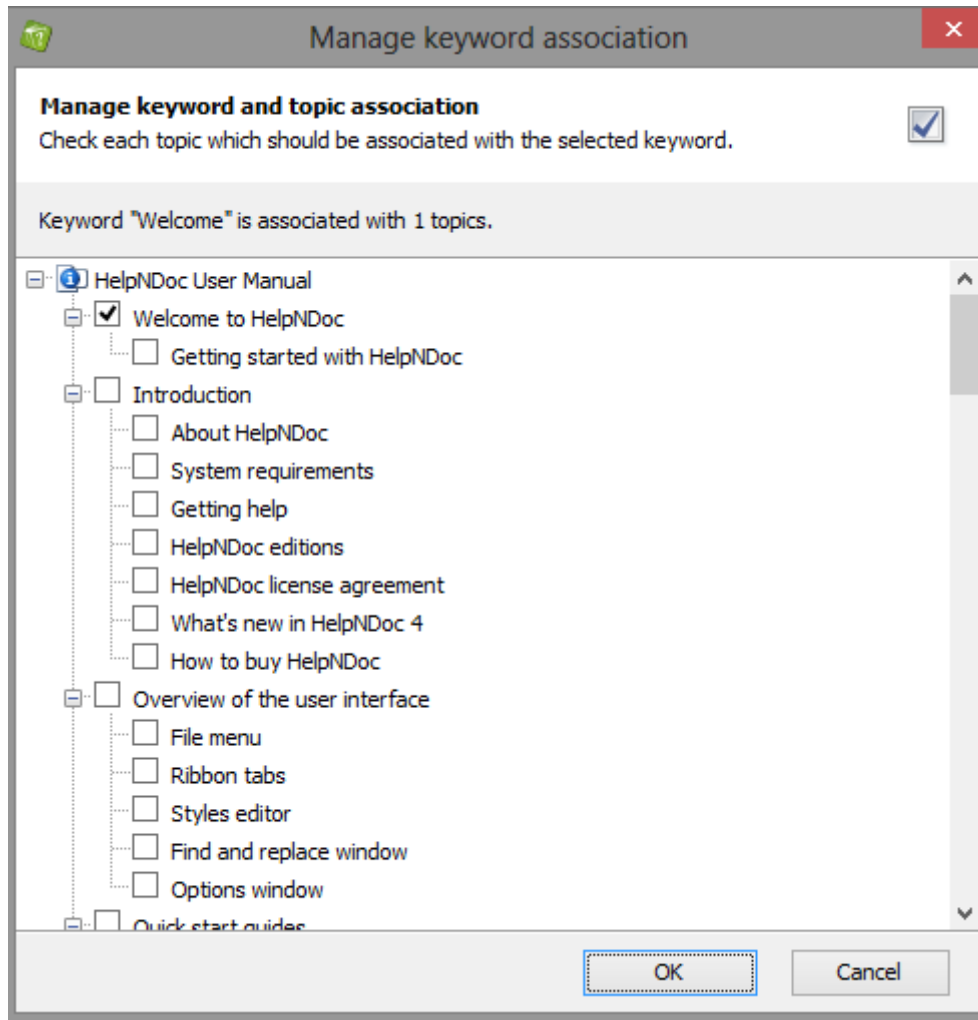
Manage keyword association

Keywords' association with the current topic can rapidly be done using the check-box in front of the keyword.

To rapidly manage every topics associated with a specific keyword, select the keyword then click the "Associated topics" button in the "Keywords" group of the "Home" ribbon tab to open the "Manage keyword association" window.

Manage keyword association window

This window lists all topics within the project and adds a checked box before the caption of the topics associated with the currently selected keyword.



To associate the currently selected keywords with additional topics, check the boxes before the caption of those topics. Un-check those boxes to remove the association.

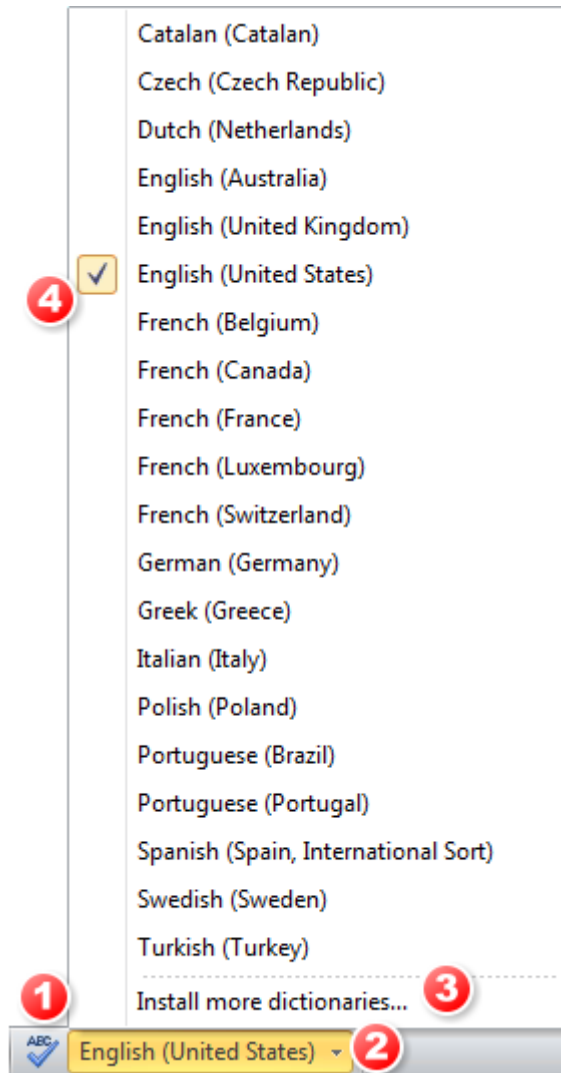
To save the modified associated topic list, hit the OK button at the bottom of the window.

See the [How to manage the association between a keyword and topics](#) step-by-step guide.

Using the spell checker

The live spell checker is an integral part of HelpNDoc, covering any input made throughout the user interface: once a potential spelling error has been identified, the live spell checker will underline the problematic word with a red line. Right clicking on the word will give a list of

possible alternative words, and options to ignore it or add it to the user dictionary.



1. Spelling options

This shows the spelling options dialog which is where the spell checker's settings can be configured.

2. Active dictionaries

This indicates the currently active dictionaries. A click on that button shows a list of all installed dictionaries on the current computer as well as options to install new dictionaries and change currently active ones.

3. Install dictionaries

New dictionaries can be downloaded from the [OpenOffice.org extensions](https://extensions.openoffice.org/) web-site and installed using this dialog: just browse for the *.ext file you saved on your computer and HelpNDoc will install it and add it to the list

4. Managing dictionaries

Dictionaries with a check mark are the ones currently activated and used by HelpNDoc to spell

check the current project. To activate a dictionary, click on it to check it. To deactivate a dictionary, click on it to un-check it. HelpNDoc supports multiple dictionaries activated at the same time: when activating a dictionary, it won't deactivate the currently activated ones.

See the following step-by-step guides:

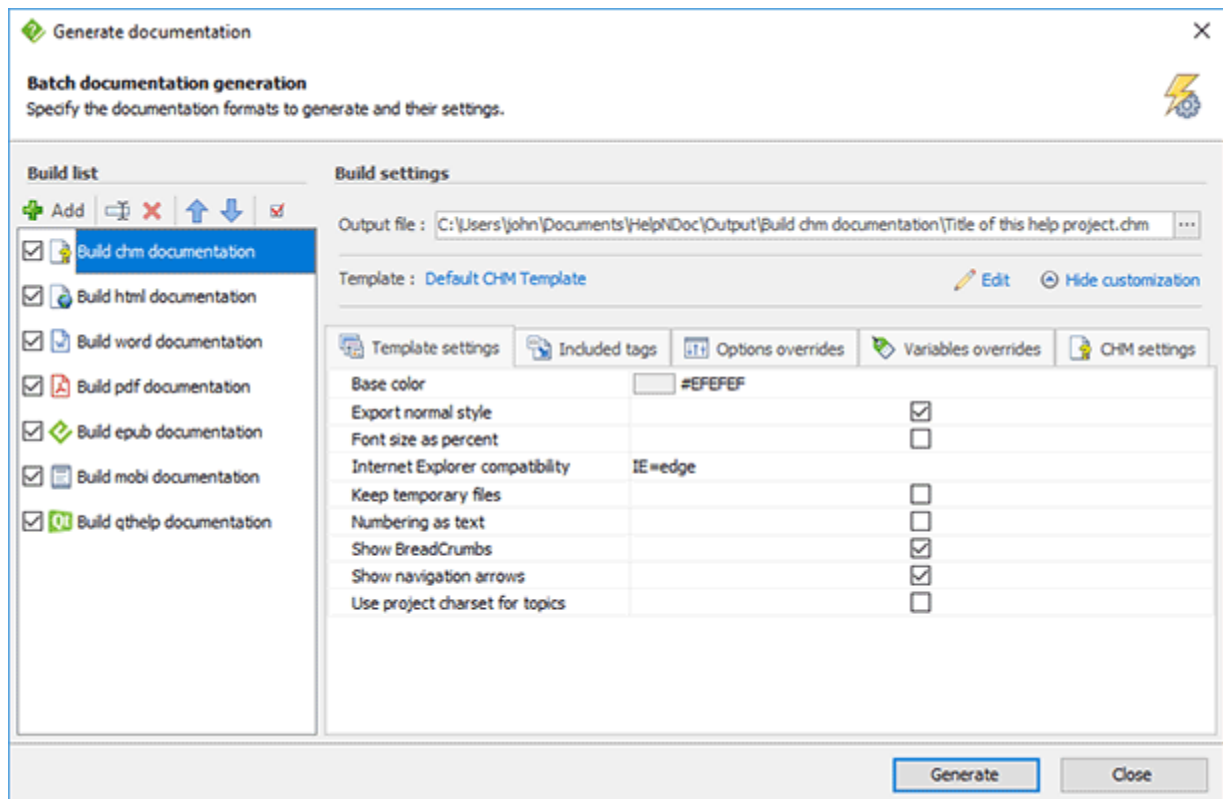
[How to check the spelling in HelpNDoc](#)

[How to activate and deactivate a dictionary in HelpNDoc](#)

[How to install a new dictionary in HelpNDoc](#)

[How to maintain your spell check settings in HelpNDoc](#)

Publishing documentation



From the "Home" ribbon tab, click the top part of the "Generate help" button to show the "Generate documentation" window. From this window, you can specify:

- The kinds of documentation formats to generate by adding builds and enabling them
- The output path of the final documentation
- The template and settings to use for each individual build
- The order of the build execution
- Custom [template settings](#), tags settings, project options and variables overrides as well as build specific settings

HelpNDoc will then process the templates and generate the documentation for each build accordingly.

See also: [How to publish your documentation](#)

HelpNDoc affords you tremendous flexibility to support dynamic requirements when you publish your documentation. After you've completed your documentation, you can publish it in a variety of formats using a range of options.

See also: [How to create a new documentation output to be published](#)

Your HelpNDoc documentation can be published in multiple formats. It can also be published multiple times with different content and settings in each of those formats. Let's see how easily this can be done.

See also: [How to rename a publishing output in HelpNDoc](#)

You can define the outputs that are generated when you publish your documentation. After you've created a publishing output, it is displayed in your build list. When you publish your documentation, it is displayed with its assigned name. To support your specific requirements, you may update this name for any output at any time.

See also: [How to delete a build in HelpNDoc](#)

You can define the builds that are generated when you publish your documentation. These builds are displayed in your build list and can be enabled or disabled at will. When a build becomes obsolete, it is possible to remove it from your build list.

See also: [How to reorder your publishing outputs in HelpNDoc](#)

You can define the order of your outputs in your build list. This order determines the order in which your documentation is generated. This order can also dictate which settings are applied when you publish documentation using 'Quick generate.'

See also: [How to enable your publishing outputs in HelpNDoc](#)


HelpNDoc gives you the flexibility and control to determine which of your builds are published when you generate documentation. When you enable a build, it is generated when you use your build list. You also have the option to temporarily disable a build to prevent it from being generated. This flexibility permits you to maintain builds in your build list without requiring you to publish them each time you generate documentation.

See also: [How to define build settings in HelpNDoc](#)

When you're ready to generate documentation, HelpNDoc allows you to define the location of your output files and select the templates that are used to generate your documentation. In addition, you can further customize settings such as color, font size, numbering style, and create conditional tags to tailor your documentation to support specific requirements.

Template settings

A template can define multiple custom settings (variables) for quick and easy modification from HelpNDoc's "generate documentation" dialog. They are used to customize parts of the template, such as custom colors, string translations, custom logos, optional elements... and provide a fast way to customize a template without altering its code. See also: [Variables](#) using the template editor, and [Template variables](#) for low level details.

Template settings	
Sitemap priority: others	0.8
Sitemap: Generate ?	<input type="checkbox"/>
Table of content expand level	1
Table of contents tab title	Contents
Table of contents width	350
Theme	Light-Blue 
Translation for "Close"	Close
Translation for "Incorrect or corrup..."	Incorrect or corrupt search data. Please check your HelpNDoc template
Translation for "Loading..."	Loading...
Translation for "No results"	No results
Specify the theme to use. (default: Light-Blue)	

To access the template settings for a specific build:

- Select the build in the build list of the [Generate documentation dialog](#)
- Select the template to use for that build
- If the "Template settings" tab is not visible, click "Customize"

See the [Customize documentation formats](#) topic to learn more about some of the available template settings.

Customize documentation formats

In addition to being able to [create custom templates](#), it is possible to easily and rapidly customize HelpNDoc's default templates. To access the template settings:

- From HelpNDoc's "Home" ribbon tab, locate the "Project" group
- Click the top part (without the arrow) of the "Generate help" button to display the "Generate documentation" window
- Select the build to customize in the "Build list" on the left
- Click "Customize" on the right if the customization tabs are not visible
- Go to the "Template setting" tab

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

Here are some of the documentation formats which can be customized:

- [CHM documentation settings](#)
- [HTML documentation settings](#)
- [Word documentation settings](#)

- [PDF documentation settings](#)
- [ePub documentation settings](#)
- [Kindle / Mobi documentation settings](#)
- [Qt help documentation settings](#)
- [Markdown documentation settings](#)

CHM documentation settings

Some of the [templates settings](#) available for CHM builds are:

Field	Description
Base color	The base color of the theme
Clean output directory	Optionally clean the output directory before generation starts. Warning: this will delete any file and folder in the output directory.
Custom CSS	Add custom CSS in all generated HTML files
Custom JavaScript	Add custom JavaScript code in all generated HTML files
Export normal style	Define the normal style as the default style. Use web browser's default otherwise
Font size as percent	Use percent values for font sizes
Footer (HTML)	Specify the HTML content of the footer displayed at the bottom of each page
Force image size generation	Generated HTML code will contain image's width and height to avoid content shifting and enhanced Core Web Vitals
Force image sizes as Inches	Use Inches instead of Pixels for image sizes. This can produce better results for high resolution screens
Internet Explorer compatibility	Choose which version of Internet Explorer is used to display topics. Newer versions have better CSS / JS support but might

	not be available with each OS
Keep temporary files	Keep the temporary files needed by the compiler to build the final documentation
Library item directory	Define the sub-directory where library items are generated
Numbering as text	Use text instead of OL/LI elements when generating lists
Show BreadCrumbs	Show or hide the breadcrumbs at the top of each topic
Show navigation arrows	Show or hide the navigation arrows at the top of each topic
Use project charset for topics	Use the project encoding to generate topic files instead of UTF-8. This can be useful to fix problems for some East-European and Asian languages

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

HTML documentation settings

Some of the [templates settings](#) available for HTML builds are:

Field	Description
Clean output directory	Optionally clean the output directory before generation starts. Warning: this will delete any file and folder in the output directory.
Custom CSS	Add custom CSS in all generated HTML files
Custom JavaScript	Add custom JavaScript code in all generated HTML files
Export normal style	Define the normal style as the default style. Use web browser's default otherwise
Font size as percent	Use percent values for font sizes

Footer (HTML)	Custom HTML code which will be added at the bottom of each topic
Force image size generation	Generated HTML code will contain image's width and height to avoid content shifting and enhanced Core Web Vitals
Google Analytics Id	Google Analytics property tracking Id to collect user stats. Google Analytics code is not added if Id is not specified
Google Tag Manager Id	Google Tag Manager (GTM) container ID which installs marketing tags without modifying the documentation's code. See Setup Google tag manager
Inline table of content's width	Define the preferred width of the inline table of contents. Note: can be any CSS units such as "auto", "250x", "50vw"...
Keywords expand level	The default keywords tree expansion level: 1 will expand root level keywords, 2 will also expand first level keywords...
Keywords tab title	The title of the keywords tab. See: How to localize your documentation output
Library item directory	Define the sub-directory where library items are generated
Link format to anchors override	Override the template defined link format to anchors. Available variables include %topicid%, %helpid% and %anchorname%. See Handle the generated topic links
Link format to topics override	Override the template defined link format to topics. Available variables include %topicid%, %helpid% and %anchorname%. See Handle the generated topic links
Logo	A custom image from the project's library to use as a logo. See also "Logo URL". Note: if both "Logo" and "Logo URL" are indicated, "Logo URL" will be used
Logo alt	Alternative text for the logo

Logo link	Link to redirect to when clicking the logo
Logo URL	The URL of a picture to use as a logo. See also "Logo". Note: if both "Logo" and "Logo URL" are indicated, "Logo URL" will be used
Numbering as text	Use text instead of OL/LI elements when generating lists
Search tab title	The title of the search tab. See: How to localize your documentation output
Show BreadCrumbs	Show or hide the breadcrumbs at the top of each topic
Show inline table of contents	Show or hide the topic's inline table of contents. See: Inline table of contents
Show navigation arrows	Show or hide the navigation arrows at the top of each topic
Show splitter bar	Show or hide the resizable bar between the table of contents and topic content
Show the keywords tab	Show or hide the keywords tab
Show the search tab	Show or hide the search tab
Show the table of contents tab	Show or hide the table of contents tab
Sitemap base URL	The URL of the generated documentation. Will be used to generate the sitemap links
Sitemap change frequency: home	How often will the home page change
Sitemap change frequency: others	How often will the topics change
Sitemap priority: home	Priority of the home page in the sitemap

Sitemap priority: others	Priority of the topics in the sitemap
Sitemap: Generate ?	Generate or not the sitemap file
Table of contents expand level	The default table of contents expansion level: 1 will expand root level topics, 2 will also expand second level topics...
Table of contents tab title	The title of the table of contents tab. See: How to localize your documentation output
Table of contents width	The width of the table of contents in pixels
Theme	The color theme to use
Topic file extension override	Override the template's topic file extension. Templates usually use this to generate individual topic filenames. Note: Use space characters for empty extensions. See: HTML Based templates' general settings
Translation for...	The translation for the specified English term

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

Inline table of contents

HelpNDoc's default HTML template supports inline table of contents for topics. This table of contents displays a list of headings named "In this topic". To activate this feature, enable the "Show inline table of contents" [custom setting](#). It is possible to customize its width in the "Inline table of content's width" [build settings](#).

To setup a topic's title of contents, the default template will look at HTML headings in the current topic (e.g. H1, H2...). To define them, you can either:

- Use styles which have an outline level defined, such as "Heading 1", "Heading 2"... an apply them to titles which should appear in the inline table of contents;
- Or set the "[Outline level](#)" [paragraph property](#) to those titles by right-clicking on them, then click "Paragraph...", go to the "Advanced" tab, and set an "Outline level" other than "Body text".

Setup Google tag manager

HelpNDoc's default HTML template supports [Google Tag Manager](#), which is used to install, store and manage marketing tags without modifying the documentation's code. It follows Google Tag Manager's best practice by adding the correct code both in the `head` and in the `body` sections of the generated HTML documentation.

Retrieve Google Tag Manager's ID

HelpNDoc will only add Google Tag Manager's code if the ID is specified for a build. Here is how to get the ID:

- Navigate to [Google Tag Manager's home page](#)
- Locate the desired container's row, select and copy the value of the "Container ID" column

Assign the ID to a HTML build

Now that we have the ID, here is how you can set it in your HelpNDoc HTML build:

- From HelpNDoc's "Home" ribbon tab, in the "Project" group, click the top part of the "Generate help" button
- Select the HTML build on the list
- Make sure that it uses the "Default HTML template"
- If the "Template settings" tab is not visible, click "Customize"
- In the "Template settings" tab, locate the "Google Tag Manager Id" row and paste the previously retrieved ID

That build is now ready to use Google Tag Manager: next time it is generated, it will automatically produce the correct code to use Google Tag Manager.

Handle the "History Change" event

As HelpNDoc's default HTML template uses various optimization techniques to speed-up the display of HTML documentation pages, it won't reload the whole page when the user clicks another topic in the table of contents, which will lead to lost Google Tag Manager PageView events. To work around that, in your Google Tag Manager's administrator console:

- Create a new "History Change" trigger. See: <https://support.google.com/tagmanager/answer/7679322>
- Access your tag's configuration, then in the "Triggering" panel add the newly created "History Change" as a "Firing Triggers"
- Click "Save"

Word documentation settings

Some of the [templates settings](#) available for Word builds are:

Field	Description
Counters as text	By default, counters are generated as Word fields. When checked, they will be generated as raw text instead
Generate RTF format	Word builds generate DocX files by default. This forces the build to generate RTF documents instead
Hide the cover page	Do not generate any cover page
Hide the table of contents	Do not generate the table of contents
Number of levels in table of contents	Number of heading levels visible in the generated table of contents
Table of contents title	The title for the "Table of contents" text. See: How to localize your documentation output

See the [How to define build settings in HelpNDoc step-by-step guide](#).

PDF documentation settings

Some of the [templates settings](#) available for PDF builds are:

Field	Description
Do not generate bookmarks	Bookmarks won't be included in the generate PDF document
Hide the cover page	Do not generate any cover page
Hide the table of contents	Do not generate the table of contents
Number of levels in table of contents	Number of heading levels visible in the generated table of contents

Table of contents title	The title for the "Table of contents" text. See: How to localize your documentation output
Use Windows Uniscribe API	The Windows Uniscribe API can produce better looking PDF for some Middle-East and Asian languages. Using it can be slower to produce PDF documents. Note: Only available when the "Use legacy PDF generator" option is off

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

ePub documentation settings

Some of the [templates settings](#) available for ePub builds are:

Field	Description
Book ID	Unique ePub book ID or ISBN
Clean output directory	Optionally clean the output directory before generation starts. Warning: this will delete any file and folder in the output directory.
Cover picture	The picture library item to use as a cover page
Export normal style	Define the normal style as the default style. Use web browser's default otherwise
Font size as percent	Use percent values for font sizes
Inline cover page	Include a cover page within the eBook content
Inline table of contents	Include a table of contents within the eBook content
Keep temporary files	Keep the temporary files needed by the compiler to build the final documentation
Numbering as text	Use text instead of OL/LI elements when generating lists

Table of contents title	The title for the "Table of contents" text. See: How to localize your documentation output
-------------------------	--

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

Kindle / Mobi documentation settings

Some of the [templates settings](#) available for Kindle / Mobi builds are:

Field	Description
Book ID	Unique ePub book ID or ISBN
Clean output directory	Optionally clean the output directory before generation starts. Warning: this will delete any file and folder in the output directory.
Compression level	The eBook compression level
Cover picture	The picture library item to use as a cover page
Export normal style	Define the normal style as the default style. Use web browser's default otherwise
Font size as percent	Use percent values for font sizes
Inline cover page	Include a cover page within the eBook content
Inline table of contents	Include a table of contents within the eBook content
Keep temporary files	Keep the temporary files needed by the compiler to build the final documentation
Numbering as text	Use text instead of OL/LI elements when generating lists
Table of contents title	The title for the "Table of contents" text. See: How to localize your documentation output

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

Qt help documentation settings

Some of the [templates settings](#) available for Qt help builds are:

Field	Description
About icon	The picture library item to use as the icon in the "About" dialog
About menu text	The text to use as the "About" menu item
About text	The text to display in the "About" dialog
Address bar enabled	Control if address bar visibility can be changed in Qt Assistant
Address bar visible	Is the address bar shown by default
Application icon	The picture library item to use as the application icon
Base color	The base color of the theme
Clean output directory	Optionally clean the output directory before generation starts. Warning: this will delete any file and folder in the output directory.
Documentation manager enabled	If disabled, the documentation manager is not shown in Qt Assistant preference dialog
Export normal style	Define the normal style as the default style. Use web browser's default otherwise
Filter enabled	Control if the filter can be changed in Qt Assistant
Filter visible	Is the filter bar shown by default
Font size as percent	Use percent values for font sizes

Full text search fallback	Use full text search if a keyword can't be found in the index
Keep temporary files	Keep the temporary files needed by the compiler to build the final documentation
Namespace	The unique documentation namespace
Numbering as text	Use text instead of OL/LI elements when generating lists
Show BreadCrumbs	Show or hide the breadcrumbs at the top of each topic
Show navigation arrow	Show or hide the navigation arrows at the top of each topic
Virtual folder	Virtual directory name in the specified namespace

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

Markdown documentation settings

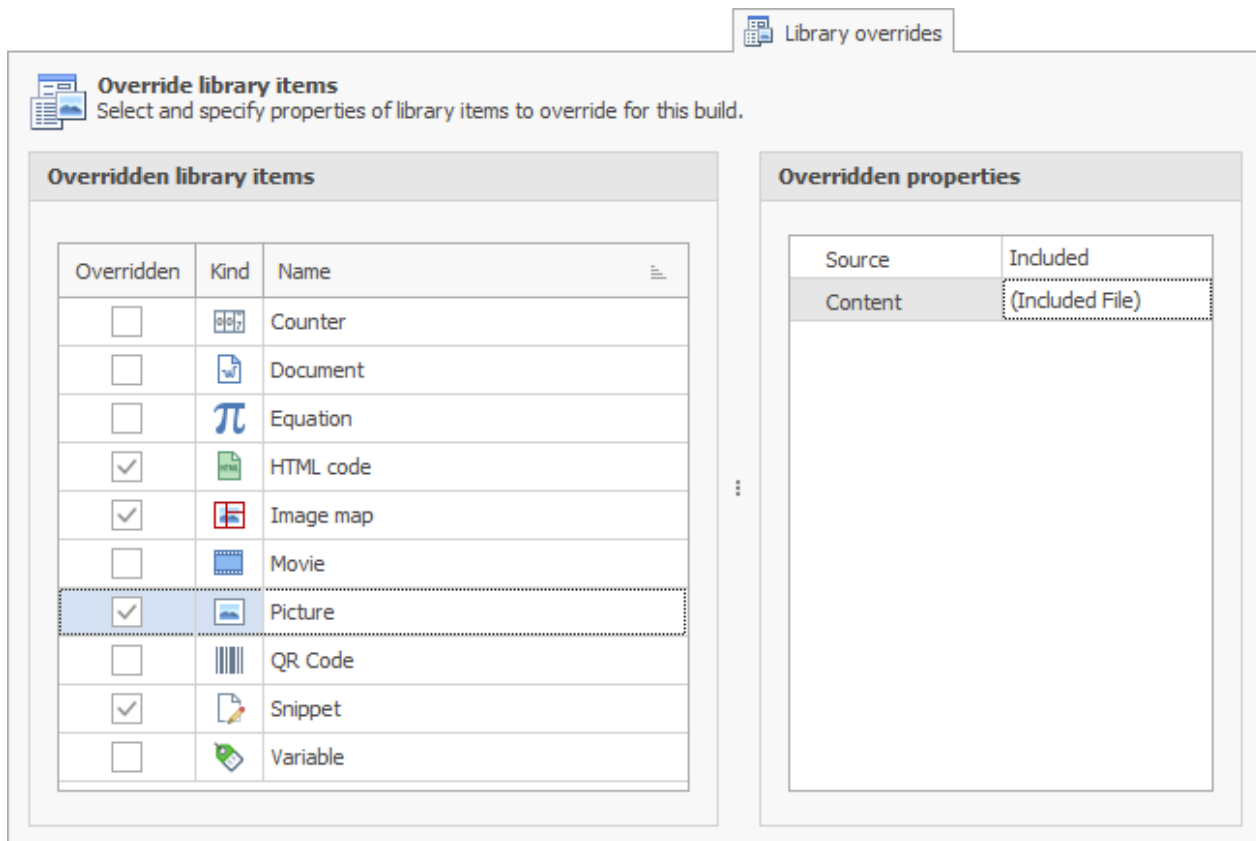
Some of the [templates settings](#) available for Markdown builds are:

Field	Description
Clean output directory	Optionally clean the output directory before generation starts. Warning: this will delete any file and folder in the output directory.
Generate compact Markdown	Generated Markdown is more compact, but less readable.
Line width	Preferred number of characters per line before wrapping. Purely cosmetic effect.
Use Setext-style headings	If enabled, headings are underlined. If not, atx-style headings with leading hash marks (#) is used.

See the [How to define build settings in HelpNDoc](#) step-by-step guide.

Override library items

Any [library item](#) created within the project can be overridden by each documentation build: every documentation output can have its own set of customized library items. Once a library item has been overridden for a specific build, the overridden values set for this library items will be used next time the build is generated.



Access the "Library overrides" panel

Each build have its own set of overridden library items. To access the "Library overrides" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Library overrides" tab

Overriding library items

To override a specific library item for the selected build, its "Overridden" column must be checked:

- Locate the library item to override in the "Library overrides" panel
- Check its "Overridden" checkbox

An overridden library item also needs overridden properties or it will be empty.

Overriding library items properties

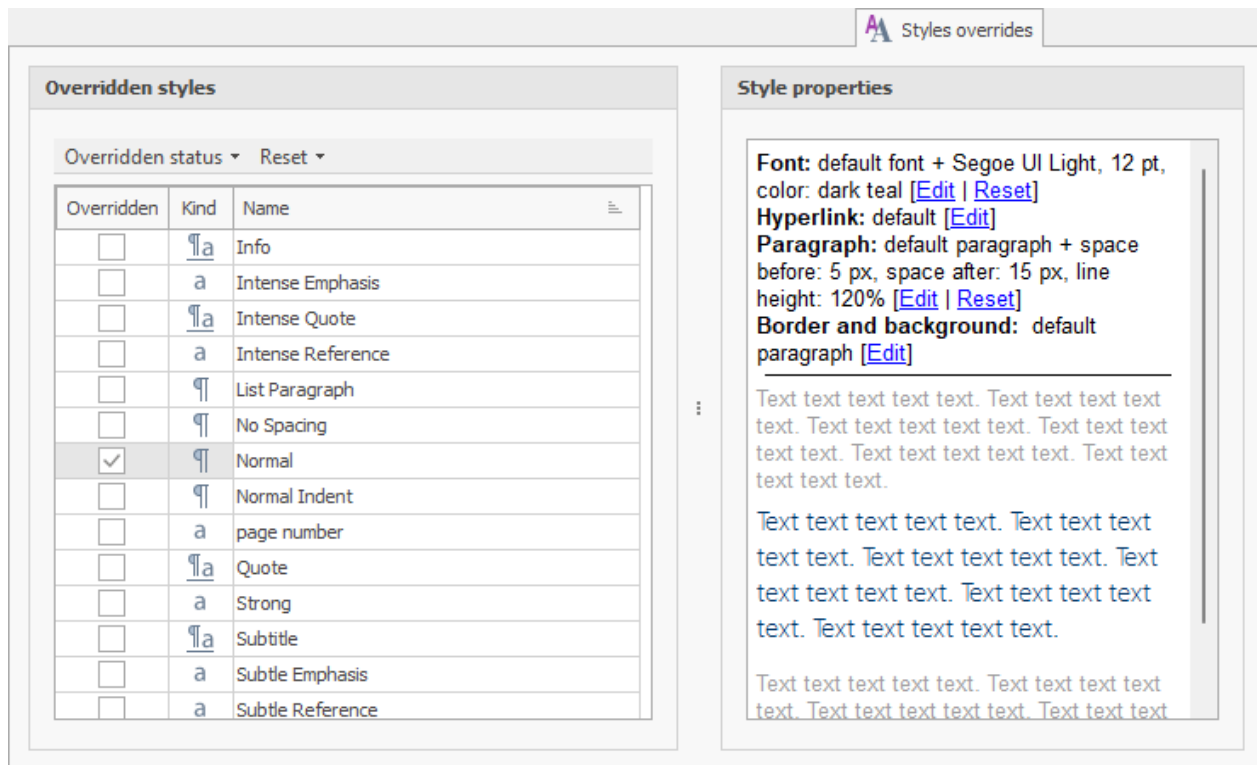
Each library items kinds have their own set of properties to override:

Item kind	Overridden properties
Barcode	<ul style="list-style-type: none"> • Barcode: Bar-code content for this item. See: Barcode editor
Counter	<ul style="list-style-type: none"> • Caption format: Define the caption format for this counter. See: Counter library item • Empty caption format: Define the empty caption format for this counter. See: Counter library item
Document	<ul style="list-style-type: none"> • Source: Source of the document's content (e.g. Included, External) • Content: Load and store the document's content from an external file, when source is set to "Included" • File path: Path of the document file to load at generation time, when source is set to "External"
Equation	<ul style="list-style-type: none"> • Equation: Define the new equation using the equation editor. See: Equation library item
HTML code	<ul style="list-style-type: none"> • Source: Source of the HTML content (e.g. Included, External) • HTML code: Actual HTML content stored within the project, when source is set to "Included" • File path: Path of the HTML file to load at generation time, when source is set to "External"
Image map	<ul style="list-style-type: none"> • Content: Image content for this image map • Image map: Image map data (e.g. shapes and links). See: Working with the image map editor
Movie	<ul style="list-style-type: none"> • Source: Source of the movie's content (e.g. Included, External, URL) • Content: Load and store the movie's content from an external file, when source is set to "Included"

	<ul style="list-style-type: none"> • File path: Path of the movie file to load at generation time, when source is set to "External" • URL: URL of the movie, when source is set to "URL" • Width: Width of the movie, in pixels • Height: Height of the movie, in pixels
Picture	<ul style="list-style-type: none"> • Source: Source of the picture's content (e.g. Included, External, URL) • Content: Load and store the picture's content from an external file, when source is set to "Included" • File path: Path of the picture file to load at generation time, when source is set to "External" • URL: URL of the picture, when source is set to "URL"
Snippet	<ul style="list-style-type: none"> • Snippet content: Specify content for that snippet. See: Snippet editor
Variable	<ul style="list-style-type: none"> • Variable value: Specify new value for that variable

Override styles

Any [style created](#) within the project can be overridden by each documentation build: every documentation output can have its own set of customized styles. Once a style has been overridden for a specific build, the overridden values set for this style will be used next time the build is generated.



Access the "Styles overrides" panel

Each build have its own set of overridden styles. To access the "Styles overrides" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Styles overrides" tab

Overriding styles

To override a specific style for the selected build, its "Overridden" column must be checked:

- Locate the style to override in the "Styles overrides" panel
- Check its "Overridden" checkbox

Changing an overridden styles' properties

Each style available in the project is listed in the "Styles overrides" panel. To change the properties of a specific style:

- Select the desired styles in the list
- In the "Style properties" panel, click the "Edit" link next to its properties to update it

Quickly change the overridden status of styles

From the "Styles overrides" panel, use the "Overridden status" popup menu to:

- Override all: Check all styles as overridden
- Override none: Un-check all styles override status
- Invert overridden status: Invert the overridden status for all styles
- Override selection: Override only the selected styles
- Insert selection's overridden status: Invert the overridden status for the selected styles

Resetting overridden styles

From the "Styles overrides" panel, use the "Reset" to reset the properties of overridden styles:

- Reset all: Reset the modified styles properties for all styles
- Reset selected: Reset the modified styles properties for the selected styles only

Sign documents

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.

Word and PDF documents can be signed using an invisible signing certificate. Viewer applications such as Microsoft Word, Adobe Reader... check that a signed document is coming from the advertised author and that it hasn't been altered in any ways since it has been signed. HelpNDoc supports CER, PEM and PFX encoded certificates.


For more specific details, see:


- [Sign Word documents](#)
- [Sign PDF documents](#)

Sign Word documents

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.

Word documents generated by HelpNDoc can be signed using an invisible signing certificate. Microsoft Word displays a message to confirm that a document has been signed by its author, and that it hasn't been altered since then. HelpNDoc supports CER, PEM and PFX encoded certificates.

 Sign document



Add a Digital Signature
 Ensure the integrity of the document by adding a digital signature

☒ Sign Document

Signing options

Signing certificate : ...

Certificate password :

Additional Options

☒ Sign Signature Origin

☒ Sign Core Properties

Commitment

Commitment Type :

Purpose for signing :

Signature Info

Signer Role/Title :

Address :

Address (2) :

City :

Zip / Postal Code :

Country :

State / Province :

Access the "Sign document" panel

Each Word build can be signed independently. To access the "Sign document" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a Word build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Sign document" tab

Sign a Word document

Check the "Sign Document" checkbox to enable document signing for this build. Available options:


Option name	Description
-------------	-------------


Signing certificate	Specify the file to use as the signing certificate. Supported formats: CER, PEM and PFX encoded certificates
Certificate password	Password for the signing certificate selected
Sign signature origin	Specifies whether to sign the XPS document's signature origin
Sign core properties	Sign core properties which are a set of elements that describe common and well-known properties of the document such as creator, version, revision...
Commitment type	Type of commitment made by the signer. E.g. "Created and approved this document"
Purpose for signing	Signer's purpose for signing this document
Signature info	Additional information about the signer

Sign PDF documents

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.

PDF documents generated by HelpNDoc can be signed using an invisible signing certificate. Adobe Reader displays a message to confirm that a document has been signed by its author, and that it hasn't been altered since then. HelpNDoc supports CER, PEM and PFX encoded certificates.

 Sign document


Add a Digital Signature
 Ensure the integrity of the document by adding a digital signature

☒ Sign Document

Signing options

Signing certificate : ...
 Certificate password :

Timestamp

☒ Request a timestamp from TSA server
 Timestamp server URL :

Signature Info

Signer Role/Title :
 Purpose for signing :

Access the "Sign document" panel

Each PDF build can be signed independently. To access the "Sign document" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a PDF build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Sign document" tab

Sign a PDF document

Check the "Sign Document" checkbox to enable document signing for this build. Available options:

Option name	Description
Signing certificate	Specify the file to use as the signing certificate. Supported formats: CER, PEM and PFX encoded certificates

Certificate password	Password for the signing certificate selected
Request a timestamp from TSA server	If checked, a request will be made to the address specified in "Timestamp server URL" to certify the signing date and time
Timestamp server URL	URL of the timestamp server to request for time-stamping
Signature Info	Additional information about the signer

Encrypt documents

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.

Word and PDF documents can be encrypted and password protected using strong encryption algorithms. It won't be possible for anyone to view the document or access any of its content without the specified password.

For more specific details, see:

- [Encrypt Word documents](#)
- [Encrypt PDF documents](#)

Encrypt Word documents

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.

Word documents generated by HelpNDoc can be encrypted using strong encryption algorithms. HelpNDoc supports the following Word encryption algorithms: RC2, RC4, DES, 3DES, AES128, AES192, AES256.



Access the "Encrypt document" panel

Each Word build can be encrypted independently. To access the "Encrypt document" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a Word build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Encrypt document" tab

Encrypt a Word document

Check the "Encrypt Document" checkbox to enable document encryption for this build. Available options:

Option name	Description
Encryption algorithm	Specify the encryption algorithm to use for document encryption
Password	Specify the password to use for document encryption

Encrypt PDF documents

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.

PDF documents generated by HelpNDoc can be encrypted using strong encryption algorithms. HelpNDoc supports the following PDF encryption algorithms: RC4 40 bits, RC4 128 bits, AES 128

bits, AES 256 bits with hardened key generation.

Access the "Encrypt document" panel

Each PDF build can be encrypted independently. To access the "Encrypt document" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a PDF build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Encrypt document" tab

Encrypt a PDF document

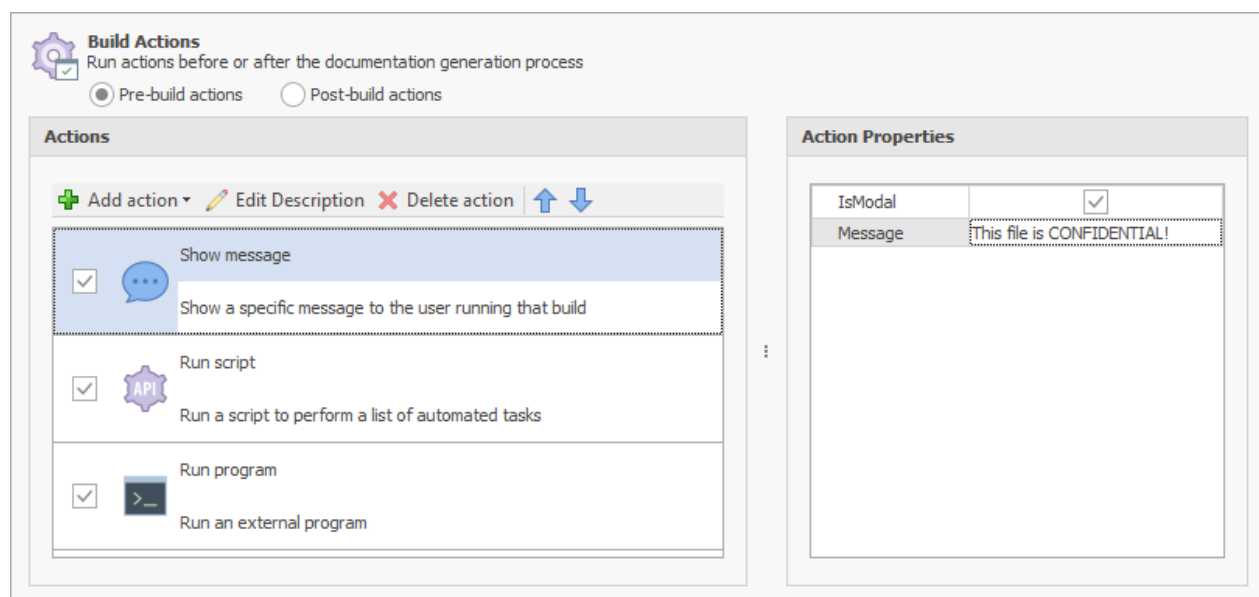
Check the "Encrypt Document" checkbox to enable document encryption for this build. Available options:

Option name	Description
-------------	-------------

Encryption algorithm	Specify the encryption algorithm to use for document encryption
Encrypt Metadata	Encrypts document metadata if checked
Owner password	Password for the document's owner. Owner has all permissions
User password	Password for the document's user. Might have limited permissions
User permissions / Printing	Specify if the end user can print the document: disabled, low resolution only, or high resolution
User permissions / Copying	Specify if the end user can copy parts of the document: disabled, text access for accessibility settings, allowed
User permissions / Modifying	Specify if the end user can modify, annotate or assemble the document

Build actions

Note: This feature might not be available on all editions of HelpNDoc. Check [HelpNDoc's feature comparison page](#) to learn more.



Each build can include an unlimited number of pre-build and post-build actions:

- Pre-build actions are executed before the generation process for that build
- Post-build actions are executed after the generation process for that build

Build actions contain the following properties:

- A name to uniquely identify the build action
- A description to optionally describe the build action
- Custom properties which are specific to each action types

Access the "Build actions" panel

Each build can have an independent list of build actions. To access the "Build actions" panel:

- From HelpNDoc's "File" menu, click the top part of the "Generate help" button to show the "Generate documentation" window
- Select a build in the build list
- Click "Customize" if the build customization tabs are not visible yet
- Navigate to the "Build actions" tab

Build action types

The following build actions can be created:

Build action type	Description
FTP(S) upload	<p>Upload generated content to a FTP(S) or SFTP server. List of available properties:</p> <ul style="list-style-type: none"> • Protocol: Protocol to use to connect to the remote server, either FTP(S) or SFTP • Server address: Domain name or IP address of the server • Server port: Port of the FTP server • Username: Username used to login to the server • Password: Password used to login to the server • Remote directory: Remote directory to upload to • Transfer more: Specify the transfer mode used for the connection (e.g. Text, Binary)

	<ul style="list-style-type: none"> • FTP(S) encryption: Determines how the FTP client starts the SSL negotiation (FTPS only) • FTP(S) passive mode: Whether or not to direct the server into a passive mode
HTTP Request	<p>Perform a HTTP request to a remote URL. List of available properties:</p> <ul style="list-style-type: none"> • URL to request: The remote URL to request • Request method: HTTP method used to perform the request (e.g. GET, POST...) • Content of the request: Request body • Content type header: Value added to the request's content-type header, specifying the type of the request's body <p>Samples:</p> <ul style="list-style-type: none"> • Call an API to display the current date time using build actions
Run program	<p>Run an external program available on the computer. List of available properties:</p> <ul style="list-style-type: none"> • File path: The path of the program to run • Arguments: Arguments to add to the program • Working directory: Full path of the current directory for the process. If not specified, it will be set to the current build's output directory • Expected return code: If provided, HelpNDoc checks the program's return value and displays an error if it is different from this value • Show window: Shows a window for GUI applications. Disable it for console applications to only capture the output without displaying a console window <p>Samples:</p> <ul style="list-style-type: none"> • Copy files using the run program action • Compress output using the run program action

Run script	<p>Run a script using the HelpNDoc API. List of available properties:</p> <ul style="list-style-type: none"> • Script path: Specify the path of the external script file to execute. • Script content: Specify the content of the script file. <p>Note: If both "Path" and "Content" fields are specified, the "Content" is used and the "Path" is ignored.</p> <p>Samples:</p> <ul style="list-style-type: none"> • Store last generation date and time using build actions
Show message	<p>Show a message during the generation process. List of available properties:</p> <ul style="list-style-type: none"> • Modal window: Specify if the message is shown as a blocking dialog box (when checked), or as a discreet message in the build generation log • Message content: Textual content of the message to display <p>Samples:</p> <ul style="list-style-type: none"> • Show a warning message using build actions

Add a build action

To add a build action, click the "Add action" button, then choose the build action type in the list. The newly created build action is added at the bottom of the existing build action list.

Edit a build action

Once a build action is selected in the list:

- Click its checkbox to enable or disable it
- Click its name, then click "Edit Name" to edit its name
- Click its description, then click "Edit Description" to edit its description
- Change its properties in the "Action Properties" panel

Delete a build action

To delete a build action, select it then click the "Delete action" button.

Move a build action

Build actions are executed in order from top to bottom:

- To execute a build action earlier, select it then click the "Move Up" button;
- To execute a build action later, select it then click the "Move Down" button.

See also: [Build actions samples](#)

Build actions samples

Here are some samples and ideas showing how to use HelpNDoc's [build actions](#) to improve the documentation generation process.

HTTP request action samples

- [Call an API to display the current date time using build actions](#)

Run program action samples

- [Copy files using the run program action](#)
- [Compress output using the run program action](#)

Run program action samples

- [Store last generation date and time using build actions](#)

Show message action samples

- [Show a warning message using build actions](#)

Call an API to display the current date time using build actions

Using the "HTTP Request" [Build action](#), it is possible to call a remote API and get the current date time. The result will be displayed in the generation log and could be useful for future reference:

Action property	Value	Description
URL to request	<code>http://worldtimeapi.org/api/timezone/Europe/Paris.txt</code>	Use the WorldTimeAPI public web service to request the current local time for a given timezone as text
Request method	GET	Perform a GET HTTP request

Copy files using the run program action

It could be useful to copy files either before or after the generation process. To achieve this, the `copy` command line can be used to copy files using the "Run program" [build action](#). As it needs to be run from a command prompt, the `cmd.exe` program needs to be run with the `/C` arguments:

Action property	Value	Description
File path	<code>cmd.exe</code>	Run a command prompt
Arguments	<code>/C copy "SOURCE_PATH" "DESTINATION_PATH"</code>	<p>The <code>/C</code> argument carries out the <code>copy</code> command and terminates.</p> <p><code>SOURCE_PATH</code> is the source file's path. It can be absolute, or relative to the <code>WorkingDirectory</code> property.</p> <p><code>DESTINATION_PATH</code> is the source file's path. It can be absolute, or relative to the <code>WorkingDirectory</code> property.</p>
Working directory	<code>WORKING_DIRECTORY_PATH</code>	Full path of the current directory for the process. If not specified, it will be set to the current build's output directory.

Compress output using the run program action

Once the documentation has been generated, it could be useful to compress it for backup or faster transfer purposes. To achieve this, the `zip` program can be run using the "Run program" [build action](#):

Action property	Value	Description
File path	<code>zip</code>	Run the zip compression command

Arguments	<code>-r backup.zip *</code>	Recursive compression of all files in the current directory to the <code>backup.zip</code> archive
-----------	------------------------------	--

Show a warning message using build actions

Once the documentation has been generated, it could be useful to display a warning message to remind some important information about the generated documentation. To achieve this, the "Show Message" [Build action](#) can be used to display a modal window:

Action property	Value	Description
Modal window	Checked	This displays a modal window showing the message: it has to be closed by clicking on a button
Message content	Content of the message	Specify the content of the message to display

Store last generation date and time using build actions

Once the documentation generation is complete, it could be useful to store the latest generation date and time somewhere in the project. It could be done using a "Run script" [Build action](#) which executes a script using HelpNDoc's [API methods](#) to store the last generation date and time in the project topic's [custom properties](#):

Action property	Value	Description
Script content	<pre>begin var aProjectTopicId := HndTopics.GetProjectTopic(); HndTopicsProperties.SetTopicCustomPropertyValue(aProjectTopicId, 'Last HTML Generation', FormatDateTime('yyyy-mm-dd hh:nn:ss', Now)); end.</pre>	Custom script to set the project topic's custom property to the current date and time.

Advanced usages

Some more advanced usages are covered in the following sections:

- [Keyboard shortcuts](#) - Various keyboard shortcuts available in HelpNDoc
- [Conditional content generation](#) - How to generate multiple versions of your documentation
- [Analyzing a project](#) - Use the project analyzer to help spot problems
- [Working with templates](#) - Understand the powerful templates and learn how to customize the output of your documentation
- [Usage from the command line](#) - Learn how you can leverage the HelpNDoc command line parameters to automate documentation generation
- [CHM files and programming languages](#) - How to integrate your CHM help files with some programming languages
- [Customize default project styles](#) - How to define a default set of styles for your projects
- [Using the Script Editor](#) - Leverage HelpNDoc's API to automate help file creation
- [License key management](#) - How to register your purchased software

Keyboard shortcuts

HelpNDoc implements various keyboard shortcuts which can be used throughout the application to rapidly execute common actions.

User interface

Keyboard shortcuts available in HelpNDoc's main window.

Keyboard Shortcut	Action	Remarks
F1	Display the program help	
ALT	Show the keyboard shortcut for the ribbon elements	Press and release the shortcuts indicated to go to next step. As an example ALT, then H, then O will show the project options
CTRL + F1	Minimize / restore the ribbon	

CTRL + F2	Focus the table of contents panel	
CTRL + F3	Focus the topic editor panel	
CTRL + F4	Focus the library panel	
CTRL + F5	Focus the keywords panel	
CTRL + F6	Focus the search result panel	
CTRL + F7	Focus the topic properties panel	

Tree controls

Keyboard shortcuts available for all tree controls, including the table of contents tree, the library tree, the keywords tree.

Keyboard Shortcut	Action	Remarks
+	Expand the current node	
*	Expand the current node and its children hierarchy	
-	Collapse the current node	
/	Collapse the current node and its children hierarchy	
CTRL + F	Show / Hide the find panel	Used to filter items
CTRL + UP	Move the element up	Not available in the library and keywords trees
CTRL +	Move the element down	Not available in the library and keywords trees

DOWN		
CTRL + LEFT	Move the element left	Not available in the library tree
CTRL + RIGHT	Move the element right	Not available in the library tree
CTRL + INSERT	Create a new item	Not available in the library tree
CTRL + SHIFT + INSERT	Create a new child item	Not available in the library tree
CTRL + DEL	Delete the item	
SPACE	Associate the keyword with the topic	Only for the keywords tree
...	Search: start typing the beginning of a node to select it	Use CTRL+UP and CTRL+DOWN to move to the next found item

Topic Editor

Keyboard shortcuts available when editing a topic.

Keyboard Shortcut	Action	Remarks
CTRL + UP	Move cursor to the beginning of current / previous paragraph	
CTRL + DOWN	Move cursor to the beginning of next paragraph	
CTRL + A	Select all the content	

CTRL + C	Copy the selected content	
CTRL + V	Paste the content	
CTRL + SHIFT + F	Find and replace text in current topic or entire project	
CTRL + L	Create / Edit hyperlink	Select some text to rapidly create an hyperlink or nothing to create a new blank one
CTRL + SPACE	Display the auto-completion dialog	Used to quickly create hyperlinks, insert library items...
CTRL + SHIFT + SPACE	Insert non-breaking space	
CTRL + SHIFT + N	Apply the Normal style to selection	
ALT + SHIFT + LEFT	Apply the previous heading level style to selection	
ALT + SHIFT + RIGHT	Apply the next heading level style to selection	
CTRL + ALT + 1...3	Apply the heading level 1 to 3 style to selection	
CTRL + +	Zoom in	
CTRL + -	Zoom out	
CTRL + 0	Reset zoom to 100%	
TAB	Increase paragraph or bullet indent level	

SHIFT + TAB	Decrease bullet indent level	
-------------	------------------------------	--

Equation editor

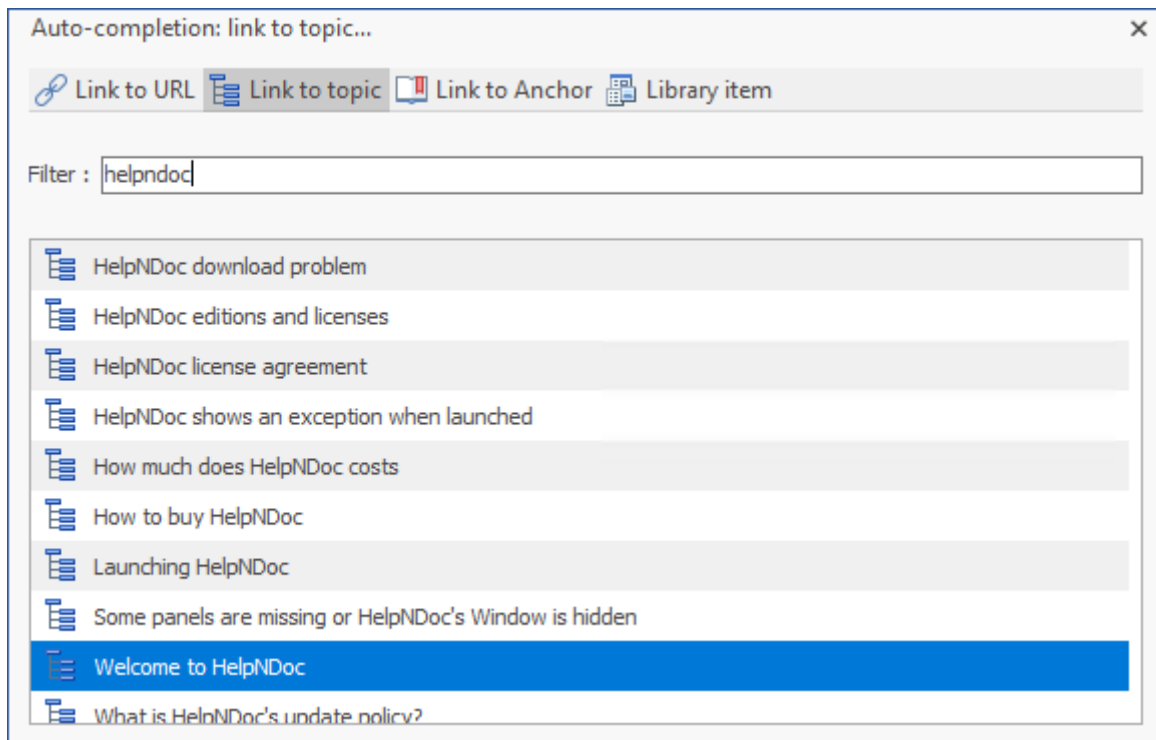
Keyboard shortcuts available when editing [equations](#).

Keyboard Shortcut	Action	Remarks
LEFT	Move cursor backward	
RIGHT	Move cursor forward	
UP	Move cursor up	
DOWN	Move cursor down	
SHIFT + LEFT	Extend selection backward	
SHIFT + RIGHT	Extend selection forward	
SHIFT + UP	Extend selection upward	
SHIFT + DOWN	Extend selection downward	
DEL	Delete backward	
ALT + DEL	Delete forward	
CTRL + A	Select all	
CTRL + Z	Undo	
CTRL + Y	Redo	

SPACE	Move after parent	
SHIFT + SPACE	Move before pare	
CTRL + 5	Move to opposite	superscript/subscript upper/lower
/	Fraction	
CTRL + 2 ALT + V sqrt	Square root	
ALT + P pi	Pi	

Keyboard auto-completion

When writing documentation, the user interface might become a distraction and could slow down the writing process, in particular when adding special and non-textual elements in the topic editor. That's why HelpNDoc provides the `CTRL+SPACE` auto-completion keyboard shortcut to speed up the writing process.



To show the auto-completion dialog, hit the `CTRL+SPACE` keyboard shortcut in the topic editor. The following actions are then available:

- Enter any text in the filter field to filter the list and show only relevant items
- Use the Up and Down keyboard keys to select the desired item
- Use the Enter keyboard key (or click the item) to insert that item
- Use the `CTRL-SPACE` keyboard keys to switch to the next auto-completion mode
- Use the `CTRL-SHIFT-SPACE` keyboard keys to switch to the previous auto-completion mode
- Use the Escape keyboard key (or click the close button) to discard the dialog

The auto-completion dialog is context sensitive. It's initial action depends on the content at position of the cursor in the topic editor.

When open, use the buttons or the `CTRL- (SHIFT-) SPACE` keyboard shortcut to switch to another mode:

Link to a URL (Internet address)

- When used after the "http://", "https://" or "ftp://" characters, it will provide a way to continue to input a URL
- After entering the complete URL, it will be inserted as a clickable link in the topic editor

Link to a topic

- When used after a space character or at the start of a new line, it will provide a list of all available topics to link to
- When used within or right after a word, it will use that word as the filter. E.g. typing "help" then CTRL+SPACE will filter all topics containing the text "help"
- After selecting the topic in the list, a link with its caption will be inserted in the topic editor

Link to an anchor in the current topic

- When used right after the "#" character, it will display a list of all available anchors in the current topic
- When used within or right after a word starting with the "#" character, it will use that word as a filter for the anchor name
- After selecting an anchor in the list, a link to this anchor will be inserted in the topic editor

Insert a library item

- When used right after the "!" character, it will display a list of all available library items
- When used within or right after a word starting with the "!" character, it will use that word as a filter for the library item
- After selecting a library item in the list, it will be inserted in the topic editor

Customizing keyboard shortcuts

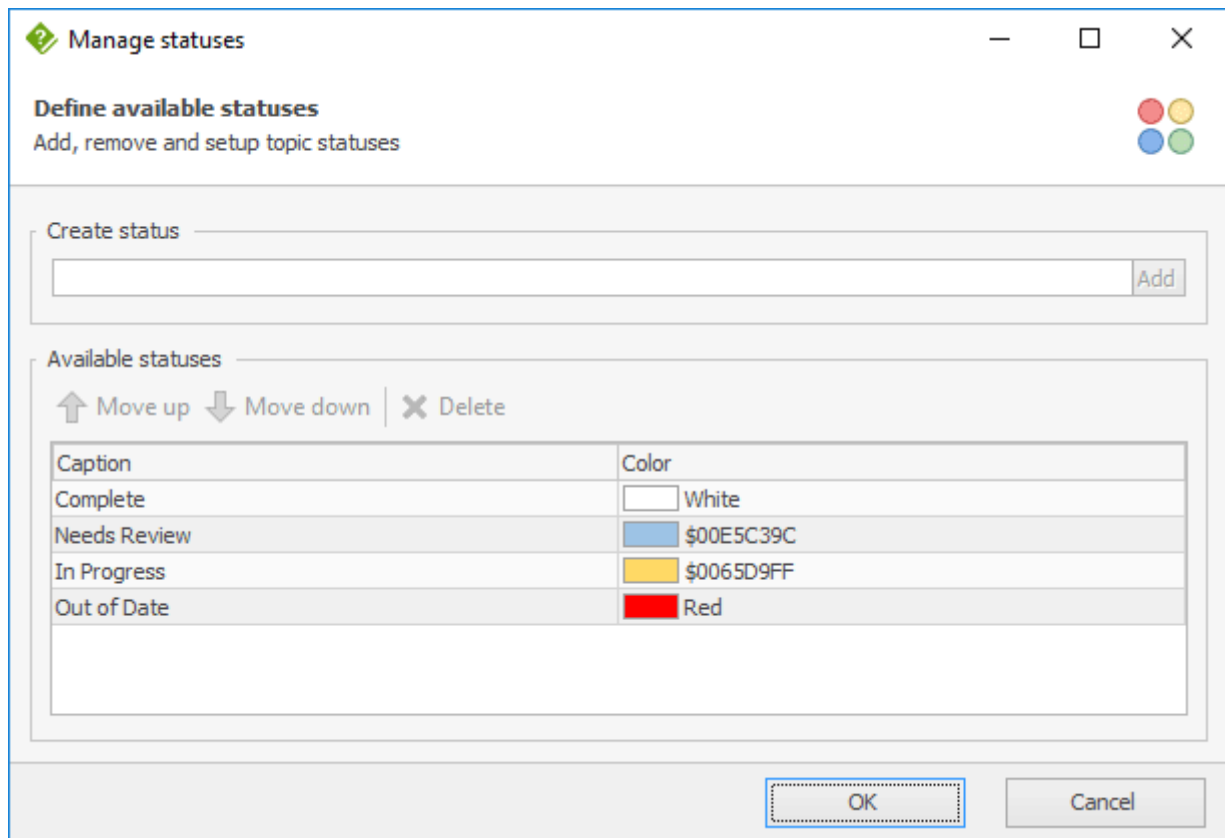
Starting with HelpNDoc 6.7, it is possible to customize some keyboard shortcuts. See the [options window](#) topic to learn how this can be done.

Topic status

Each topic can have a status, representing its state at the current time. By default, available statuses are:

- **Complete** - the topic is considered done and no further editing is necessary
- **Needs Review** - the topic is done but might need corrections
- **In Progress** - the topic is actively being worked on
- **Out of Date** - the topic's content is not up to date and should be reworked in the future

Manage statuses



It is possible to manage status (add, edit or delete them):

1. Right click on a topic
2. Hover over "Status"
3. Click "Manage Statuses..."

Conditional generation based on status

Each build can include topic with a set of statuses. By default, every topics are included in a build, but each build can be customized to only include a subset of available statuses. For example: only generate "Complete" topics.

Conditional content generation

By default, all topics and content created in an HelpNDoc project will be generated in every builds and documentation formats. It is possible to conditionally generate topics and content using build tags and conditions:

- Build tags represent unique identifiers which can be associated with a topic or a part of a topic
- Conditions are instructions indicating whether a section is included or not based on specific tags

Conditional topic generation

A topic can be included or not in specific builds based on build kind and build tags. By default a topic is included in all builds. To choose which build will include a topic, select the topic in the table of contents then:

- From the "Home" ribbon tab, click "Topic properties", then "Include in builds" and select each build kind and custom tag that applies
- Or right click on the topic and choose options in the "Include in builds" popup menu

See the [How to setup conditional topic generation](#) step-by-step guide.

Conditional content generation

HelpNDoc provides an easy way to define sections (parts of topics) which will only be included in specific builds using conditional sections. Those logical statements (If, Else, End) can be inserted within a topic using the "Insert" ribbon tab.

Using the "Insert conditional operation" dialog box, choose between one of the operations:

- IF: Start of a conditional section. The content written after this operation will be included only if the tags are included (IF) or not included (IF NOT) in the current build;
- ELSE: Will negate any previous IF operation. As an example, if the previous IF operation included "CHM and HTML", the ELSE operation will included everything but those;
- END: Will close the open conditional sections. Any content written after an END statement will be included in every builds without any condition.

Note: Conditional tags can't be nested.

See the [How to setup conditional content generation](#) step-by-step guide.

Define custom tags

To define custom tags, you can either:

- Right click on a topic, hover "Included in builds" then click "Manage build tags"
- From the "Generate help" window, select a build and click "customize" if the "Included tags" is not visible" then click "Manage tags"

See the [How to manage build tags](#) step-by-step guide.

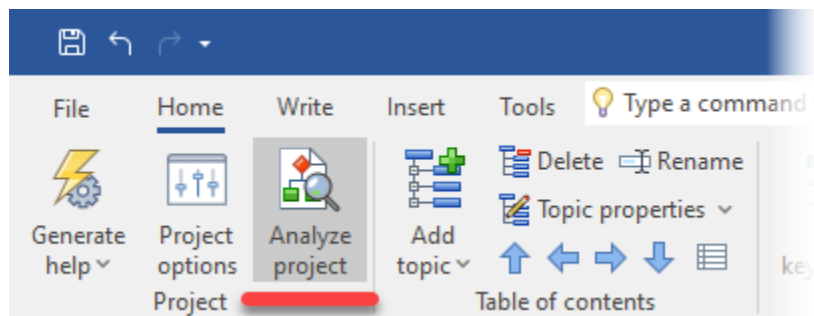
Associate tags with builds

Each build can include one or more custom tag. To add custom tags to a specific build:

- Click the top part of the "Generate help" button in the "Home" ribbon tab
- Select a build
- Click "Customize" if the "Included tags" tab is not already visible
- Select the "Included tags" tab
- Check any custom tag that will be included in that build

See the [How to maintain tags associated with output builds](#) step-by-step guide.

Analyzing a project



Analyzing an HelpNDoc project provides a centralized way to obtain advanced details on the project structure and content. The project analyzer can be launched from the "Home" ribbon tab, by clicking the "Analyze project" button in the "Project" group.

The project analyzer can be used to get various information about the project, including:

- Paragraph, character, hyper-links and library item statistics. See: [General information](#)
- A chart representing a visual overview of the project layout. See: [Visualize the project's structure](#)
- Hyper-links details and usage. See: [Analyzing hyperlinks](#)
- Anchors details and usage. See: [Analyzing anchors](#)
- Library items usage and details. See: [Analyzing library items](#)
- Keywords usage and association details. See: [Analyzing keywords](#)
- Conditional items added throughout the project. See: [Analyzing conditions](#)
- Spelling mistakes. See: [Analyzing spelling](#)

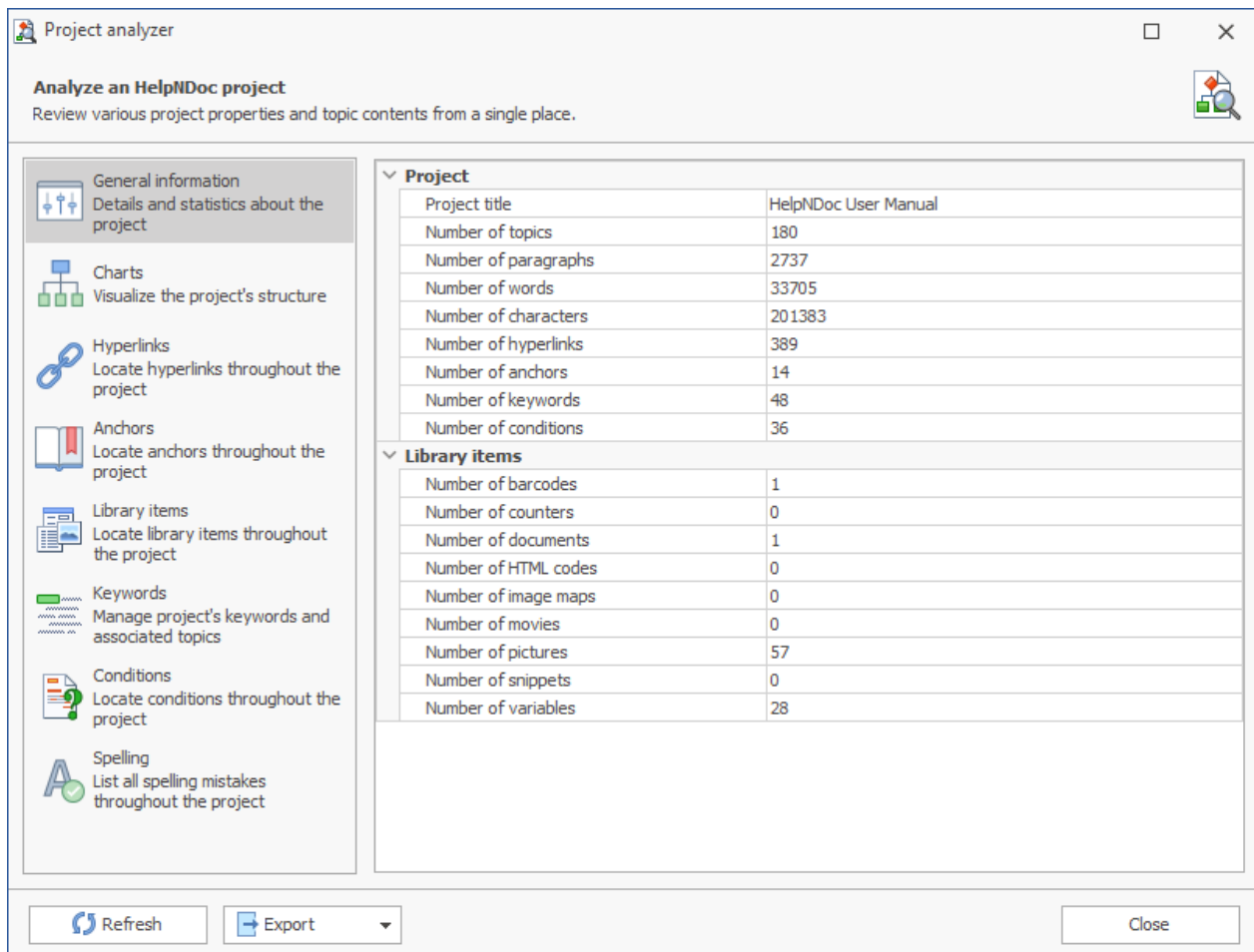
The project analyzer can be started by clicking the "Analyze" button: it may take some time as it will analyze each topic to report useful information. When the project analyzer window is visible, it

is still possible to modify the project in the background. However, any modification made to the project will not update the analyzer reports: hitting the "Refresh" button is required in that case.

Learn more about the project analyzer:

- [General information](#)
- [Visualize project structure using charts](#)
- [Analyzing hyperlinks](#)
- [Analyzing anchors](#)
- [Analyzing library items](#)
- [Analyzing keywords](#)
- [Analyzing conditions](#)
- [Analyzing spelling](#)

General information



Project analyzer

Analyze an HelpNDoc project
Review various project properties and topic contents from a single place.

General information
Details and statistics about the project

Charts
Visualize the project's structure

Hyperlinks
Locate hyperlinks throughout the project

Anchors
Locate anchors throughout the project

Library items
Locate library items throughout the project

Keywords
Manage project's keywords and associated topics

Conditions
Locate conditions throughout the project

Spelling
List all spelling mistakes throughout the project

Project

Project title	HelpNDoc User Manual
Number of topics	180
Number of paragraphs	2737
Number of words	33705
Number of characters	201383
Number of hyperlinks	389
Number of anchors	14
Number of keywords	48
Number of conditions	36

Library items

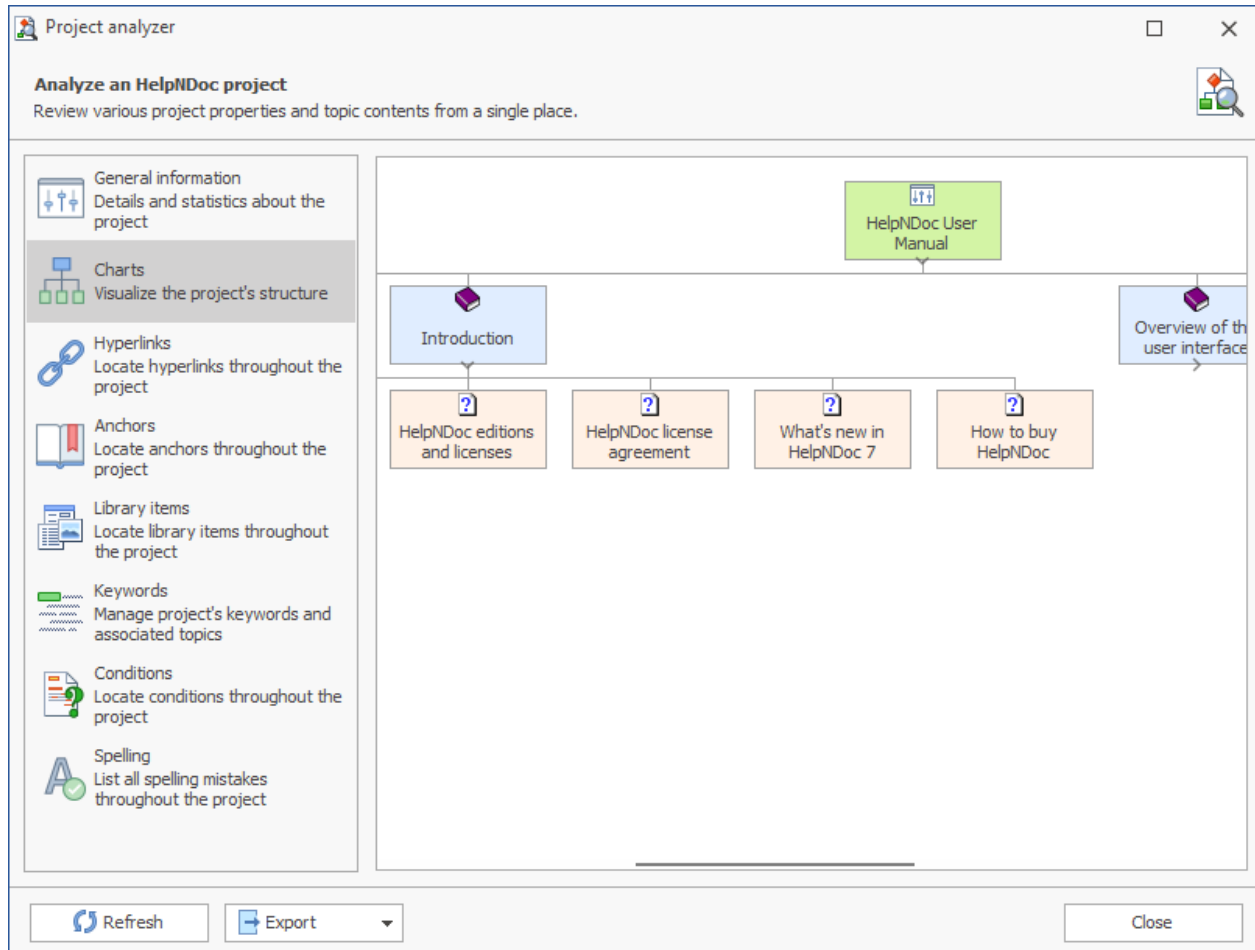
Number of barcodes	1
Number of counters	0
Number of documents	1
Number of HTML codes	0
Number of image maps	0
Number of movies	0
Number of pictures	57
Number of snippets	0
Number of variables	28

Refresh **Export** **Close**

The project analyzer's "General Information" section provides details and statistics about elements in the currently opened project:

- Number of topics and keywords
- Number of paragraphs, words and characters
- Number of hyperlinks and anchors
- Number of library items by type

Visualize the project's structure



The project analyzer's "Charts" section provides an interactive visual representation of the project's structure (or table of contents). Clicking an item in the chart selects it in the table of contents and vice-versa.

Analyzing hyperlinks

Analyze an HelpNDoc project
Review various project properties and topic contents from a single place.

Locate and select | Filter ▼

Drag a column header here to group by that column

Kind	Caption	Action	Extra	Topic
Topic	Quick Start Guides	Quick start guides		Getting started with HelpN
Topic	What's new in HelpNDoc	What's new in HelpNDoc :		Getting started with HelpN
Topic	Quick Start Guides	Quick start guides		Getting started with HelpN
Topic	Introduction	Introduction		Getting started with HelpN
Topic	How to buy HelpNDoc	How to buy HelpNDoc		Introduction
Topic	What's new in HelpNDoc	What's new in HelpNDoc :		Introduction
Topic	HelpNDoc license agreem	HelpNDoc license agreem		Introduction
Topic	HelpNDoc editions	HelpNDoc editions and lic		Introduction
Topic	Getting help	Getting help		Introduction
Topic	System requirements	System requirements		Introduction
Topic	About HelpNDoc	About HelpNDoc		Introduction
Topic	advanced functionalities	Advanced usages		About HelpNDoc
Topic	Compile	Writing documentation		About HelpNDoc
Internet/E-Mail	Qt Framework	http://www.qt.io/downlo		System requirements
Internet/E-Mail	Amazon KindleGen	https://www.helpndoc.cc		System requirements
Internet/E-Mail	Microsoft HTML Help Wor	https://www.helpndoc.cc		System requirements
Internet/E-Mail	https://www.helpndoc.cc	https://www.helpndoc.cc	Target: blan	Getting help
Internet/E-Mail	539153B41342472380D5	https://www.helpndoc.cc	Target: blan	Getting help
Internet/E-Mail	https://www.helpndoc.cc	https://www.helpndoc.cc	Target: blan	Getting help

Refresh | Export ▼ | Close

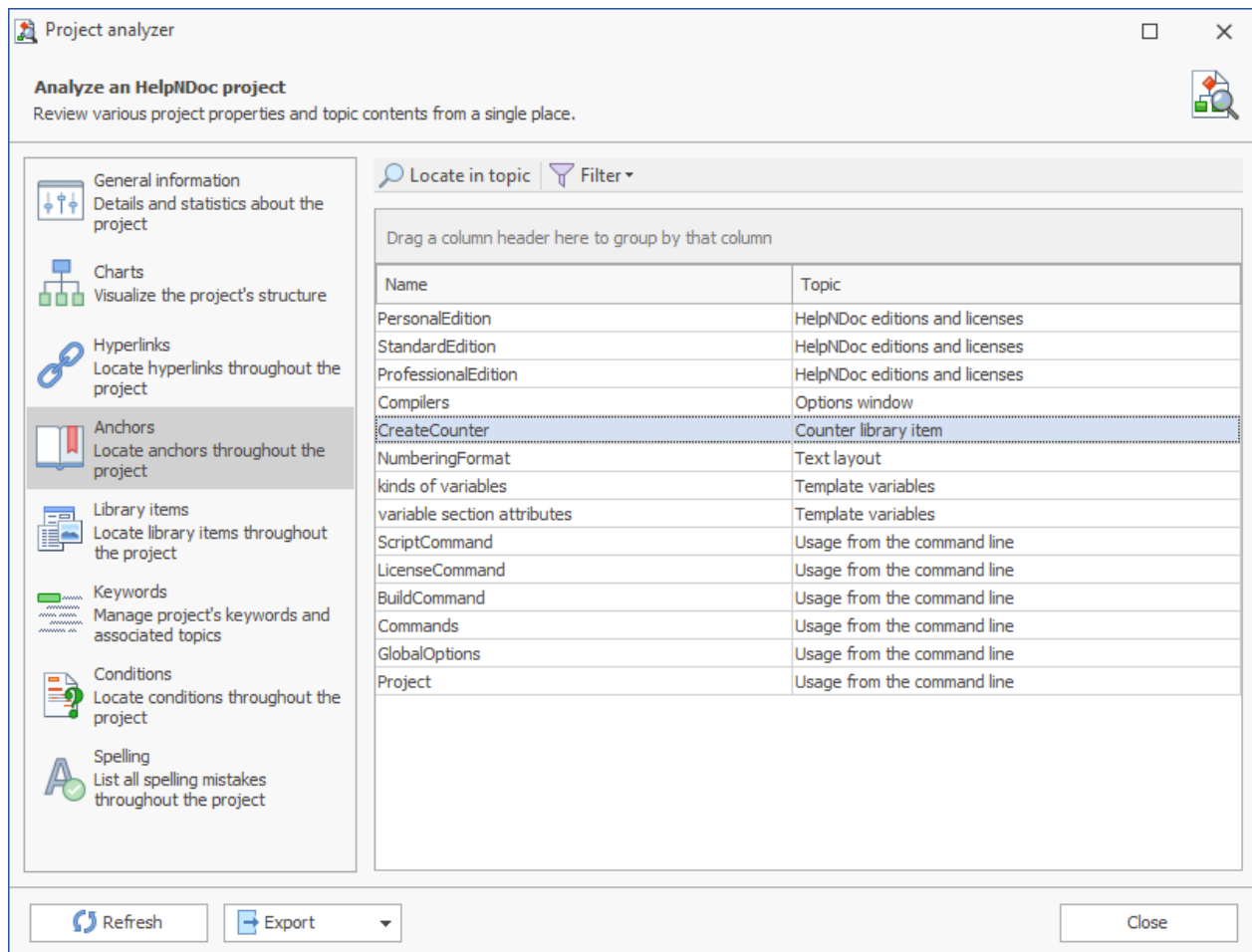
The "Hyperlinks" section of HelpNDoc's project analyzer lists all hyperlinks found throughout the project. The hyperlink analyzer can be used to:

- List all links used in the project and their properties
- Locate a link within the project
- Spot links pointing to deleted topic (broken links)
- Spot duplicate links
- Filter links by kind, caption, target or topic

Filtering broken links

Broken links point to a specific topic which has been deleted since the link creation. HelpNDoc's project analyzer makes it easy to spot broken links by clicking "Filter" then "Show broken items only". This will filter the view to display only broken links. By double-clicking a link in the view, or selecting it then clicking "Locate and select", the topic containing the link will be shown in HelpNDoc's main window so that the link can easily be corrected.

Analyzing anchors



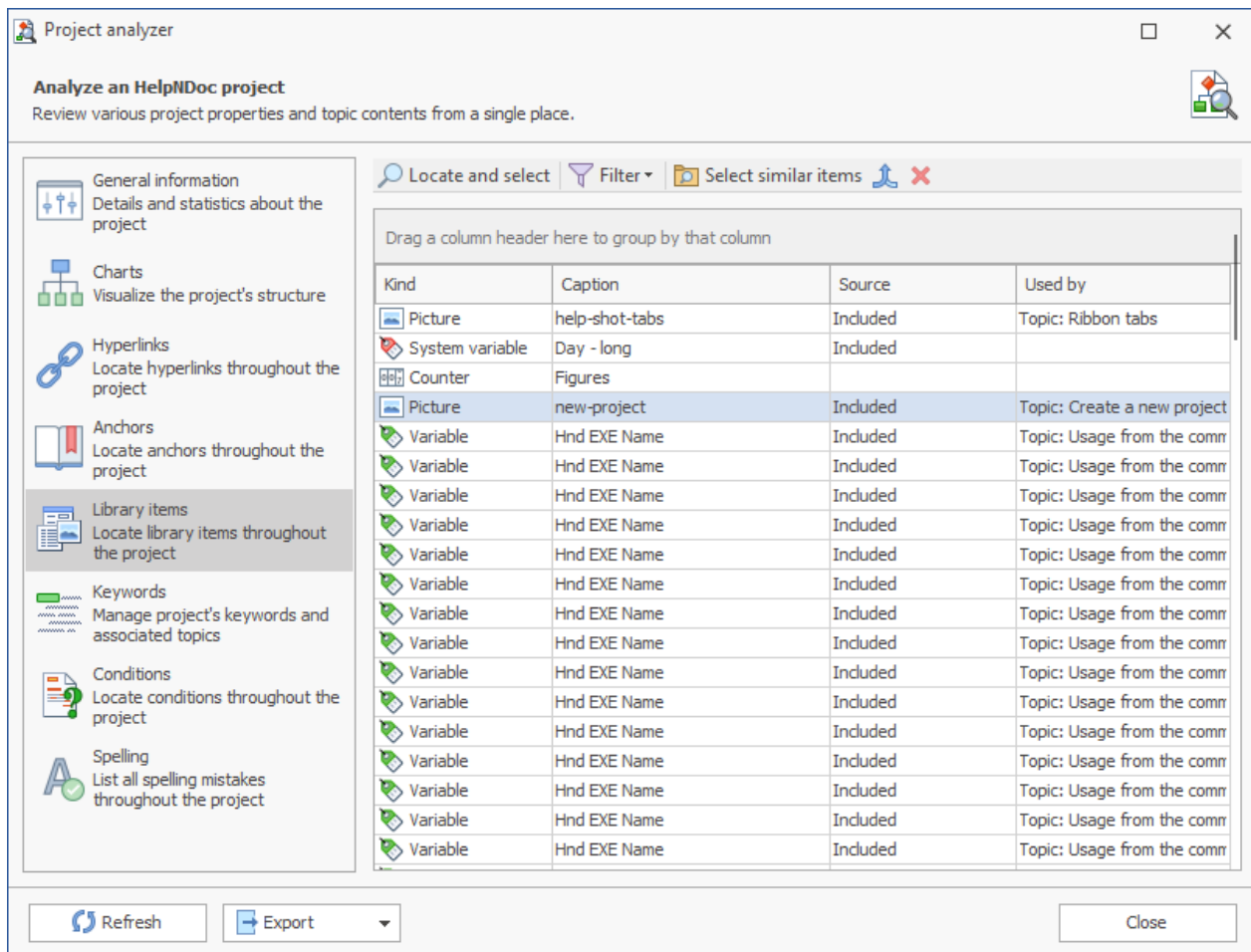
The "Anchors" section of HelpNDoc's project analyzer lists all anchors found throughout the project.

Actions

HelpNDoc can show a specific anchor's position within the project. To locate an anchor, either:

- Select it in the list then click "Locate in topic"
- Double click on it

Analyzing library items



The "Library items" section of HelpNDoc's project analyzer lists all library items present in the library, and how they are used throughout the project: a library item can be displayed multiple times in the list if it is used multiple times in the project. The library items view is very powerful to manage library items and can help save time by:

- Showing how many times and where each library item is being used
- Filtering library items by kind
- Filtering library items which are not used at all
- Filtering library items which are included in topics but not available in the library anymore (broken items)
- Locating library items with the exact same content
- Merging multiple library items into one final item

Filtering unused library items

By clicking the "Filter" button then "Show un-used items only", HelpNDoc will display library items which are available in the library but not used in any topic. This is useful to purge the project and

clean the library.

Filtering broken library items

Broken library items are items which have been added to topics at some point, but which are not available in the library anymore: broken library items can result in a broken documentation with missing parts. To view broken library items, click the "Filter" button then "Show broken items only". Double-clicking an item, or selecting it then clicking "Locate and select" will show the topic containing it so that it can be deleted or replaced.

Finding similar items

If you suspect some items are duplicates within the library, this feature will check all of them and select the duplicates. Here is how to proceed:

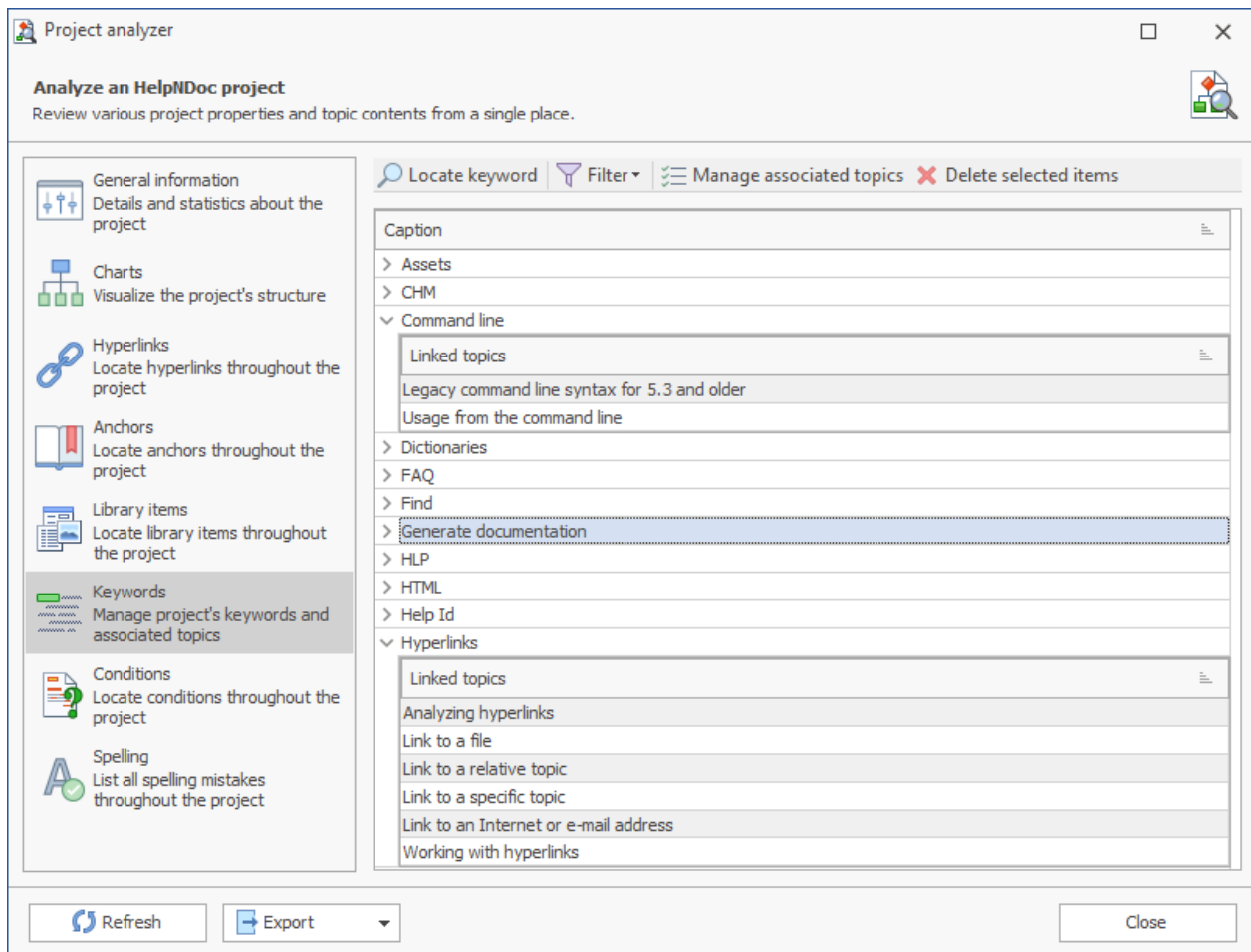
- Select an item in the view
- Click the "Select similar items" button
- HelpNDoc will automatically select all items of the same kind and with the same content

Merging library items

Select multiple library items and click "Merge selected items" to merge them into a final one. This will:

- Delete all selected items from the library except the final one
- Replace all items in the topics by the final one

Analyzing keywords



The "Keywords" section of the project analyzer lists all keywords available in the current project with a list of associated topics. The keywords analyzer can be used to:

- Get a representation of keyword usage within the project: spot rarely used or overused keywords easily
- Spot broken keywords which are not associated with any topic
- Manage topics associated with a specific keyword using the [Manage keyword association](#) window.

Filtering unused keywords

To filter the list and see only keywords with no associated topics, click "Filter" then "Show un-used items only".

It is possible to delete those keywords by selecting them and hitting the delete keyboard shortcut, or clicking the "Delete selected items" button. **Warning:** it will delete their children keywords too.

Manage associated topics

When a keyword is selected, click the "Manage associated topics" button to show the window and check each topic which needs to be associated with that keyword.

Analyzing conditions

Project analyzer

Analyze an HelpNDoc project
Review various project properties and topic contents from a single place.

Locate and select | Filter ▾

Drag a column header here to group by that column

Operation	Condition	Used by
END		Getting started with HelpNDoc
ELSE		Getting started with HelpNDoc
IF	CHM,HTML	Getting started with HelpNDoc
END		Getting help
ELSE		Getting help
IF	CHM,HTML	Getting help
END		What's new in HelpNDoc 7
ELSE		What's new in HelpNDoc 7
IF	CHM,HTML	What's new in HelpNDoc 7
END		Topic status
ELSE		Topic status
IF NOT	CHM,HTML	Topic status
END		Using the integrated web server
ELSE		Using the integrated web server
IF	PDF,WORD	Using the integrated web server
END		Backing up projects
ELSE		Backing up projects
IF NOT	CHM,HTML	Backing up projects
END		Microsoft Access integration
ELSE		Microsoft Access integration

Refresh | Export ▾ | Close

The "Conditions" section of HelpNDoc's project analyzer lists all [conditional items](#) throughout the project: IF, IF NOT, ELSE and END. The conditions analyzer can be used to:

- List all conditions used within a project
- Locate and select conditions within topics
- Filter items by operations, condition or topic

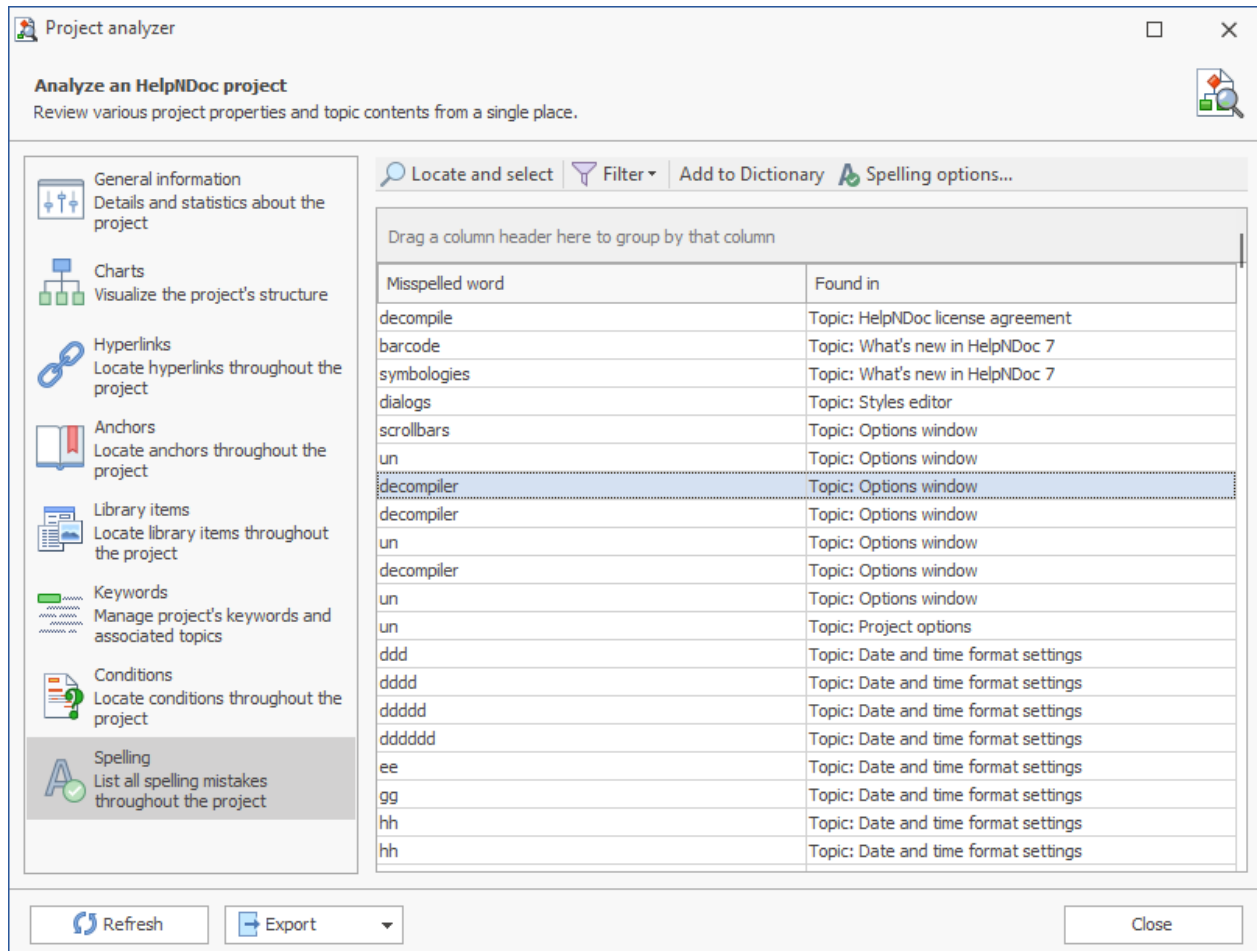
Filters

By clicking the "Filter" button, it is possible to filter by a specific operation or clear the filter. Alternatively, it is possible to place the mouse over the grid header to show the column's filter icon: clicking that icon provides additional filter capabilities for that column.

Actions

When a row is selected, clicking the "Locate and select" button will select the topic, and conditional operation within that topic in HelpNDoc's topic editor.

Analyzing spelling



The "Spelling" section of HelpNDoc's project analyzer lists all spelling mistakes throughout the project. It will analyze topics contents as well as snippets placed in the library.

Filters

Click "Filter" to display a list of available filters:

- Show snippets only: will only show misspelled words in snippets
- Show topics only: will only show misspelled words in topics
- Clear filter: will remove any filter

Actions

To locate and select the problematic word within the project, click the "Locate and select" button.

Click "Add to Dictionary" to add the currently selected word to the user dictionary and "Spelling options" to manage project-wide spelling options.

Vacuuming a project

The HND project file format is based on the SQLite database format which includes various optimizations to speed-up disk reading and writing operations:

- When a large amount of data is deleted from the project (such as library items), it leaves behind empty space. This means the project file might be larger than needed;
- Frequent modifications (such as inserts, updates, and deletes) can cause the project file to become fragmented. This means that project operations can be slower than usual and the project file might be larger than needed;

Using the "Vacuum Project" command from HelpNDoc's "Tools" ribbon tab will optimize the currently opened HND project file by rebuilding it and repack its content into a minimal amount of disk space. This leads to smaller and faster HND project files.

Vacuuming HelpNDoc projects should be done from time to time.

Working with templates

HelpNDoc includes two very powerful template systems which are used to fine-tune the look of the generated documentations: HelpNDoc will read the selected template's instructions prior to generating the documentation, and will adapt the generated output based on those instructions.

The two parts of the template system are:

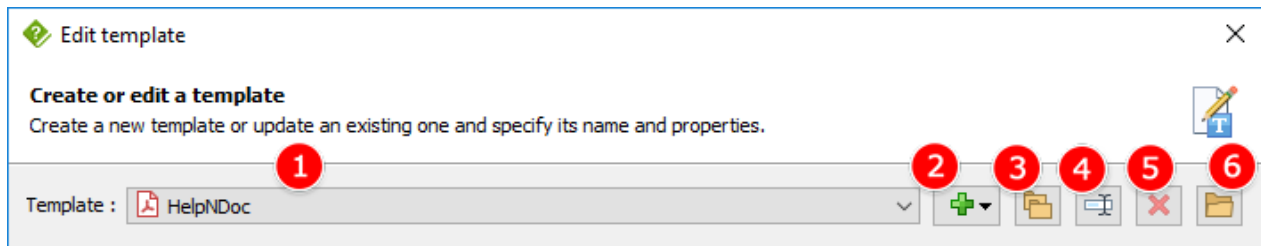
- **The CHM, HTML, ePub, Kindle, Markdown and Code template system** which can control almost all aspects of the documentation generation for those formats. It is based on the Pascal programming language which is interpreted to define how the documentation is generated;
- **The Word and PDF template system** which can control the page size, covers, headers and footers, layout and headings appearances.

Using the [template editor](#) is the recommended way to manage and customize templates. It is also possible to manually alter the template files: read the [low-level details](#) about templates for such cases.

Using the template editor

The easiest way to manage templates is by using the template editor. The template editor can be used to create, rename, modify and delete all kinds of templates.

It can be accessed using the "Template Editor" button from the "Templates" group in HelpNDoc's "Tools" ribbon tab.



Managing templates

The template editor is used to manage all kinds of templates:

1. The template selection lists all templates available on the current computer. To manage a specific template, select it in that list.
Standard templates (included with HelpNDoc's installation) are marked as such: they can't be edited;
2. The create template button can be used to create a new template of any kind: select the template kind and enter a unique name to create that template;
3. The duplicate template button can be used to duplicate the currently selected template. This can be useful to create a small variation of a template or test some modifications without altering the original template;
4. The rename template button can be used to rename the currently selected template. HelpNDoc will make sure the newly entered name is valid and unique;
5. The delete template button can be used to delete the currently selected template. **Warning:** this action will permanently delete the selected template from the current system and this action can't be undone;
6. The open template location button can be used to open the template using Windows Explorer and edit it from there. **Note:** the template editor will be closed to avoid problems due to edition in multiple places.

Customizing a template

Once a template is selected in the template editor, various customizable sections are available. Those sections differ based on the kind of templates currently being selected.

Note: Standard templates, which are included with HelpNDoc's installation are read-only and can't be modified.

- [HTML based templates](#): the general settings, variables, script files and assets can be customized. HTML based templates include the following documentation formats: CHM,

HTML, ePub, Kindle, Qt Help and Markdown;

- [Word and PDF templates](#): the page settings, cover page, headers, footers, table of contents and topic titles can be customized.

HTML based templates

HTML based templates include CHM, HTML, ePub, Kindle, Qt Help and Markdown templates. In the [template editor](#), select a HTML based template to access its customizable settings:

- [General settings](#): customize the file extension, inheritance and other general settings for that template;
- [Variables](#): manage variables which can be used by the template;
- [Script files](#): manage script files used to generate the final documentation;
- [Assets](#): manage assets bundled with the final documentation;
- [HTML Tags](#): manage HTML code generated for some library items such as videos;
- [Hooks script](#): manage hooks which are executed during template generation to customize the output;

Note: Changes made in the template editor (including editing scripts and assets) are only saved to disk when clicking the "Save" button.

General settings

Once you have selected an [HTML based template](#) in the [template editor](#), access the "General settings" group in the "Edit template..." panel to manage the following settings for that template:

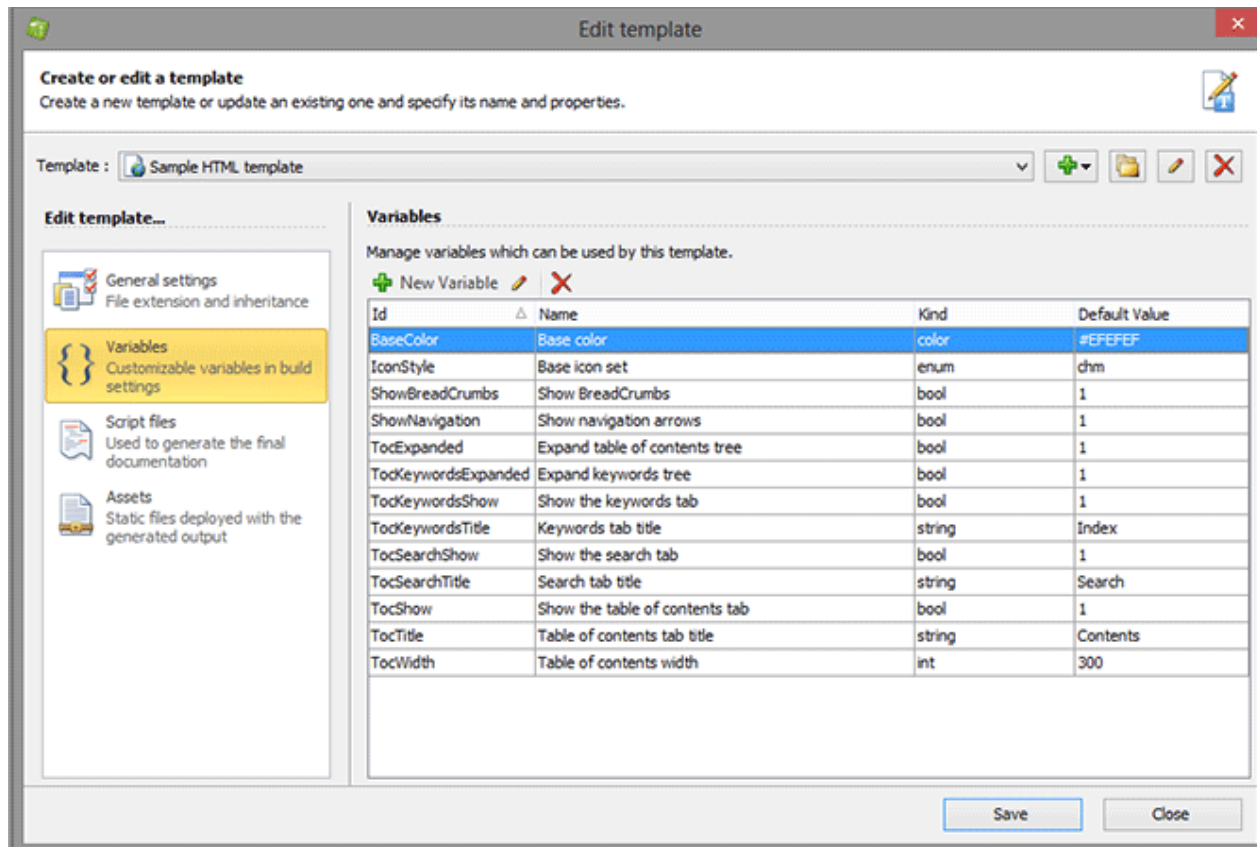
- Inherits from: templates can inherit from other templates and override only specific settings, scripts and assets. Select the parent template here if needed. See: [Template inheritance](#)
- Default file extension: indicates the extension which will be suggested by HelpNDoc when you create a new build using that template;
- Topic file extension: indicates the extension which should be used by templates to produce topic files;
- Link settings: define how internal links will be generated by HelpNDoc. See: [Handle the generated topic links](#)

Other HTML based template settings:

- [Variables](#)
- [Script files](#)
- [Assets](#)

Variables

Once you have selected an [HTML based template](#) in the [template editor](#), access the "Variables" group in the "Edit template..." panel to manage variables which can be used by this template. When a variable is defined for a template, it can easily be customized from the [build window](#) for each build using that template, and [scripts](#) from this template can easily access its customized value to act upon it.



Create a new variable

By using the "New Variable" button, the "manage a template variable" window is shown with the following fields:

- **Id:** specify a unique identifier for that variable. This identifier will be used in the script files to access the customized content for that variable
- **Name:** the name of the variable as shown in the build window
- **Description:** the description of the variable as shown in the build window
- **Kind:** the kind of the variable which will provide a simpler way to specify its value in the build window.
 - Bool: the variable contains a boolean value which can be either true or false
 - Color: the variable contains a color value

- Enum: the variable contains a specific value chosen from one of the "Options" field (see "Options" bellow)
- Int: the variable contains an integer value
- Libpicture: the variable contains a reference to a picture available in the library of the currently open project. This is used to select an eBook cover for example
- String: the variable contains a piece of text
- Memo: the variable contains multi-line text
- **Default value:** when needed, the variable can have a default value which will be used if no other value is entered in the build window
- **Options:** for "Enum" variables, provides a list of available items to choose from, separated using the | character. E.g. "value1|value2|value3" will provide a choice between 3 values
- **Translations:** it is possible to translate the name and description of the variable to other languages supported by HelpNDoc. If needed, enter the translated values here so they can be stored in the template files, and displayed if HelpNDoc is set up in one of those languages

Edit a variable

Once a variable is selected in the list, the **Edit Variable** button will display the same window as when creating a new variable is shown (see "Create a new variable" above). Only the "Id" field is grayed out as it can't be modified for an existing variable.

Delete a variable

Once a variable is selected in the list, the **Delete Variable** button will delete that variable. Deleted variables won't be displayed in build settings anymore and can't be used by script files.

Other HTML based template settings:

- [General settings](#)
- [Script files](#)
- [Assets](#)

Script files

Once you have selected an [HTML based template](#) in the [template editor](#), access the "Script files" group in the "Edit template..." panel to manage script files for this templates.

Script files are the heart of HTML based templates as they include [a mix of HTML and Pascal code](#) used to instruct HelpNDoc on how to generate the final documentation. Using those very powerful script files, it is possible to customize almost any part of the generated documentation

files.

Script files' names must adhere to the following pattern: FILENAME.pas.EXTENTION where:

- FILENAME is the name of the script file and can be any valid file name;
- EXTENTION is usually the final extension this file will generate.

As an example HelpNDoc's default HTML template includes the "topics.pas.html" file which is used to generate HTML files for topics.

Create a new script

By using the "New Script" button, the Script Editor window is displayed where it is possible to:

- Enter the new script's name
- Build the script to make sure it doesn't contain any error
- Access the help file with [methods available](#) in templates
- Enter the content of the script using an editor with syntax highlighting
- Save or cancel the edition of the script

Edit a script

Once a script is selected in the list, use the "Edit Script" button to access the script editor and change its name or content.

Rename a script

Once a script is selected in the list, use the "Rename Script" button to rename it.**Note:** a script must contain the ".pas." text in its name before its extension (e.g. index.pas.html).

Delete a script

Once a script is selected in the list, use the "Delete Script" button to delete that script from the currently selected template. This script won't be run anymore when the final documentation is generated.

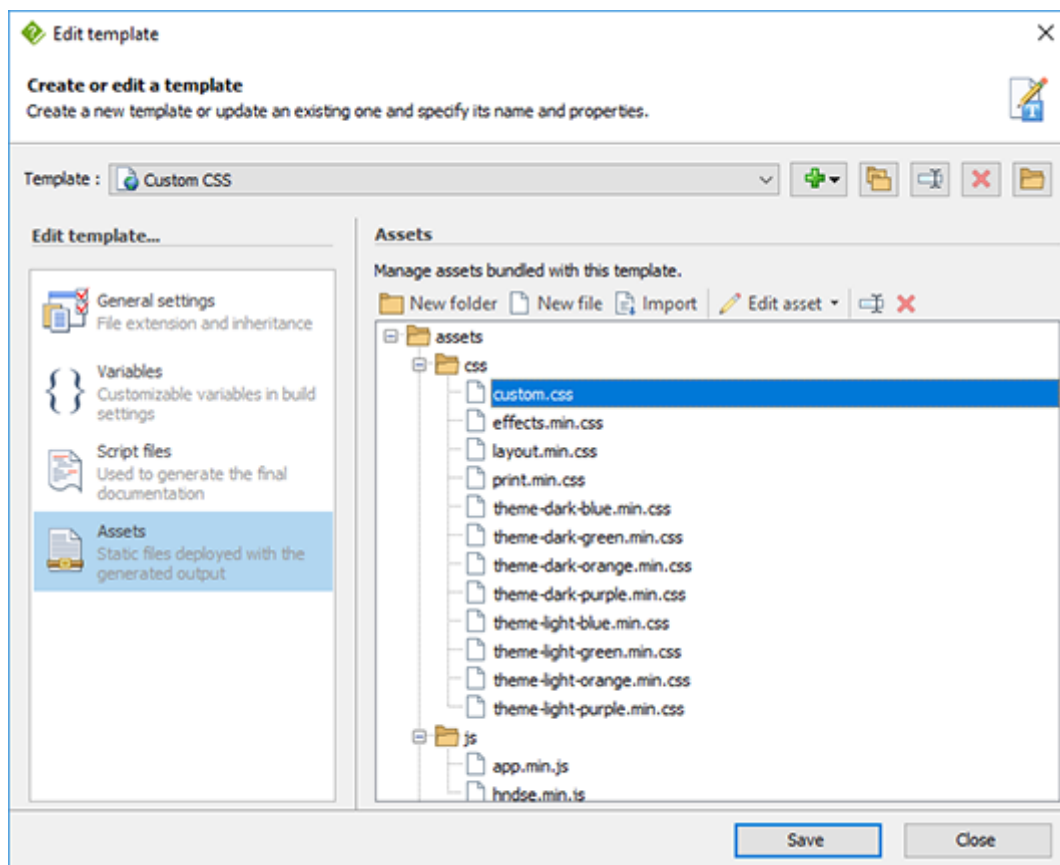
Other HTML based template settings:

- [General settings](#)
- [Variables](#)
- [Assets](#)

Assets

Once you have selected an [HTML based template](#) in the [template editor](#), access the "Assets" group in the "Edit template..." panel to manage assets bundled with the currently selected template.

Assets are static files which will be deployed in the same directory as the generated documentation. Assets are usually used to add CSS, JavaScript or Images to the final documentation but they are not limited to those kind of files: any file type can be added as an asset to a template.



The assets hierarchy displays a list of all assets bundled with the currently selected template. It is possible to:

- Create folders using the "New folder" button
- Create files using the "New file" button
- Import new files as assets using the "Import" button
- Edit the currently selected file using an external editor. See [Editing assets](#)
- Rename the currently selected folder or asset using the "Rename asset" button
- Delete the currently selected folder or asset using the "Delete asset" button
- Move an asset to a different folder by dragging it and dropping it to the desired location

Other HTML based template settings:

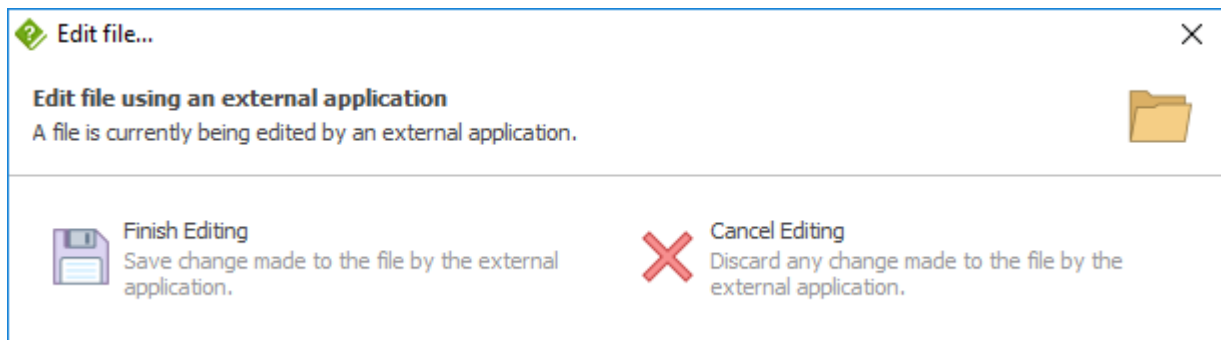
- [General settings](#)
- [Variables](#)
- [Script files](#)

Editing assets

There are two ways of editing an asset:

- **Edit asset** opens the system's default editor for this file kind;
- **Edit asset with...** opens the standard Windows "Open With" dialog to choose the editing application.

While an asset is being edited by a third-party application, the "Edit file" dialog is shown by HelpNDoc.



To edit an asset:

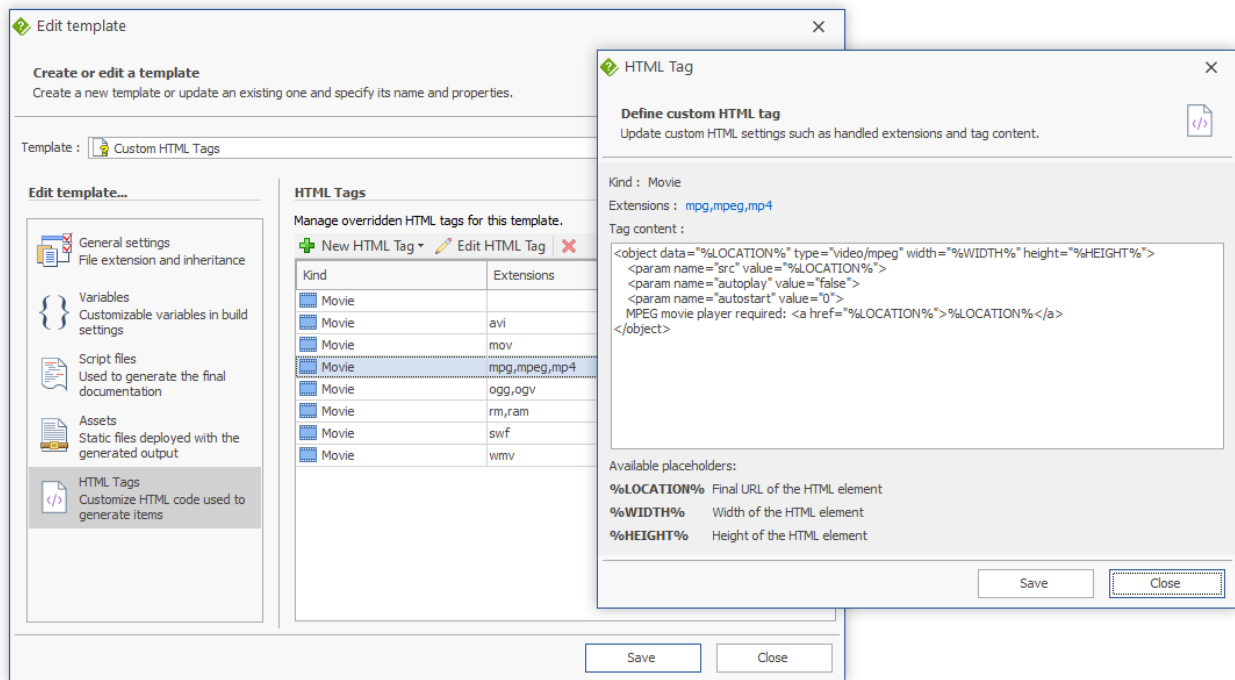
- Update the file using the external editor (e.g. Notepad);
- Save the file from that external editor and close it if needed;
- Click "Finish Editing" in the "Edit file" dialog

Note: the asset is not yet saved to disk, the template needs to be saved from the template editor dialog.

To cancel an asset edition, click "Cancel Editing" in the "Edit file" dialog. Even if the asset file has been edited and saved using a third party editor, it won't be updated in the template.

HTML tags

Once you have selected an [HTML based template](#) in the [template editor](#), access the "HTML tags" group in the "Edit template..." panel to manage HTML tags which are generated by this template. HTML tags are HTML code snippets which are used by HelpNDoc to generate support code for some library items such as videos.



Create a new HTML tag

By using the "New HTML Tags" drop-down menu, a list of available HTML tag kind is displayed:

- **New movie tag** - Create a new HTML tag to customize the export of movies

The following properties must be defined for each HTML tag:

- **Extensions** - This HTML tag will only be used for these extensions. E.g. "mov,avi,mpeg". **Note:** The following extensions have a specific meaning:
 - Empty extension or **<all>** - Use an empty extension to specify the default HTML embed code to use for all extensions not defined in that template
 - **"youtube"** or **"vimeo"** - Use those extensions to customize the HTML embed code for those online video hosting providers;
- **Tag content** - Specify the HTML code snippet which will be used to generate the final HTML code for these extensions. Use the following placeholder to let HelpNDoc generate the proper HTML code:
 - **%LOCATION%** - Final URL of the HTML element as defined within the project
 - **%WIDTH%** - Width of the HTML element as defined within the project
 - **%HEIGHT%** - Height of the HTML element as defined within the project

Example

Let's say that our template should produce a custom HTML code for the following video extensions: MPG, MPEG and MP4. Here is how we can proceed:

- Create a "New HTML Tag" / "New movie tag"
- Add the following extensions to the list: "mpg", "mpeg" and "mp4"
- Specify the custom HTML code to use. For example:


```
<object data="%LOCATION%" type="video/mpeg" width="%WIDTH%"
height="%HEIGHT%">
  <param name="src" value="%LOCATION%">
  <param name="autoplay" value="false">
  <param name="autostart" value="0">
  MPEG movie player required: <a href="%LOCATION%">%LOCATION%</a>
</object>
```
- Click "Save"

HelpNDoc will now use this HTML snippet to generate the proper HTML code each time one of those video extensions is used within the project.

Hooks script

Once you have selected an [HTML based template](#) in the [template editor](#), access the "Hooks script" group in the "Edit template..." panel to manage the custom script used to hook some template generation methods. Using hooks, it is possible to alter HelpNDoc's generation process to fine-tune it for specific requirements.

Hook "Help Id" generation

The `Hook_HelpIds` function can be used to customize how the [Help ID topic property](#) is produced. It is particularly useful for the HTML documentation format as the Help ID is used to produce each topic's file name and therefore, the URL used to access specific topics. Customizing the Help ID can therefore be used for **search engine optimization (SEO) purposes**.

Simple hook function

The following `Hook_HelpIds` function simply makes the Help ID upper-case:

```
function Hook_HelpIds(aTopicID: string; aHelpId: string): string;
begin
  Result := aHelpId.ToUpper();
end;
```

URL-encoded caption hook function

The following `Hook_HelpIds` function will not use HelpNDoc's Help ID at all, but instead it will:

- Get the topic's caption
- Encode it to make it a valid URL

```
function Hook_HelpIds(aTopicID: string; aHelpId: string): string;
begin
    Result := HndUtils.UrlEncode(HndTopics.GetTopicCaption(aTopicId));
end;
```

More advance hook function

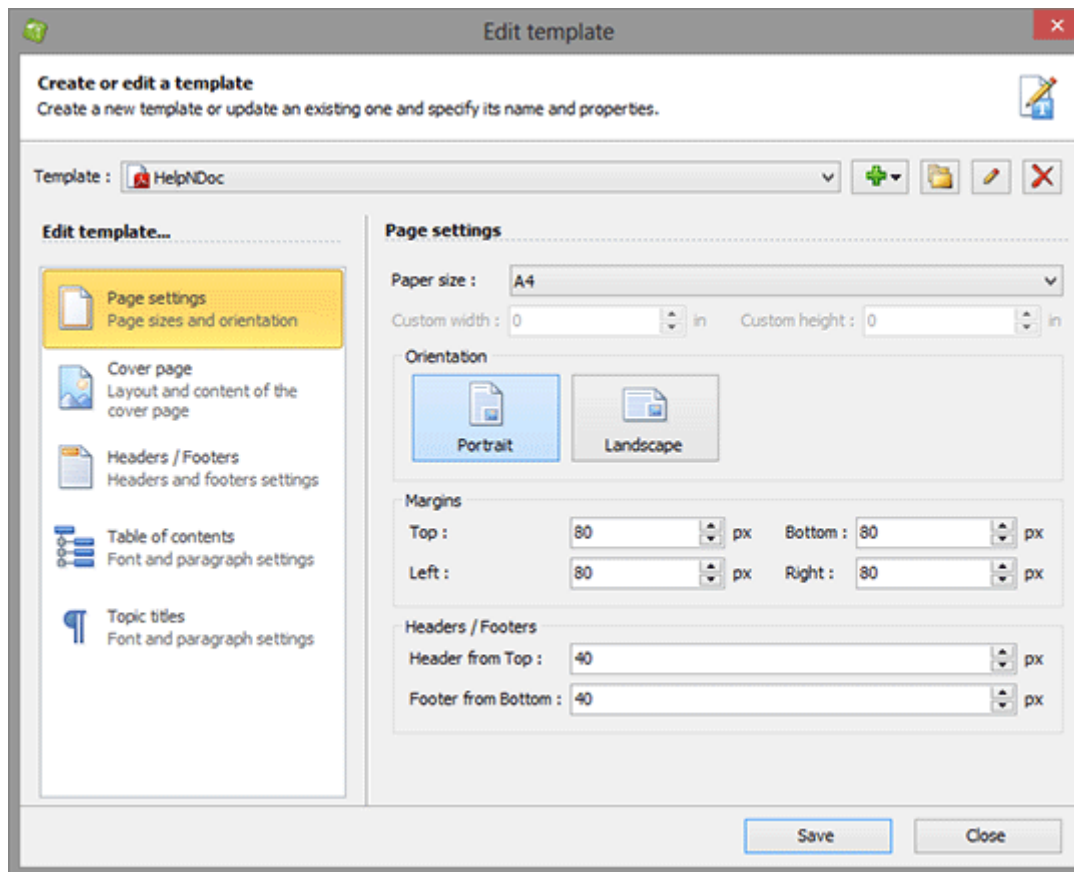
The following Hook_HelpIds function will not use HelpNDoc's Help ID at all, but instead it will:

- Get the topic's caption
- Make it lower-case
- Replace any spaces with a dash
- Filter any non alpha-numeric content

```
function Hook_HelpIds(aTopicID: string; aHelpId: string): string;
begin
    Result :=
HndUtils.FilterAlphaNumericString(HndTopics.GetTopicCaption(aTopicID).toLowerCase().replace(' ', '-'), False, True, True);
end;
```

Word and PDF templates

Word and PDF templates can be customized using the [template editor](#). Select a Word or PDF template to access its customizable settings:



- Page settings: manage page size, orientation, margins, headers and footers sizes
- Cover page: customize the content of the cover page
- Header / Footers: customize the content of the headers and footers
- Table of contents: customize the look and feel of the table of contents including [Text layout](#)
- Topic titles: customize the look and feel of the topic titles including [Text layout](#)

Text layout

Use the text layout dialog to define how the specified text (title, table of contents entry...) is displayed in the generated documentation.

- Font settings: define the font and its formats
- Paragraph settings: define paragraph settings including numbering format
- Borders: define visible borders and settings

Numbering format

Define how a topic numbering is formatted. E.g. topic "2.3.4" with numbering format "n-R-a" will be displayed as "2-III-d"

Possible values:

Placeholder value	Definition	Sample
n	Integer value	2 becomes 2
r	Lowercase Roman value	2 becomes ii
R	Uppercase Roman value	2 becomes II
a	Lowercase alpha value	2 becomes b
A	Uppercase alpha value	2 becomes B
\t	Tabular character	
Anything else	Displayed as-is	

Low-level template details

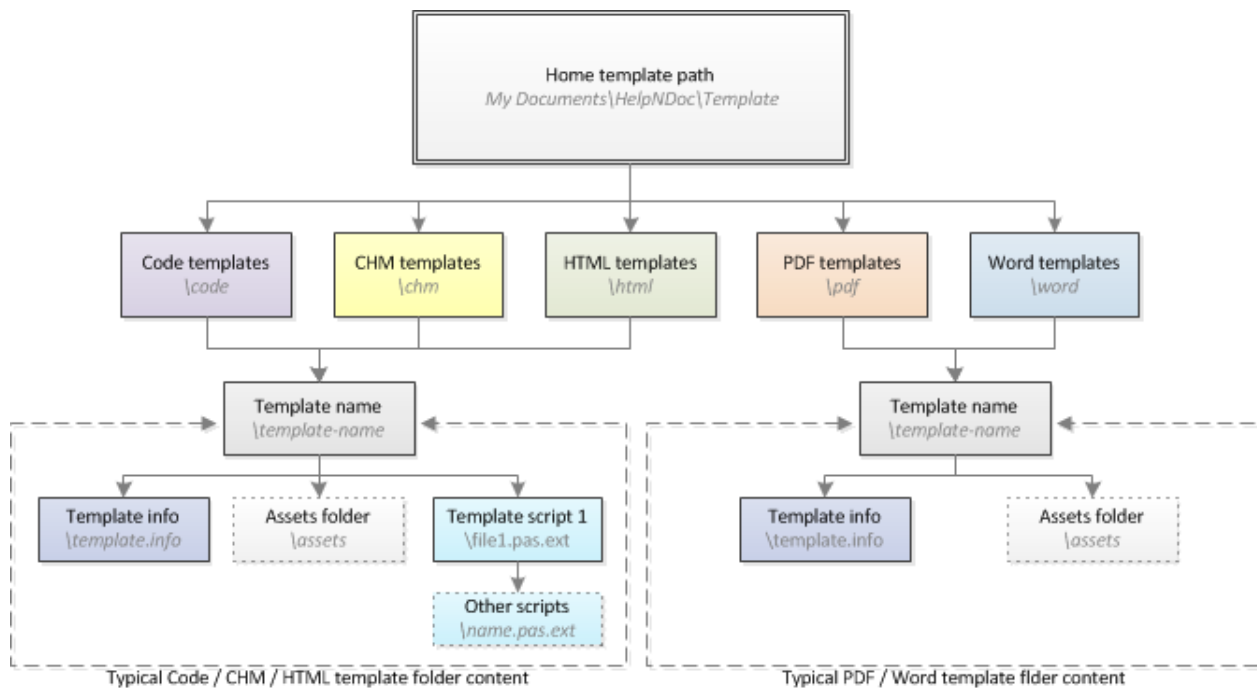
The [template editor](#) is a visual interface which simplifies the creation and management of templates. Behind the scenes, when a modification is made through the template editor, it is stored to the template files on the hard drive, following a specific convention. This section explains the template's low-level details, starting with the [best practices](#).

Best practices

HelpNDoc comes with a set of default (Standard) templates for all the documentation formats. Those templates are located in the "Templates" sub-directory of the HelpNDoc's installation directory, usually under "Program Files\IBE Software\HelpNDoc\Templates".

In addition to that, a user template directory is created when HelpNDoc is installed. It is located under "My Documents\HelpNDoc\Templates" and can be customized in HelpNDoc's options window.

Recent versions of Windows won't allow non-administrator users to change anything in the "Program Files" directory, that's why it is recommended to edit all the templates in the "My Documents" template directory instead.



Template kind sub-directories

Templates are located in the following sub-directories based on their action:

- `chm` - Templates used to generate compiled HTML Help documentation
- `code` - Templates used to generate code for various programming languages
- `epub` - Templates used to generate ePub eBooks
- `html` - Templates used to generate on-line HTML documentation
- `markdown` - Template used to generate Markdown files
- `mobi` - Templates used to generate MobiPocket / Kindle eBooks
- `pdf` - Templates used to generate PDF documentation
- `qthelp` - Templates used to generate Qt Help files
- `word` - Templates used to generate Word documentation

Assets

A template can contain an optional "assets" folder. All the files and sub-folders contained in that folder will be copied in the documentation's output directory. This is useful to add external files to the templates, such as CSS or JavaScript to HTML templates. Note: The content of the "assets" folder will be copied directly in the generated documentation's output directory, not in an "assets" sub-directory.

Modify a default template

- Copy the default template's directory from "Program Files\IBE

Software\HelpNDoc\Templates\TEMPLATE-KIND\TEMPLATE-NAME" to the user's template directory under "My Documents\HelpNDoc\Templates\TEMPLATE-KIND\NEW-TEMPLATE-NAME"

- Edit the [template.info](#) file to change the template's name
- Add, delete or modify any other file to update the template's content

Template configuration file

The template.info file is a standard INI file located in all the templates folders and is used to specify basic information on that template such as the name, category and extension.

The template.info file requires a "config" section with the following values:

- name - defines the name of the template as shown in the project options and help generation dialog
- category - defines the category of the template and used to combine code templates in the quick generation popup menu
- extension - defines the extension of the main file which will be generated by this template (e.g. "chm" for CHM help files). This only serves as a suggestion for the default file name, as the file extension can be changed for each build
- topicextension - defines the extension of the topics file (e.g. "htm" for CHM help files).
Templates can use this value to generate topic files.

Sample template.info file

The following template.info file describes a sample CHM template:

```
[config]
name=Sample CHM template
category=CHM Documentation
extension=chm
```

Template information can be accessed from script files using the `HndGeneratorInfo.TemplateInfo` object which has the following properties:

- category: string
- name: string
- folder: string
- kind: string
- extension: string
- topicextension: string
- inherits: string

- `standard:` `boolean`
- `variables:` `THndTemplateVariableInfoArray`

Template inheritance

As mentioned in the [best practices](#), it is not advised to modify the default templates provided with HelpNDoc usually located in the "program files" directory. Furthermore, some templates might need only subtle changes to a subset of the files to suit the requirements. That's why HelpNDoc introduces the template inheritance concept where a template can "inherit" from a parent template, thus using all its template files, and only override the required files.

Inheriting from a parent template

From the final children template, just add the "inherits" key in the [template.info](#) file's "config" section and mention the parent template's name as the value. As an example:

```
[config]
name=Child template
category=HTML Documentation
extension=html
inherits=Default HTML Template
```

This configuration file instructs HelpNDoc to use the template named "Default HTML Template" as the parent template. Only template files from the same documentation format can be used as a parent template.

Overriding a template file

To change the content of a file, just place a file with the same name in the same directory within the child template. HelpNDoc will use this file instead of the one from the parent template. As an example, if the parent template contains the file "index.pas.html" it is possible to override this file by creating a file named "index.pas.html" in the child template's folder with alternative content.

How is it working

At generation time, HelpNDoc reads the template selected for the project. If that template inherits from a parent template, HelpNDoc will first generate a new temporary template as follows:

1. The parent template's entire content is copied into a temporary folder;
2. The child template (the one selected for the documentation generation) is copied over this parent template, replacing any duplicate file if required;
3. The `template.info` files are merged to preserve the sections and keys from both templates, and override the duplicate ones using the child template's data.

Limitations

Some limitations apply to the template inheritance feature:

- It is not possible to inherit from a template from a different documentation format: an HTML template can't inherit from a CHM template and a Word template can't inherit from a PDF template for example;
- It is not possible to remove files from the original template, just add or override existing files

Code templates

Code templates are very similar to [CHM and HTML templates](#) except for the fact that they usually won't need access to the topic's contents and can't access to their HTML content. Here are the steps involved to create a code template:

- Create a new folder under "My Documents\HelpNDoc\Templates\Code" with the name of the new template
- Create a new [template.info file](#) in that template folder and add the required name, category and extension
- Create a new file containing the ".pas" text before the extension. As an example, we will create a "sample.pas.txt" file. Only files containing the ".pas" text will be interpreted by HelpNDoc
- Add sample code to that template

In HelpNDoc, the new code template will appear in the "Code Generation" category of the "Generate help" popup menu in the "Project" section of the "Home" ribbon tab.

CHM and HTML templates

The CHM and HTML template system can be used to tailor the output for HTML-based documentation generation. The template system is very similar to the [code template](#) but can also access the topic's HTML content. To learn how to create a new HTML template, see the "[Building a single page HTML template](#)" topic. Also learn how to:

- [Handle the generated topic links](#)
- [Methods available in templates](#)
- [Generate multiple files from a single template file](#)
- [Template variables](#)
- [Assets](#)
- [HTML tags](#)

Handle the generated topic links

HelpNDoc will automatically generate links to topics and anchors for you. By default, it assumes that topics will be generated in a file called "%helpid%.html" where "%helpid%" is the value of the help id of that topic. However, this is not always the case so HelpNDoc provides a way of customizing the format of the generated topic and anchor links. They can be modified in the "config" section of the [template.info](#) file. Here is a sample:

```
linkformattopic=##%helpid%
linkformatanchor=##%anchorname%
```

The possible variables to be used in the topic and anchors link formats are:

- %topicid% - The internal HelpNDoc's managed unique topic id
- %helpid% - The topic's help id value
- %anchorname% - The name of the anchor as specified in HelpNDoc, if any

Methods available in templates

HTML and CHM templates can leverage various methods to get information or manipulate the currently opened project. See [methods available in the HelpNDoc API](#).

Generate multiple files from a single template file

When interpreting a template file, HelpNDoc will automatically save the printed content to the documentation's output directory and use the template file name, without the ".pas" part. For example, the "topics.pas.html" file will be interpreted and the result will be written in the "topics.html" file.

As it isn't possible nor sane to create a template file for every single file HelpNDoc has to generate, the template system has a special property to switch the file currently being written. The code required to do that is:

```
HndGeneratorInfo.CurrentFile := 'my-new-file.html';
```

When the template system interprets that line, it will automatically output any further content to the file specified, in the documentation's output directory. This trick is used by the CHM and HTML templates to output each individual topics into their own HTML file:

```
// Loop through each topics
for nCurTopic := 0 to length(aTopicList) - 1 do
begin
  // Change the current output
  HndGeneratorInfo.CurrentFile := aTopicList[nCurTopic].HelpId + '.html';
  // Write the content of the topic to that file
```

```
// ...
end;
```

Template variables

CHM and HTML templates can define template variables in the [template.info](#) file. Those variables will be presented in a user-friendly way in the documentation generation dialog and will be saved within the project file. The template can request for user-defined values and act upon them to customize itself based on user input. Possible template variables usages include:

- Making a section optional. This is done in the default CHM and HTML templates with the breadcrumbs line which can be hidden from generated documentation
- Customizing the documentation appearance. In the HTML template, it is possible to specify a base color, an icon set, default tree expansion status...
- Provide localized texts. The default HTML template defines the captions for the "Index", "Search" and "Content" tabs as variables so that it is possible to translate them from within HelpNDoc

Defining template variables

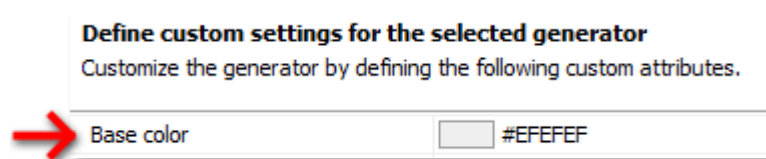
Template variables are defined as sections in the `template.info` file. Those sections' name must begin with the `var_` keyword followed by a unique variable identifier. Let's consider the following sample variable section:

```
[var_BaseColor]
name=Base color
kind=color
default=#EFEFEF
description=Customize the documentation's base color.
```

This defines a new variable with the following information:

- Identifier is BaseColor
- Name is Base color
- Kind is color
- Default value is #EFEFEF
- Description is Customize the documentation's base color.

This will be displayed in the HelpNDoc template settings dialog as follows:



Variables section attributes

The following attributes should be defined in a variable sections:

Attribute	Description	Remarks
name	Name of the variable	Will be displayed to identify the variable in the settings dialog
kind	Kind of variable	Can be bool, color, enum, int, libitem, memo, string. See kinds of variables
default	Choose between multiple values	Default value for this variable if not set in HelpNDoc
values	Possible values for this variable	Only for enum variables, using the pipe character as a separator. Example: "blue red"
description	Explanation for this variable	Will be displayed to explain the purpose of this variable

Kinds of variables

A variable can be set as one of the following kinds depending on its purpose:

Kind	Description	Remarks
bool	Conditional Yes/No value	Often used for a conditional section which will be displayed or not based on its value
color	Standard color value	Can be used to define the color of a specific element
enum	Choose between multiple values	Use the "values" attribute to specify the possible values. Example: "values=chm folder vista"
int	Integer value	

libitem	Select a library item	Gives the ability to select an item from the library
memo	Memo value	
string	Character, word or sentences	

Setting-up template variables

Variables are set-up from within HelpNDoc's document generation dialog. CHM and HTML templates provide a "Customize" link which show the template customization dialog. This dialog lists all the variables for this template, and provides a way to customize them. Customized variable values will be saved with the current project.

Requesting template variables

To request the value of a template variable from within the template itself, use the `HndGeneratorInfo.GetCustomSettingValue` method specifying the variable identifier. As an example, the `BaseColor` value will be requested as follows:

```
HndGeneratorInfo.GetCustomSettingValue( 'BaseColor' );
```

Assets

Any file or folder placed in the "assets" sub-folder of an HTML based template won't be processed by HelpNDoc: they will be copied to the root directory of the generated documentation.

Assets can be used to generate additional images, CSS style-sheets, JavaScript code or any other baggage file or folder that might be needed alongside the generated HTML based documentation.

Managing assets

To manage assets for a specific template, simply create the "assets" sub-folder in that template's directory, and add files to that directory. Any file added there will be copied as-is in the final output folder.

Note: Even though they are placed in the template's "assets" sub-directory, they will be copied to the root of the generated folder.

HTML tags

The HTML specification frequently evolves and some library items (such as Movies) might need a custom HTML code to be generated based on the targeted platform and its mime type. For this purpose, HelpNDoc's HTML based templates can be customized to generate custom code based on item extension.

Naming convention

HTML tags source files are placed in the templates' "tags" sub-directory using the following file name pattern:

```
KIND_EXTENSIONS.html
```

Where:

- **KIND** - is the library item kind (e.g. movie). Required. Available kinds are: movie
- **EXTENSIONS** - the list of file extensions which will use this source code. Multiple extensions are separated by a comma. Not required.

Sample file names:

File name	Description
movie_avl,mov.html	Will be used to generate the source code for movies with the .AVI or .MOV extension
movie_mp4.html	Will be used to generate the source code for movies with the .MP4 extension
movie.html	Will be used to generate the source code for all movies not handled by another HTML tag file

Placeholders

Within the source code for a specific tag, the following place-holders are available and will be replaced by HelpNDoc at generation time:

Name	Description
------	-------------

%LOCATION%	Final URL of the HTML element
%WIDTH%	Width of the HTML element
%HEIGHT%	Height of the HTML element

Sample content

Let's that we'd like to handle movies with the MP4 file extension. In the template's directory, we can create the tag file named "tags\movie_mp4.html". Its content could be:

```
<video width="%WIDTH%" height="%HEIGHT%" controls>
  <source src="%LOCATION%" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

Samples

- [Building a single page HTML template](#)

Step by step tutorial on how to build a template which will output all the topics in a single page

Building a single page HTML template

In this section, we will create a new HTML template from scratch. This template will create a single-page HTML documentation where all the topics are grouped on that single page. The final version of this template is installed with HelpNDoc and can be found in the "My Documents\HelpNDoc\Templates\html\SinglePage" directory.

Template directory

First we need to create a directory for the new template. Custom templates are located in the "My Documents\HelpNDoc\Templates" directory. As we are creating an HTML template which we'll call "SinglePage", we will create the following directory for our new template: "My Documents\HelpNDoc\Templates\html\SinglePage".

The template.info file

Each template must include a "template.info" file with basic template information. Let's create one in the template directory with the following content:

```
[config]
name=Single page HTML template
```

```
extension=html
```

Template code file

It's time to go to the heart of our template by creating a new file which will contain the code to instruct HelpNDoc on how to generate our HTML file. To be interpreted by HelpNDoc, the file must contain the ".pas" text before its extension, which means it will contain Pascal code. So let's create a file named "index.pas.html" in the template directory and edit using any text editor.

Any text which is added to that file will be written as is in the final output, except if it is included in the `<% %>` tags, which are used to insert special instruction code. The steps involved to create the initial code are:

- Instruct the template system to output the HTML file BOM ([Byte Order Mark](#)). This is only necessary for HTML files in Internet Explorer on Window with some languages (1)
- Instruct the template to output to the user defined file as we are only generating a single file (2)

So the "index.pas.html" file now contains:

```
<%
begin
    // 1. Output BOM for HTML UTF8 files
    HndGeneratorInfo.BOMOutput := True;
    // 2. Instruct the generator to generate the desired output file
    HndGeneratorInfo.CurrentFile := ExtractFileName(HndGeneratorInfo.OutputFile);
%>

<html>
<head>

</head>

<body>
    Sample HTML Code
</body>
</html>

<%
end.
%>
```

Get the topics list

Templates have access to a number of functions, objects and variables to help them generate an output. We first need to get a list of topics available in the current project. To do so, we create a new variable before the "begin" keyword and request the topic list (3). The "index.pas.html" file now contains:

```
<%
```

```
// Variable declarations
var
    // List of topics available in the current project
    aTopicList: THndTopicsInfoArray;

begin
    // 1. Output BOM for HTML UTF8 files
    HndGeneratorInfo.BOMOutput := True;
    // 2. Instruct the generator to generate the desired output file
    HndGeneratorInfo.CurrentFile := ExtractFileName(HndGeneratorInfo.OutputFile);
    // 3. Get the list of topics available
    aTopicList := HndTopics.GetTopicList(False);
%>

<html>
<head>

</head>

<body>
    Sample HTML Code
</body>
</html>

<%
end.
%>
```

Output the topics' content

Now that we have a list of topics, we can output their content by looping through that list. The steps involved are:

- Create an iteration variable (4) - This variable will be used by the loop
- Loop through the topics (5) - The topics are treated one by one in that loop
- Notify the template system about the current topic being generated (6) - The template system can't know which topic is currently treated, that's why we notify it using the HndGeneratorInfo object
- Output the topic content (7) - We ask for the HTML content of the topic and we output it

The "index.pas.html" file now contains:

```
<%
// Variable declarations
var
    // List of topics available in the current project
    aTopicList: THndTopicsInfoArray;
var
    // 4. Current topic index
    nCurTopic: Integer;

// Main program
begin
    // 1. Output BOM for HTML UTF8 files
    HndGeneratorInfo.BOMOutput := True;
```

```

// 2. Instruct the generator to generate the desired output file
HndGeneratorInfo.CurrentFile := ExtractFileName(HndGeneratorInfo.OutputFile);
// 3. Get the list of topics available
aTopicList := HndTopics.GetTopicList(False);
%>

<html>
<head>

</head>

<body>
    <%

        // 5. Loop through all the topics
        for nCurTopic := 0 to length(aTopicList) - 1 do
        begin
            // 6. Notify about the topic being generated
            HndGeneratorInfo.CurrentTopic := aTopicList[nCurTopic].id;
            // 7. Output the topic content
            print(HndTopics.GetTopicContentAsHtml(HndGeneratorInfo.CurrentTopic));
        end;

    %>
</body>
</html>

<%
end.
%>

```

Add the titles

The template now display the whole content of all the topics in a single page. However, no title is displayed. Let's add the topic titles before the topics as well as an HTML anchor to be able to link to that topic later on. The steps involved are:

- Declare a new variable nTopicLevel (8) - This variable will be used to get the level of the topic and output the correct HTML heading to the topic title
- Output an HTML anchor (9) - This anchor will be used to link to that specific topics afterwards
- Get the topic level (10) - We request the level of the current topic so we can output the correct HTML heading
- Output the topic title (11) - We can now correctly output the title of the topic

Between steps (6) and (7) we now add the following lines in the "index.pas.html" file:

```

// 9. Add an anchor to be able to link to that topic
printf('<a name="%s"></a>', [aTopicList[nCurTopic].helpid]);
// 10. Get the topic level
nTopicLevel := HndTopics.GetTopicLevel(HndGeneratorInfo.CurrentTopic);
// 11. Add the topic title
printf('<h%d>%s</h%d>', [nTopicLevel,
HndTopics.GetTopicHeaderTextCalculated(HndGeneratorInfo.CurrentTopic), nTopicLevel]);

```

Adding some style

The output now contains all the content from our documentation but it doesn't look like what we've designed in HelpNDoc. That's due to the fact that we didn't add any style coming from HelpNDoc. This can be done in a single step, by requesting for the style content and adding it into the HTML's head section (12). To do so, we add the following lines in the "<head>" section of the "index.pas.html" file:

```
<style type="text/css">
<%
    // 12. Output global CSS content
    print(HndProjects.GetProjectCssContent());
%>
</style>
```

Fixing the links

The output looks as we designed it now but links to topics are not working correctly. This is due to the fact that by default, HelpNDoc assumes that each topic will be generated in its own file which will be named "%helpid%.html" where "%helpid%" is the help id of that topic, as explained in the "[Handle the generated topic links](#)" topic. This can be customized: to change this default behavior, we need to edit the [template.info](#) file and add the following key/values in the config section. They define the format for topic links and anchor links:

```
linkformattopic=##helpid%
linkformatanchor=##anchorname%
```

Final touches

As we have seen, the possibilities are endless: we could add some custom-made CSS file in the assets folder to customize the HTML headings, add the title of the project, the copyright, completely modify the look and feel of our web-page, split it in sections... Some of those ideas are added in the final sample file which is installed with HelpNDoc and can be found in the "My Documents\HelpNDoc\Templates\html\SinglePage" directory.

Template specifics

Each template provided with HelpNDoc is custom-made to achieve a specific task. Sometimes, templates can have additional features, limitations, or gotchas which are only available on this specific template. Explore detailed insights on template-specific features in the subsequent sections:

- [Default HTML template: JavaScript on page change](#)

Default HTML template: JavaScript on page change

The standard HTML template in HelpNDoc has undergone significant optimizations for enhanced speed in downloading and navigating. This optimization means that when a user selects a topic from the table of contents, HelpNDoc only refreshes the topic's content instead of reloading the entire page. As a result, certain elements like page-specific JavaScript code may not execute when a page is accessed via the table of contents. To address this, HelpNDoc provides a specific event, `app.EVENTS.onTopicChanged`, to track changes in topics.

The usage of this event is as follows:

```
app.EVENTS.onTopicChanged = (sUrl) => { /* Custom code run when a new topic is loaded
*/ }
```

There are multiple ways to listen to this event:

- Add "Custom JavaScript" code to the default HTML template's settings. See: [HTML documentation settings](#)
This will automatically add the code in all generated documentation page
- Create a new custom template and enter the code within the template's generated content.
See: [Working with templates](#)
This provides greater flexibility at the price of additional initial complexity

Usage from the command line

HelpNDoc handles various command line parameters to be able to update and generate documentation without user interface. This is useful to integrate the documentation generation process with an automated build process for example.

Note: The command line syntax has changed in HelpNDoc 5.4 and is not backward compatible. Check [Legacy command line syntax for 5.3 and older](#) to learn more about command line syntax in HelpNDoc 5.3 and earlier.

Command line syntax

The overall command line syntax is as follows:

```
hnd9.exe [project] [global-options] [command] [command-options]
```

Where:

Command line option	Explanation

hnd9.exe	HelpNDoc program
[project]	Path of the HND project file to open or build. Optional. See Project
[global-options]	Options available for every command. Optional. See Global Options
[command]	The command to perform. Optional. See Commands
[command-options]	Command specific options. Optional. See Commands

Project

Indicates the full or relative path of a *.HND project file.

When specified without any command, HelpNDoc's user interface is shown and opens with the specified project.

Note: Mandatory for the build command.

Global options

The following options are global: they can be useful for any commands:

Command line option	Explanation
-help or -h	Help on HelpNDoc: show a list of available commands and options
-log or -l	Indicates the log file path: any information displayed on the command prompt will also be saved to that file. <i>> hnd9.exe myproject.hnd -log=c:\tmp\log.txt build</i>
-openheloid or -oi	Focus the topic with the specified Help ID once the project is opened
-openhelpctx or -oc	Focus the topic with the specified Help context once the project is opened
-reset or -r	Reset HelpNDoc settings. When indicated, settings such as window positions, compiler location... won't be loaded from the registry. The new settings are saved to the registry when the application closes. This can be useful to troubleshoot potential settings problems. <i>> hnd9.exe -r</i>

-silent or -s	Command line is in silent mode: it will automatically close without user interaction
-verbose or -v	Command line is verbose: it will display additional information if available
-verysilent or -ss	Command line is in very silent mode: it won't even open a command window

Commands

The following commands are available from the command line.

Note: Use the -help or -h after the command to get more information about that command.

Command line option	Explanation
build	Build the specified *.hnd project file using either the project's settings or overrides from the command line. See "build" command
license	Information and management of the license. See "license" command
script	Run a script using the HelpNDoc API to automate project creation or modification. See "script" command

"build" command

The build command is used to build a *.hnd project file from the command line, without showing HelpNDoc's user interface.

When run without any options, it will use the project settings to generate all enabled builds. It is possible to override some project settings with command options:

Command line option	Explanation
-except or -e	Generates all project builds except the specified ones. It is possible to use this command multiple times. <i>> hnd9.exe myproject.hnd build -e="Build HTML documentation" -</i>

	<i>e="Build CHM documentation"</i>
-help or -h	Help on this command. <i>> hnd9.exe build -h</i>
-only or -x	Generates only the specified build names. It is possible to use this command multiple times. <i>> hnd9.exe myproject.hnd build -x="Build PDF documentation" -x="Build Word documentation"</i>
-output or -o	Override the output path for a specific build. <i>> hnd9.exe myproject.hnd build -o="Build HTML documentation:c:\www\index.html"</i>
-statuses or -u	Override the statuses included for a specific build. <i>> hnd9.exe myproject.hnd build -u="Build HTML documentation:Complete,Needs Review"</i>
-tags or -a	Override tags generated for a specific build. <i>> hnd9.exe myproject.hnd build -a="Build HTML documentation:clientA,clientB"</i>
-template or -t	Override template used for a specific build. <i>> hnd9.exe myproject.hnd build -t="Build HTML documentation:Legacy HTML framed template"</i>

"license" command

The license command can be used to manage the license of the full version of HelpNDoc.

Available options are:

Command line option	Explanation
-activate or -a	Activate a specific license key on this computer. Make sure the previous license key is deactivated first.

	<p>> <i>hnd9.exe license -a="ABCDE-FGHIJ-KLMNO-PQRST-UVWXY"</i></p> <ul style="list-style-type: none"> • Warning: This requires Internet access to the license servers. If you are behind a proxy, see how to set the proxy below. • Offline activation: To proceed with an offline deactivation, use both the -d and -o command line arguments. > <i>hnd9.exe license -a="ABCDE-FGHIJ-KLMNO-PQRST-UVWXY" -o="c:\tmp\activation-request.xml"</i>
-deactivate or -d	<p>Deactivate a previously activated license on this computer. This makes it possible to move HelpNDoc's license key to another computer.</p> <p>> <i>hnd9.exe license -d</i></p> <ul style="list-style-type: none"> • Warning: The number of allowed deactivations is limited to 5 to limit abuses. If you need more deactivations, please contact us with your license details. • Offline deactivation: To proceed with an offline deactivation, use both the -d and -o command line arguments. > <i>hnd9.exe license -d -o="c:\tmp\deactivation-request.xml"</i>
-forcecheck or -f	<p>If you've recently updated your license of HelpNDoc, it is possible that the license key is still not updated: using that command will connect to the license servers to retrieve the latest license details.</p> <p>> <i>hnd9.exe license -f</i></p> <ul style="list-style-type: none"> • Warning: Make sure you are connected to the Internet to access the license servers.
-help or -h	<p>Help on this command.</p> <p>> <i>hnd9.exe license -h</i></p>
-info or -i	<p>Provides information about the activation of HelpNDoc on this computer.</p> <p>> <i>hnd9.exe license -i</i></p>
-offline or -o	<p>Generate activation / deactivation XML file for offline process.</p> <p>> <i>hnd9.exe license -a="ABCDE-FGHIJ-KLMNO-PQRST-UVWXY" -o="c:</i></p>

	<pre>\tmp\activation-request.xml"</pre> <pre>> hnd9.exe license -o="c:\tmp\deactivation-request.xml"</pre>
-proxy or -p	<p>By default, the license checker will use the proxy set up in Internet Explorer. If you need to customize the proxy, you can indicate the proxy address so that the activation process is able to correctly connect to the license servers. Once done, the proxy address is saved and restored each time the application is launched.</p> <p>Proxy must be in the form "http://username:password@host:port/". Note: if the port is not specified, it will default to 1080.</p> <pre>> hnd9.exe license -p="http://username:password@127.0.0.1:8080"</pre>

"script" command

The script command can be used to execute a custom script using the available [HelpNDoc API methods](#) to automate project creation or modification. Small scripts can be written in the command line, while large scripts can be loaded from a file.

Note: If a project is specified in the command line syntax, it will be opened first before executing the script, thus simplifying existing projects modification.

Command line option	Explanation
-file or -f	<p>Load the script content from the file path specified.</p> <pre>> hnd9.exe script -f="c:\helpndoc\script.pas"</pre> <p>Note: If both the -f and -x command line options are defined, only the -x one will be used.</p>
-help or -h	<p>Help on this command.</p> <pre>> hnd9.exe script -h</pre>
-execute or -x	<p>Run the script code provided directly on the command line.</p> <pre>> hnd9.exe script -x="ShowMessage('OK');"</pre> <p>Note: If both the -f and -x command line options are defined, only the</p>

	-x one will be used.
--	----------------------

Legacy command line syntax for 5.3 and older

Warning: The following command line syntax is valid for HelpNDoc 5.3 and older only. Starting with HelpNDoc 5.4, a [new command line syntax has been introduced](#) which is not backward compatible.

HelpNDoc's command line options use the syntax "**hnd5.exe [FileName] [Parameters]**" where [FileName] is the optional HND file to be processed and the parameters are described bellow. When run using the command line parameters bellow, HelpNDoc won't show any user interface except for a DOS prompt window.

Command line help

At any time, use the "**hnd5.exe /?**" command line to get help on the various command line syntax and parameters.

Command line parameters

- **/g** - Generate the HelpNDoc "FileName" using project's settings
- **/b=[value]** - Override the list of build to generate from the project (Semi-colon separated list of built)
- **/v[name]=[value]** - Set the [value] of variable [name] or create a new variable named [name]
- **/silent** - Silent mode: no user input required. Useful for automated build processes to avoid user interaction
- **/l=[value]** - Output the generation log to the specified file
- **/lic=[value]** - License management. See bellow.

Command line examples

The following is a simple example of a possible use of the HelpNDoc's command line options:

```
> hnd5.exe myHelp.hnd /g
```

This translates to: generate the file "myHelp.hnd" according to the settings saved in that file.

```
> hnd5.exe myHelp.hnd /g /l=c:\log\hnd-log.txt
```

This translates to: generate the file "myHelp.hnd" according to the settings saved in that file and save the log to the file "c:\log\hnd-log.txt"

> hnd5.exe myHelp.hnd /g /b="Build chm documentation";"Build pdf documentation" /vMyVariable=MyValue

This translates to: generate the file "myHelp.hnd" by using the builds named "Build chm documentation" and "Build pdf documentation" and modify or declare the variable "MyVariable" with the value "MyValue".

License key management

The following commands are available from the command line to manage your license.

Command line option	Explanation
/lic=info	Provides information about the activation of HelpNDoc on this computer. Example: <i>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=info</i>
/lic=activate:KEY	Activates a new license key. Make sure the previous license key is deactivated first. Example: <i>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=activate:ABCD-EFGH-IJKL-MNOP-QRST-UVWX-YZAB</i> Warning: This requires Internet access to the license servers. If you are behind a proxy, see how to set the proxy below.
/lic=deactivate	De-activates the current license key. This makes it possible to move HelpNDoc's license key to another computer. Example: <i>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=deactivate</i> Warning: The number of allowed de-activations is limited to 5 to limit abuses. If you need more deactivations, please contact us with your license details.
/lic=forcecheck	If you've recently updated your license of HelpNDoc, it is possible that the license key is still not updated: using that command will connect to the license servers to retrieve the latest license details. Example: <i>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=forcecheck</i> Warning: Make sure you are connected to the Internet to access the

	license servers.
/lic=setproxy:PROXY_ADDRESS	<p>By default, the license checker will use the proxy set up in Internet Explorer. If you need to customize the proxy, you can indicate the proxy address so that the activation process is able to correctly connect to the license servers. Once done, the proxy address is saved and restored each time the application is launched.</p> <p>Proxy must be in the form "http://username:password@host:port/". Note: if the port is not specified, it will default to 1080.</p> <p>Example:</p> <pre>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=setproxy:http://username:password@127.0.0.1:8080</pre> <p>To reset the proxy, simply pass an empty value:</p> <pre>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=setproxy:</pre> <p>NTLM Proxies: The license checker also support NTLM proxies on Windows. To use NTLM proxies you must also specify the domain. For example:</p> <pre>> c:\program files\ IBE Software\HelpNDoc 5\hnd5.exe /lic=setproxy:http://DOMAIN\username:password@127.0.0.1:8080</pre> <p>Warning: If setting the proxy doesn't work, try running HelpNDoc as an administrator. If that still doesn't work, try using the following command line from HelpNDoc's installation directory instead:</p> <pre>> TurboActivate.exe --proxy="http://username:password@host:port/"</pre>

Customize default project styles

It is possible to create and save a set of customized default styles which will be used by default each time a new project is started. Here is a step by step guide on how to achieve that:

1. Create a customized set of styles using the [styles editor](#)
2. At the bottom of the styles editor, click "export..."
3. Styles must be exported to your home HelpNDoc folder, which is "My Documents\HelpNDoc\Styles\default.hns" by default and can be customized in the [Options](#)

[window](#).

For now on, each time a new project is created, the set of styles defined in this file will be loaded and available for that project.

See the [How to customize the default styles](#) for new projects step-by-step guide.

Using the Script Editor

The script editor provides a way to use a programming language, based on the Pascal syntax, to automate HelpNDoc. Almost everything in HelpNDoc can be automated, from project creation to library management. See [methods available in the HelpNDoc API](#).

Bookmarks

When [bookmarks are enabled](#), use the CTRL + SHIFT + 1 to 9 keyboard shortcuts to place bookmarks within the code. Bookmarks can then be reached using the CTRL + 1 to 9 keyboard shortcuts.

Migration

The following topic explains the script modifications needed to migrate to newer versions of HelpNDoc: [Migrating scripts and templates](#)

See the [How to use the script editor](#) step-by-step guide.

Object pascal subset

HelpNDoc's scripting system is a subset of the [Object Pascal language](#). In addition to usual object Pascal code, HelpNDoc provides an extensive set of [API methods](#) which can be used to automate multiple tasks from project creation to control how documentation is generated.

A typical script includes:

1. Variable and constant definitions

```
const
  constant1 = 'Hello';
var
  variable1, variable2: string;
var
  variable3: integer;
```

2. Procedure and function definitions

```

procedure test1();
begin
    variable1 := constant1 + ' World';
end;

function test2(): string;
begin
    Result := constant1 + ' Universe';
end;

```

3. Main program code

```

begin
    test1();
    Print(variable1);
    Print(test2());
end.

```

Template specific code structure

[Template script files](#) usually include a mix of text (e.g. HTML) and Pascal code. To simplify this process and avoid having to use multiple `print` statements, template script files uses a convention similar to ASP.net or PHP code: everything not surrounded by `<%` and `%>` is treated as raw content and exported as-is. As an example:

```

<html>
<head>
    <title><% print('Hello World');%></title>
</head>
<body>
    <% for var i := 0 to 10 do begin %>
    Hello <%= i %>
    <% end; %>
</body>
</html>

```

Will output:

```

<html>
<head>
    <title>Hello World</title>
</head>
<body>
    Hello 0
    Hello 1
    Hello 2
    Hello 3
    Hello 4
    Hello 5
    Hello 6
    Hello 7
    Hello 8
    Hello 9
    Hello 10
</body>
</html>

```

Note: `<%= 'Hello World' %>` is the same as `<% print('Hello World'); %>`

Base types

The following types can be used in scripts and templates:

Type	Description
Boolean	<p>True or False</p> <p>When casting a Boolean as Integer, by convention True maps to 1, False maps to 0. When casting an Integer as Boolean, 0 maps to False, all other values map to True.</p> <p>Methods:</p> <ul style="list-style-type: none"> ToString Conversion to string
Float	<p>Double-precision floating point. 15 significant digits, exponent -308 to +308.</p> <p>Presence of a dot "." differentiates literals from an Integer.</p> <p>Notations:</p> <ul style="list-style-type: none"> Normal. E.g. 948.4685 Exponential. E.g. 9484685e23 <p>Methods:</p> <ul style="list-style-type: none"> ToString Conversion to string
Integer	<p>64-bit signed integer. From -9223372036854775808 to 9223372036854775807.</p> <p>Notations:</p> <ul style="list-style-type: none"> Decimal. E.g. 948 ; -65190 Hexadecimal. E.g. \$1AF or 0x1AF Binary. E.g. 0b10011101 <p>For formatting purposes, the underscore character "_" is accepted and ignored in Hexadecimal and Binary literals.</p>

	Methods: <ul style="list-style-type: none"> ToString Conversion to string
String	<p>Mutable, copy-on-write, 1-based, UTF-16 string. Strings can be delimited by a single or a double-quote.</p> <p>In a single-quoted string, a single quote can be expressed by doubling it. In a double-quoted string, a double-quote can be expressed by doubling it. Double-quoted strings can span multiple lines.</p> <p>Explicit Unicode characters can be specified by using # followed by an integer value (decimal or hexadecimal). Characters specified this way are always understood as Unicode value:</p> <pre>Print('Hello'#13#\$0D'World');</pre> <p>Will print 'Hello' followed by CR+LF (ASCII code 13 and 10), followed by 'World', it can also be defined with:</p> <pre>Print("Hello World");</pre> Methods: <ul style="list-style-type: none"> After(Delimiter: string): string Returns characters after a delimiter Before(Delimiter: string): string Returns characters before a delimiter Contains(SubString: string): Boolean Returns true if the string contains the sub-string DeleteLeft(N: integer): string Deletes N characters to the left DeleteRight(N: integer): string Deletes N characters to the right Dupe(N: integer): string Duplicate the string N times EndsWith(SubString: string): Boolean

	<p>Returns true if the string ends with the sub-string</p> <ul style="list-style-type: none"> • <code>High</code> Index of last letter • <code>Left(N: integer): string</code> Return N characters to the left • <code>Low</code> Index of first letter • <code>LowerCase</code> Converts to ASCII lower case • <code>Length</code> Length of the string • <code>Reverse</code> Returns a version of the string with the characters reversed • <code>Right(N: integer): string</code> Return N characters to the right • <code>Split(Separator: string): array of string</code> Split a string on a separator and returns an array of strings • <code>StartsWith(SubString: string): Boolean</code> Returns true if the string starts with the sub-string • <code>ToBoolean</code> Converts to Boolean • <code>ToFloat</code> Converts to float • <code>toFloatDef(DefaultValue: float)</code> Tries to convert to float, use DefaultValue if not possible • <code>ToInteger</code> Converts to integer • <code>toIntegerDef(DefaultValue: integer)</code> Tries to convert to integer, use DefaultValue if not possible • <code>ToLower</code>
--	--

	<p>Converts to lower case</p> <ul style="list-style-type: none"> • <code>ToUpper</code> Converts to upper case • <code>Trim</code> Trim control characters left and right • <code>TrimLeft</code> Trim left control characters • <code>TrimRight</code> Trim right control characters • <code>UpperCase</code> Converts to ASCII upper case
--	--

Built-in functions

The following functions are built-in HelpNDoc's script engine and can be used in scripts and templates.

String related functions

Function	Description
<code>Chr(i: Integer): Char;</code>	Returns the character for a specified ASCII value.
<code>DupeString(str: string; count: Integer): string;</code>	Returns the concatenation of a string with itself a specified number of repeats.
<code>IntToStr(i: Integer): string;</code>	Converts an integer to a string.
<code>StrToInt(str: string): Integer;</code>	Converts a string that represents an integer into a number.
<code>StrToIntDef(str: string; def: Integer): Integer;</code>	Converts a string that represents an integer into a number with error default.
<code>IntToHex(v: Integer; digits: Integer): string;</code>	Returns the hexadecimal representation of an ordinal number.

<code>HexToInt(hexa: string): Integer;</code>	Returns the integer value of an hexadecimal representation.
<code>BoolToStr(b: Boolean): string;</code>	Converts a Boolean value into a string.
<code>StrToBool(str: string): Boolean;</code>	Converts a string to a Boolean value.
<code>FloatToStr(f: Float): string;</code>	Converts a floating-point value to a string.
<code>StrToFloat(str: string): Float;</code>	Converts a given string to a floating-point value.
<code>StrToFloatDef(str: string; def: Float): Float;</code>	Converts a given string into a floating-point value, with a default error.
<code>Format(fmt: string; args array of const): string;</code>	<p>Returns a formatted string assembled from a format string and an array of arguments.</p> <p>The following table summarizes the possible values for type:</p> <ul style="list-style-type: none"> • <code>d</code> - Decimal. The argument must be an integer value. The value is converted to a string of decimal digits; • <code>u</code> - Unsigned decimal. Similar to <code>d</code>, but no sign is output; • <code>e</code> - Scientific. The argument must be a floating-point value. The value is converted to a string of the form "-d.ddd...E+ddd"; • <code>f</code> - Fixed. The argument must be a floating-point value. The value is converted to a string of the form "-ddd.ddd..."; • <code>g</code> - General. The argument must be a floating-point value. The value is converted to the shortest possible decimal string using fixed or scientific format; • <code>n</code> - Number. The argument must be a floating-

	<p>point value. The value is converted to a string of the form "-d,ddd,ddd.ddd...". The n format corresponds to the f format, except that the resulting string contains thousand separators;</p> <ul style="list-style-type: none"> • m - Money. The argument must be a floating-point value; • s - String. The argument must be a character, or a string; • x - Hexadecimal. The argument must be an integer value. <p>To display the character % (that is, to display a literal %, not to begin a format specifier), use the sequence %%. Example: Format('%d%%', [50]); // Displays 50%</p>
LowerCase(str: string): string;	Converts a string to lowercase.
UpperCase(str: string): string;	Converts a string to uppercase.
PadLeft(str: string; count: Integer; char: Char = ' '): string;	Left-aligns a string into a fixed length text using the character specified.
PadRight(str: string; count: Integer; char: Char = ' '): string;	Right-aligns a string into a fixed length text using the character specified.
Pos(needle: string; haystack: string): Integer; Pos(needle: string; haystack: string; offset: Integer): Integer;	Locates a substring in a given string. Start at offset if specified.
ReverseString(str: string): string;	Returns the reverse of a specified string.
SameText(str1: string; str2: string): Boolean;	Compares two strings by ordinal value without case sensitivity.

<code>StringOfChar(ch: Char; count: Integer): string;</code>	Returns a string with a specified number of repeating characters.
<code>SubStr(str: string; start: Integer; Length: Integer = MaxInt): string;</code>	Returns a specified substring the specified string.
<code>TrimLeft(str: string): string;</code>	Removes blank and control characters from the left side of a string.
<code>TrimRight(str: string): string;</code>	Removes blank and control characters from the right side of a string.
<code>Trim(str: string): string;</code>	Removes blank and control characters from both sides of a string.

HelpNDoc API methods

The HelpNDoc API is based on the [Pascal programming language](#). The following list describes the methods available via the HelpNDoc API. As an example for the `ClearDictionaries` method, it can be used as follows: `HndDictionaries.ClearDictionaries()`;

List of objects

- [HndUtils](#) - Various utility methods
- [HndStyles](#) - Styles related API methods
- [HndBuilds](#) - Properties and methods for Builds.
- [HndBuildsEx](#) - Additional properties and methods for Builds.
- [HndBuildsActions](#) - Handle pre- and post-build actions
- [HndBuildsLibraryOverrides](#) - API functions to manage library overrides for individual builds
- [HndBuildsMeta](#) - Access to builds meta data
- [HndBuildsMetaEx](#) - Additional methods related to builds meta data

Global types

- [THndBuildInfo](#) - Information about a specific build
- [THndBuildInfoArray](#) - Array of THndBuildInfo
- [THndBuildActionInstanceInfo](#) - Information about a specific build action
- [THndBuildActionInstanceInfoArray](#) - List of build action info
- [PHndBuildLibraryOverrideInfo](#) - Pointer to THndBuildLibraryOverrideInfo
- [THndBuildLibraryOverrideInfo](#) - Information about a specific build library override
- [THndBuildLibraryOverrideInfoArray](#) - Array of THndBuildLibraryOverrideInfo

- [HndBuildsStatus](#) - Handle relationship between builds and statuses
- [HndBuildsStatusEx](#) - Additional methods to handle relationship between builds and statuses
- [HndBuildsStyles](#) - Handle overridden styles for each builds
- [HndBuildsTags](#) - Handle relationship between builds and tags
- [HndBuildsTagsEx](#) - Additional methods to handle relationship between builds and tags
- [HndDictionaries](#) - Manage dictionaries and live spell check
- [HndEditor](#) - Create and manage a topic editor
- [HndEditorHelper](#) - Additional methods to manage a topic editor
- [HndJsSearchEngine](#) - Methods to manage the JavaScript search engine
- [HndKeywords](#) - Properties and methods for keywords
- [HndKeywordsMeta](#) - Access to topics meta data
- [HndLibraryItems](#) - Properties and methods for library items
- [HndLibraryItemsEx](#) - Additional properties and methods for library items
- [HndLibraryItemsMeta](#) - Access to library items meta data
- [HndProjects](#) - Properties and methods for projects
- [HndProjectsEx](#) - Additional properties and methods for projects.
- [HndProjectsMeta](#) - Access to project meta data
- [HndProjectsMetaEx](#) - Additional methods to access project meta data
- [THndBuildStyleInfo](#)
- [THndBuildStyleInfoArray](#)
- [THndDictionaryInfo](#) - Information about a specific dictionary
- [THndDictionaryInfoArray](#) - Array of THndDictionaryInfo
- [THndCaretMovement](#) - Caret movement
- [THndVAlign](#) - Vertical alignment of a picture
- [THndHVAlignment](#) - Table cell alignment
- [THndHyperlinkKind](#) - Kind of hyperlink
- [THndHyperlinkInfo](#) - Information about a hyperlink
- [THndGeneratorInfo](#) - Information and actions over the current generation
- [THndKeywordsAttachMode](#) - Specify how the moved keyword will be attached: - hkamAdd: Adds a node at the same level as the existing node. - hkamAddChild: Adds a child node to the existing node.
- [THndKeywordsInfo](#) - Minimal keyword information. Used by the THndKeywordsInfoArray
- [THndKeywordsInfoArray](#) - Array of minimal keyword information
- [THndLibraryItemAttachMode](#) - Specify how the moved library item will be attached: - hlamAdd: Adds a node at the same level as the existing node and makes the new node last. - hlamAddFirst: Adds a node at the same level as the existing node and makes the new node first. - hlamAddChild: Adds a child node to the existing node and makes the new node last. - hlamAddChildFirst: Adds a child node to the existing node and makes the new node first. - hlamInsert: Inserts a node at the same level just before the existing node.

- [HndSecrets](#) - Properties and methods for Secrets
- [HndStatus](#) - Properties and methods for Statuses
- [HndStyles](#) - Properties and methods for Styles
- [HndTags](#) - Properties and methods for Tags
- [HndTemplates](#) - Properties and methods for Templates
- [HndTemplatesEx](#) - Additional properties and methods for Templates
- [HndTopics](#) - Create, edit and manage topics within the current project.
- [HndTopicsEx](#) - Additional properties and methods for Topics
- [HndTopicsKeywords](#) - Handle relationship between topics and keywords
- [HndTopicsKeywordsEx](#) - Additional properties and methods to handle relationship between topics and keywords
- [HndTopicsMeta](#) - Access to topics meta data
- [HndTopicsProperties](#) - Handle relationship between topics and properties
- [HndTopicsTags](#) - Handle relationship between topics and tags
- [HndUI](#) - Methods and properties of HelpNDoc's user interface.
- [THndLibraryItemsInfo](#) - Minimal library item information. Used by the THndLibraryItemsInfoArray
- [THndLibraryItemsInfoArray](#) - Array of minimal library item information
- [THndProjectSettings](#) - Various project settings
- [THndProjectDateTimeFormat](#) - Date and Time format for a project
- [THndStatusInfo](#) - Information about a specific status
- [THndStatusInfoArray](#) - Array of THndStatusInfo
- [THndTopicsAttachMode](#) - Specify how the moved topic will be attached:
 - htamAdd: Adds a node at the same level as the existing node and makes the new node last.
 - htamAddFirst: Adds a node at the same level as the existing node and makes the new node first.
 - htamAddChild: Adds a child node to the existing node and makes the new node last.
 - htamAddChildFirst: Adds a child node to the existing node and makes the new node first.
 - htamInsert: Inserts a node at the same level just before the existing node.
- [THndTopicsPropertyInfo](#) - Information about a specific property
- [THndTopicsPropertyInfoArray](#) - Array of THndTopicsPropertyInfo
- [TUIControlType](#) - Reference to a specific UI control: uiMainForm, uiTreeToc, uiTreeKeywords, uiTreeLibrary, uiTopicEditor
- [THndSelectedExecMethod](#) - Method run for each selected node. Return True to break the loop: function(aSelectedId:

string): Boolean;

Var HndUtils

Various utility methods

- `function FilterAlphaNumericString(aStr: string; const doKeepSpaces: Boolean; const doKeepUnderscore: Boolean; const doKeepDash: Boolean): string;`
Filter a string to make it alpha-numeric only. Converts accented characters too.
- `function HexToTColor(sColor: string): TColor;`
Converts an hexadecimal value to a TColor
- `function HTMLDecode(S: string): string;`
Converts a string that has been HTML-encoded into a decoded string
- `function HTMLEncode(S: string): string;`
Converts a string into an HTML-encoded string
- `function HTMLEscape(S: string): string;`
Escape HTML entities so that they can be included in HTML text or attribute
- `function HTMLToText(HtmlInput: string): string;`
Converts a HTML content to simple text without any tags
- `function IdnEncode(S: string): string;`
Performs a International Domain Name (IDNA) Punycode encoding
- `function IdnDecode(S: string): string;`
Performs a International Domain Name (IDNA) Punycode decoding
- `function JSEncode(S: string): string;`
Encodes a string for use in a JavaScript string literal
- `function JSEscapeQuote(S: string): string;`
Escape single and double quotes in a JavaScript string literal
- `function TColorToHex(Color: string): string; overload;`
Converts a TColor string value to a hexadecimal string
- `function TColorToHex(Color: TColor): string; overload;`
Converts a TColor value to a hexadecimal string
- `function UrlEncode(S: string): string;`
Performs a URL percent encoding
- `function UrlDecode(S: string): string;`
Performs a URL percent decoding

Var HndStyles

Styles related API methods

- `function GetStyleAsObject(aStyleName: string): TObject;`

Return the specified style's object or nil if not found

- `function GetStyleList(const aIgnoredList: TStringDynArray): THndStyleArray;`
Return the list of style, except those in the ignored list
- `procedure UpdateModifiedTextProperties(aStyle: TObject);`
Update the list of modified text properties in style's ValidTextProperties
- `procedure UpdateModifiedParagraphProperties(aStyle: TObject);`
Update the list of modified paragraph properties in style's ValidParaProperties

Type **THndBuildInfo**

Information about a specific build

- `Id: string;`
Unique identifier of the build.
- `Kind: string;`
Kind of the build: code, chm, epub, html, kindle, pdf, qthelp, word
- `Name: string;`
Name of the build.
- `Enabled: Boolean;`
Is the build enabled ?
- `Order: Integer;`
Order of the build in the list. Greater will be built later.
- `Output: string;`
Output path of the build.
- `Template: string;`
Template name used for this build.

Type **THndBuildInfoArray**

Array of [THndBuildInfo](#)

Var **HndBuilds**

Properties and methods for Builds.

- `function CreateBuild: string;`
Create a new build.
- `procedure DeleteAllBuilds;`
Delete all builds for the current project.
- `procedure DeleteBuild(const aBuildId: string);`
Delete a specific build.
- `function GetBuildEnabled(const aBuildId: string): Boolean;`

Returns true if a build is enabled.

- `function GetBuildFirst: string;`
Returns the first build in the build list.
- `function GetBuildFirstOfKind(const aBuildKind: string): string;`
Returns the first build of the specified kind.
- `function GetBuildKind(const aBuildId: string): string;`
Returns the kind of the build (CHM, HTML...).
- `function GetBuildLast: string;`
Returns the last build in the build list.
- `function GetBuildList: THndBuildInfoArray;`
Get a list of all builds in the project.
- `function GetBuildName(const aBuildId: string): string;`
Returns the name of a specific build.
- `function GetBuildNext(const aBuildId: string): string;`
Get the next build.
- `function GetBuildOrder(const aBuildId: string): Integer;`
Returns the order of a specific build.
- `function GetBuildOutput(const aBuildId: string): string;`
Returns the output path/file of a build.
- `function GetBuildPrevious(const aBuildId: string): string;`
Returns the previous build.
- `function GetBuildTemplate(const aBuildId: string): string;`
Returns the template used by this build.
- `function GetBuildWithName(const aBuildName: string): string;`
Returns the first build with the specified name. This method is case-insensitive.
- `procedure MoveBuildAfter(const aBuildId: string; const aAfterBuildId: string);`
Move a build after another one.
- `procedure MoveBuildBefore(const aBuildId: string; const aBeforeBuildId: string);`
Move a build before another one.
- `procedure MoveBuildFirst(const aBuildId: string);`
Move a build first in the list.
- `procedure MoveBuildLast(const aBuildId: string);`
Move a build last in the list.
- `procedure SetBuildEnabled(const aBuildId: string; const aIsEnabled: Boolean);`
Set the enabled state for a build.
- `procedure SetBuildKind(const aBuildId: string; const aBuildKind: string);`

Sets the kind of the build (CHM, HTML...).

- `procedure SetBuildName(const aBuildId: string; const aName: string);`
Set the name of a specific build.
- `procedure SetBuildOutput(const aBuildId: string; const aOutput: string);`
Set the output path/file of a build.
- `procedure SetBuildTemplate(const aBuildId: string; const aTemplateName: string);`
Set the template used by this build.

Var HndBuildsEx

Additional properties and methods for Builds.

- `function DuplicateBuild(const aBuildId: string): string;`
Duplicate a specific build (including build, meta, status and tags). Returns the new build's ID
- `function GetBuildListFilteredByName(const aOnlyBuildsNamed: TStringList = nil; aExceptBuildsNamed: TStringList = nil): THndBuildInfoArray;`
Returns a list of builds filtered by name (case insensitive)
- `function GetBuildTemplateOrDefault(const aBuildId: string): string;`
Returns the template for a specific build or the default template for this build kind.
- `function GetValidBuildOutput(const aBuildId: string; const aBuildOutput: string): string;`
Returns a build output which is not empty.

Type THndBuildActionInstanceInfo

Information about a specific build action

- `id: string;`
- `name: string;`
- `description: string;`
- `actiontype: string;`
- `content: string;`
- `kind: Integer;`
- `enabled: Boolean;`
- `order: Integer;`

Type THndBuildActionInstanceInfoArray

List of build action info

Var HndBuildsActions

Handle pre- and post- build actions

- `function CreateBuildAction(aBuildId: string): string;`
Create a new action for a specific build. Returns its unique ID
- `procedure DeleteAllBuildActions(aBuildId: string);`
Delete all actions associated with a specific build
- `function DeleteBuildAction(aBuildActionId: string): Boolean;`
Delete a specific build action
- `procedure CopyBuildActionsToBuild(const aSourceBuildId: string;
const aDestBuildId: string);`
Copy all build styles for a specific build to another build
- `function GetBuildActionInfo(aBuildActionId: string):
THndBuildActionInstanceInfo;`
Get a detailed information for a specific build action
- `function GetBuildActionsList(aBuildId: string;
doIncludePreBuildActions: Boolean; doIncludePostBuildActions:
Boolean): THndBuildActionInstanceInfoArray;`
Get a detailed list of all actions for a specific build
- `procedure MoveBuildActionUp(aBuildActionId: string);`
Move a build action up: execute it earlier
- `procedure MoveBuildActionDown(aBuildActionId: string);`
Move a build action down: execute it later
- `procedure MoveBuildActionFirst(aBuildActionId: string);`
Move a build action last: first one executed
- `procedure MoveBuildActionLast(aBuildActionId: string);`
Move a build action last: last one executed
- `procedure MoveBuildActionAfter(const aBuildActionId: string; const
aAfterBuildActionId: string);`
Move a build action after another one.
- `procedure MoveBuildActionBefore(const aBuildActionId: string;
const aBeforeBuildActionId: string);`
Move a build action before another one.
- `function GetBuildActionFirst(aBuildId: string;
doIncludePreBuildActions: Boolean; doIncludePostBuildActions:
Boolean): string;`
Get the first build action for a specific build
- `function GetBuildActionNext(aBuildActionId: string;
doIncludePreBuildActions: Boolean; doIncludePostBuildActions:
Boolean): string;`
Get the build action next to the one specified

- `function GetBuildActionPrevious(aBuildActionId: string;
doIncludePreBuildActions: Boolean; doIncludePostBuildActions:
Boolean): string;`
Get the build action previous to the one specified
- `function GetBuildActionName(aBuildActionId: string): string;`
Get the name of a specific build action
- `function GetBuildActionDescription(aBuildActionId: string):
string;`
Get the description of a specific build action
- `function GetBuildActionContent(aBuildActionId: string): string;`
Get the content of a specific build action
- `function GetBuildActionActionType(aBuildActionId: string): string;`
Get the type of action of a specific build action
- `function GetBuildActionKind(aBuildActionId: string): Integer;`
Get the kind of a specific build action (0: pre-build; 1: post-build)
- `function GetBuildActionEnabled(aBuildActionId: string): Boolean;`
Get the enabled status of a specific build action
- `function GetBuildActionOrder(aBuildActionId: string): Integer;`
Get the order of a specific build action
- `procedure SetBuildActionName(aBuildActionId: string;
aBuildActionName: string);`
Set the name of a specific build action
- `procedure SetBuildActionDescription(aBuildActionId: string;
aBuildActionDescription: string);`
Set the description of a specific build action
- `procedure SetBuildActionContent(aBuildActionId: string;
aBuildActionContent: string);`
Set the content of a specific build action
- `procedure SetBuildActionActionType(aBuildActionId: string;
aBuildActionActionType: string);`
Set the action type of a specific build action
- `procedure SetBuildActionKind(aBuildActionId: string;
aBuildActionKind: Integer);`
Set the kind of a specific build action (0: pre-build; 1: post-build)
- `procedure SetBuildActionEnabled(aBuildActionId: string;
aBuildActionEnabled: Boolean);`
Set the enabled status of a specific build action

Type PHndBuildLibraryOverrideInfo

Pointer to THndBuildLibraryOverrideInfo

Type THndBuildLibraryOverrideInfo

Information about a specific build library override

- `idBuild: string;`
- `idLibraryItem: string;`
- `isEnabled: Boolean;`
- `Properties: string;`

Type THndBuildLibraryOverrideInfoArray

Array of THndBuildLibraryOverrideInfo

Var HndBuildsLibraryOverrides

API functions to manage library overrides for individual builds

- `function GetBuildLibraryOverrideInfo(const idBuild: string; const idLibraryItem: string): THndBuildLibraryOverrideInfo;`
Return build library override info for specific item
- `procedure SetBuildLibraryOverrideInfo(const idBuild: string; const idLibraryItem: string; const isEnabled: Boolean; const Properties: string);`
Set build library override info for specific item
- `procedure SetBuildLibraryOverrideEnabled(const idBuild: string; const idLibraryItem: string; const isEnabled: Boolean);`
Set build library override is enabled for specific item
- `procedure SetBuildLibraryOverrideProperties(const idBuild: string; const idLibraryItem: string; const Properties: string);`
Set build library override properties for specific item
- `procedure DeleteLibraryOverride(const idBuild: string; const idLibraryItem: string);`
Delete library override info, content and source for specific item
- `procedure DeleteAllLibraryOverridesForBuild(const idBuild: string);`
Delete all library overrides info, content and source for a build
- `procedure DeleteAllLibraryOverridesForLibraryItem(const idLibraryItem: string);`
Delete all library overrides info, content and source for a library item
- `function GetBuildLibraryOverrideInfoListForBuild(const idBuild: string): THndBuildLibraryOverrideInfoArray;`
Return list of build library override info for specific build
- `function GetBuildLibraryOverrideInfoListForLibraryItem(const idLibraryItem: string): THndBuildLibraryOverrideInfoArray;`

Return list of build library override info for specific library item

- `function GetBuildLibraryOverrideContent(const idBuild: string; const idLibraryItem: string): TMemoryStream;`

Return content of library override for specific item

- `function GetBuildLibraryOverrideContentChecksum(const idBuild: string; const idLibraryItem: string): string;`

Return content's checksum of library override for specific item

- `procedure SetBuildLibraryOverrideContent(const idBuild: string; const idLibraryItem: string; aContentStream: TMemoryStream);`

Set content of library override for specific item

- `function GetBuildLibraryOverrideSource(const idBuild: string; const idLibraryItem: string): TMemoryStream;`

Return source of library override for specific item

- `function GetBuildLibraryOverrideSourceChecksum(const idBuild: string; const idLibraryItem: string): string;`

Return source's checksum of library override for specific item

- `procedure SetBuildLibraryOverrideSource(const idBuild: string; const idLibraryItem: string; aSourceStream: TMemoryStream);`

Set source of library override for specific item

Var HndBuildsMeta

Access to builds meta data

Var HndBuildsMetaEx

Additional methods related to builds meta data

- `function GetChmButtonVisibilityHex(const aBuildId: string): string;`

Return the CHM button visibility as an hexadecimal string.

- `function GetChmNavigationPaneStyleHex(const aBuildId: string): string;`

Returns the CHM navigation pane style as an Hexadecimal string.

- `function GetProjectDateTimeFormatOverrides(const aBuildId: string; const doIncludeDefaultProjectFormat: Boolean):`

[THndProjectDateTimeFormat](#);

Returns overridden project date / time formats

- `function GetProjectSettingsOverrides(const aBuildId: string; const doIncludeDefaultProjectSettings: Boolean):` [THndProjectSettings](#);

Get project settings overrides for the specified build

- `procedure ResetProjectDateTimeFormatOverrides(const aBuildId: string);`

Reset project date/time overrides for the specified build

- `procedure ResetProjectSettingsOverrides(const aBuildId: string);`
Reset project settings overrides for the specified build
- `procedure SetProjectDateTimeOverrides(const aBuildId: string;
const aDateTimeFormat: THndProjectDateTimeFormat);`
Set the project date time overrides for the specified build
- `procedure SetProjectSettingsOverrides(const aBuildId: string;
const aProjectSettings: THndProjectSettings);`
Set project settings overrides for the specified build

Var HndBuildsStatus

Handle relationship between builds and statuses

- `function AreBuildAndStatusAssociated(const aBuildId: string; const
aStatusId: string): Boolean;`
Indicates whether the specified build and status are associated.
- `function AssociateBuildWithStatus(const aBuildId: string; const
aStatusId: string): Boolean;`
Link a specific build to a specific Status.
- `procedure DissociateAllForStatus(const aStatusId: string);`
Remove any association with the specified status.
- `procedure DissociateAllForBuild(const aBuildId: string);`
Remove any association with the specified builds.
- `function DissociateBuildFromStatus(const aBuildId: string; const
aStatusId: string): Boolean;`
Remove the association between the specified build and status.
- `function GetStatusAssociatedWithBuild(const aBuildId: string):
TStringDynArray;`
Get a list of status Ids associated with a specific build.
- `function GetBuildsAssociatedWithStatus(const aStatusId: string):
TStringDynArray;`
Get a list of build Ids associated with a specific status.

Var HndBuildsStatusEx

Additional methods to handle relationship between builds and statuses

- `function IsTopicIncludedInBuild(const aTopicId: string; const
aBuildId: string): Boolean;`
Returns true if a topic can be included in the specified build (including correct build status).

Type THndBuildStyleInfo

- `Name: string;`
- `Enabled: Boolean;`

- `Content: string;`

Type **THndBuildStyleInfoArray**

Var **HndBuildsStyles**

Handle overridden styles for each builds

- `procedure CopyBuildStylesToBuild(const aSourceBuildId: string; const aDestBuildId: string);`
Copy all build styles for a specific build to another build
- `procedure DeleteAllBuildStyles(const aBuildId: string);`
Delete all build styles for a specific build ID
- `procedure DeleteBuildStyle(const aBuildId: string; const aStyleName: string);`
Delete a specific build's overridden style
- `function GetBuildStyleContent(const aBuildId: string; const aStyleName: string): string;`
Get the content of a specific build's overridden style
- `function GetBuildStyleEnabled(const aBuildId: string; const aStyleName: string): Boolean;`
Get the enabled status of a specific build's overridden style. If not found, returns false.
- `function GetBuildStyleList(const aBuildId: string): THndBuildStyleInfoArray;`
Get a list of overridden styles for a specific build
- `procedure SetBuildStyleContent(const aBuildId: string; const aStyleName: string; const aContent: string);`
Set the content of a specific build's overridden style. Creates it if needed.
- `procedure SetBuildStyleEnabled(const aBuildId: string; const aStyleName: string; const isEnabled: Boolean);`
Set the enabled status of a specific build's overridden style. Creates it if needed.

Var **HndBuildsTags**

Handle relationship between builds and tags

- `function AreBuildAndTagAssociated(const aBuildId: string; const aTagName: string): Boolean;`
Indicates whether the specified build and tag are associated.
- `function AssociateBuildWithTag(const aBuildId: string; const aTagName: string): Boolean;`
Link a specific build to a specific Tag.
- `procedure DissociateAllForTag(const aTagName: string);`
Remove any association with the specified tag.

- `procedure DissociateAllForBuild(const aBuildId: string);`
Remove any association with the specified builds.
- `function DissociateBuildFromTag(const aBuildId: string; const aTagName: string): Boolean;`
Remove the association between the specified build and tag.
- `function GetTagsAssociatedWithBuild(const aBuildId: string): TStringDynArray;`
Get a list of tag ids associated with a specific build.
- `function GetBuildsAssociatedWithTag(const aTagName: string): TStringDynArray;`
Get a list of build ids associated with a specific tag.

Var HndBuildsTagsEx

Additional methods to handle relationship between builds and tags

- `procedure ConditionAdd(aCondition: string; var aConditionList: TStringList);`
Add condition to the list.
- `procedure ConditionInvert(var aConditionList: TStringList);`
Invert condition in the list.
- `function ConditionRemove(aCondition: string; var aConditionList: TStringList): Boolean;`
Remove condition from the list.
- `procedure ConditionReset(var aConditionList: TStringList);`
Reset condition list.
- `function IsTopicIncludedInBuild(const aTopicId: string; const aBuildId: string): Boolean;`
Returns true if a topic can be included in the specified build (including correct build tags).

Type THndDictionaryInfo

Information about a specific dictionary

- `name: string;`
- `description: string;`
- `author: string;`
- `updated: TDate;`
- `localeid: Cardinal;`
- `dictionaryfile: string;`
- `grammarfile: string;`
- `active: Boolean;`

Type THndDictionaryInfoArray

Array of THndDictionaryInfo

Var HndDictionaries

Manage dictionaries and live spell check

- `procedure ClearDictionaries;`
Clears the loaded dictionaries list and disable the spell checker component.
- `procedure DisableAllDictionaries;`
Disable all the dictionaries.
- `function EnableDictionaries(const aLocalIdList: TStringList; const DisabledOthers: Boolean): Boolean;`
Enable a list of dictionaries. Disable the other ones if needed.
- `function EnableDictionary(const aLocaleId: Cardinal; const DisabledOthers: Boolean): Boolean;`
Enable the specific dictionary and disables others if asked.
- `function GetActiveDictionariesCount: Integer;`
Returns the number of currently active dictionaries.
- `function GetActiveDictionariesList: THndDictionaryInfoArray;`
Returns the list of currently active dictionaries.
- `function GetDictionariesCount: Integer;`
Get the number of dictionaries.
- `function GetDictionariesList: THndDictionaryInfoArray;`
Get a list of dictionaries.
- `function GetDictionaryInfo(const aLocalId: Cardinal): THndDictionaryInfo;`
Returns information about the specific locale.
- `function InstallOOoDictionary(const aFileName: string): Boolean;`
Installs a new OpenOffice.org dictionary.
- `procedure PauseLiveSpell;`
Pause live spell in all possible controls.
- `function ReloadDictionaries: Boolean;`
Reload the dictionary list from the disk.
- `procedure SwitchDictionaryStatus(const aLocalId: Cardinal);`
If dictionary is enable, then disable it and vice-versa.
- `procedure UpdateLiveSpell;`
Resume the live spell after it has been paused.

Type THndCaretMovement

Caret movement

- `hcmUp`
- `hcmDown`
- `hcmLeft`
- `hcmRight`
- `hcmTop`
- `hcmBottom`
- `hcmHome`
- `hcmEnd`
- `hcmParaBeginning`
- `hcmNextParaBeginning`

Type **THndVAlign**

Vertical alignment of a picture

- `hndVaBaseline`
- `hndVaMiddle`
- `hndVaAbsTop`
- `hndVaAbsBottom`
- `hndVaAbsMiddle`
- `hndVaLeft`
- `hndVaRight`

Type **THndHVAignment**

Table cell alignment

- `hndHVaTopLeft`
- `hndHVaTopCenter`
- `hndHVaTopRight`
- `hndHVaCenterLeft`
- `hndHVaCenter`
- `hndHVaCenterRight`
- `hndHVaBottomLeft`
- `hndHVaBottomCenter`
- `hndHVaBottomRight`

Type **THndHyperlinkKind**

Kind of hyperlink

- hlkTopic
- hlkNavigation
- hlkInternetMail
- hlkFile
- hlkCounter

Type **THndHyperlinkInfo**

Information about a hyperlink

- Kind: [THndHyperlinkKind](#);
- Param1: string;
- Param2: string;
- Param3: string;
- Initialize(out Dest: [THndHyperlinkInfo](#));

Var **HndEditor**

Create and manage a topic editor

- `procedure ApplyStyleToSelection(const anEditor: TObject; const aStyleId: Integer);`
Apply the specified style to the selection.
- `procedure Clear(const anEditor: TObject);`
Clears the content of the specified editor.
- `procedure ConvertParagraphBreaksToLineBreaks(const anEditor: TObject; doProcessSelectionOnly: Boolean);`
Convert paragraph breaks to soft line breaks
- `function ConvertSelectionToSnippet(const anEditor: TObject; const doReplaceSelection: Boolean; var nbItemsConverted: Integer; var nbItemsDeleted: Integer): string;`
Convert the currently selected content to a snippet and return it ID. Replace the selection by the snippet if doReplaceSelection is True
- `function CreateTemporaryEditor: TObject;`
Creates a new temporary editor.
- `function CreateTemporaryReportHelper: TObject;`
Create a new temporary report helper.
- `function CreateTemporaryViewer: TObject;`
Creates a new temporary viewer.
- `procedure DestroyTemporaryEditor(const anEditor: TObject);`
Destroys a previously created temporary editor.
- `procedure DestroyTemporaryReportHelper(const aReportHelper: TObject);`

Destroys a previously created temporary report helper.

- `procedure DestroyTemporaryViewer(const aViewer: TObject);`
Destroys a previously created temporary viewer.
- `function GetAnchorList(const anEditor: TObject): TStringList;`
Retrieves the list of anchors in the specified editor.
- `function GetContentAsHtml(const anEditor: TObject; out aCssContent: string): string;`
Returns the editor content as HTML.
- `procedure GetContentAsStream(const anEditor: TObject; aStream: TMemoryStream);`
Returns the editor content as Stream.
- `function GetContentAsText(const anEditor: TObject): string;`
Returns the editor content as text.
- `function GetCurrentAnchorName(const anEditor: TObject): string;`
Returns the name of the current checkpoint or an empty string if there isn't any.
- `function GetCurrentItem(const anEditor: TObject): TObject;`
Returns the currently selected item.
- `function GetCurrentPictureAltText(const anEditor: TObject): string;`
Return the currently selected picture's alternative text.
- `function GetCurrentPictureBackColor(const anEditor: TObject): TColor;`
Returns the background color of the currently selected picture.
- `function GetCurrentPictureHeight(const anEditor: TObject; const DoConvertTo96Dpi: Boolean): Integer;`
Returns the currently selected picture's height.
- `function GetCurrentPictureMarginLeftRight(const anEditor: TObject): Integer;`
Get the current picture left/right margins.
- `function GetCurrentPictureMarginTopBottom(const anEditor: TObject): Integer;`
Get the current picture top/bottom margins.
- `function GetCurrentPicturePadding(const anEditor: TObject): Integer;`
Get the current picture padding.
- `function GetCurrentPictureVAlign(const anEditor: TObject): THndVAlign;`
Return the currently selected picture's vertical alignment.
- `function GetCurrentPictureWidth(const anEditor: TObject; const DoConvertTo96Dpi: Boolean): Integer;`
Returns the currently selected picture's width.

- `function GetUsedLibraryItems(const anEditor: TObject): TStringList;`
Returns a list of library items used in that document
- `procedure InsertAnchorBeforeCurrentItem(const anEditor: TObject; const aName: string);`
Inserts a checkpoint at the current cursor position.
- `procedure InsertCondition(const anEditor: TObject; const anOperation: string; const aCondition: string);`
Insert a condition item at the current cursor position.
- `procedure InsertContentFromHTML(anEditor: TObject; aHtmlContent: string);`
Insert HTML content in the editor
- `function InsertFile(anEditor: TObject; aFile: string): Boolean;`
Inserts a file in the editor.
- `function InsertFileFromLibraryItem(const anEditor: TObject; const aLibraryItem: string; const doImportNewImagesToLibrary: Boolean): Boolean;`
Insert the content of the library item.
- `procedure InsertHyperLinkToTopicId(const anEditor: TObject; const aText: string; const aLinkedTopicId: string; const aLinkedAnchor: string = '');`
Inserts an hyperlink to a specific topic ID (and optionally anchor in that topic) using the provided caption.
- `procedure InsertHyperLinkToUrl(const anEditor: TObject; const aText: string; const anUrl: string);`
Inserts an hyperlink to a specific URL.
- `function InsertLibraryItem(const anEditor: TObject; const aLibraryItem: string; const OnlyReplaceContent: Boolean): TObject;`
Insert a library item object in the specified editor at the current cursor position.
- `function InsertLibraryItemContent(const anEditor: TObject; const aLibraryItem: string): Boolean;`
Insert the content of a library item in the specified editor at the current cursor position.
- `procedure InsertPageBreakBeforeCurrentItem(const anEditor: TObject);`
Inserts a page break at the current cursor position.
- `function InsertStream(const anEditor: TObject; const aStream: TStream): Boolean;`
Insert the content of the specified stream in the specified editor.
- `procedure InsertTopicContent(const anEditor: TObject; const aTopicId: string);`
Insert the content of the specified topic in the specified editor.
- `function IsEmpty(const anEditor: TObject): Boolean;`

Returns true if the editor is empty

- `procedure MoveCaret(anEditor: TObject; aDirection: THndCaretMovement);`
Move the editor's caret to the specified direction
- `procedure MoveCaretTo(anEditor: TObject; X: Integer; Y: Integer);`
Move the specified editor's caret to another location.
- `procedure MoveCarretToEnd(const anEditor: TObject);`
Move the specified editor's caret to the end.
- `procedure ProcessConditionalsForCurrentBuild(const anEditor: TObject);`
Removes any conditional items based on current build setting.
- `procedure RemoveCurrentAnchor(const anEditor: TObject);`
Remove the current checkpoint in the specified Editor.
- `function ReplaceLibraryItems(const anEditor: TObject): Boolean;`
Replace the library items by their actual content.
- `procedure SetAsTopicContent(const anEditor: TObject; const aTopicId: string);`
Set the specific topic's content as the current editor's content.
- `function SetContent(const anEditor: TObject; const aContentStream: TStream): Boolean;`
Set the content from a stream.
- `procedure SetCurrentCellsAlignment(const anEditor: TObject; const anHVAAlignment: THndHVAAlignment);`
Sets the currently selected cells horizontal and vertical alignment.
- `procedure SetCurrentPictureAltText(const anEditor: TObject; const anAltText: string);`
Set the currently selected picture's alternative text.
- `procedure SetCurrentPictureBackColor(const anEditor: TObject; const aColor: TColor);`
Set the currently selected picture's background color.
- `procedure SetCurrentPictureHeight(const anEditor: TObject; const aNewHeight: Integer; const doConstrainProportions: Boolean);`
Defines the currently selected picture's height.
- `procedure SetCurrentPictureMarginLeftRight(const anEditor: TObject; const aNewMargin: Integer);`
Set the currently selected picture's left/right margins.
- `procedure SetCurrentPictureMarginTopBottom(const anEditor: TObject; const aNewMargin: Integer);`
Set the currently selected picture's top/bottom margins.
- `procedure SetCurrentPicturePadding(const anEditor: TObject; const aNewSpacing: Integer);`

Set the currently selected picture's padding.

- `procedure SetCurrentPictureVAlign(const anEditor: TObject; const aNewVAlign: THndVAlign);`

Defines the currently selected picture's vertical alignment.

- `procedure SetCurrentPictureWidth(const anEditor: TObject; const aNewWidth: Integer; const doConstrainProportions: Boolean);`

Defines the currently selected picture's width.

- `procedure SyntaxHighlightSelection(const anEditor: TObject; const aLanguageCode: string);`

Syntax highlight the selected content based on the language

- `procedure TogglePageBreak(const anEditor: TObject);`

Create or remove the current page break.

- `procedure UpdateLibraryItem(const anEditor: TObject; anItemId: string);`

Update a library item's visual appearance in the editor.

Var HndEditorHelper

Additional methods to manage a topic editor

- `procedure CleanContent(const anEditor: TObject);`
Cleans the content of the editor by removing any un-needed items. This will not alter the content of the editor but could make its size on disk/memory smaller
- `procedure GetCurrentStyleTemplateInfo(const anEditor: TObject; var aStyleTemplateId: Integer; var aStyleTemplateName: string);`
Returns information about the current style template.
- `function GetHyperlinkDetailedTextFromString(const aString: string): string;`
Returns a detailed text from an hyperlink string.
- `function GetHyperlinkInfoFromString(const aString: string): THndHyperlinkInfo;`
Extract the hyperlink information from a string.
- `procedure GetHyperlinkTargetExtraFromString(const aString: string; var Target: string; var Extras: string);`
Returns the Target and Extra values from hyperlink data.
- `procedure ImportImagesToLibrary(const anEditor: TObject);`
Import the images to library.
- `procedure ReplaceAnchorsToLowercase(const anEditor: TObject);`
Modify all anchors to lowercase.
- `procedure ScrollToSelection(const anEditor: TObject);`
Scrolls to make selection visible
- `function SetHyperlinkInfoToString(const anHyperLinkInfo:`

[THndHyperlinkInfo](#)): string;

Constructs a string based in hyperlink info.

- procedure SetupEditorProperties(const anEditor: TObject);
Defines default editor properties and events.

Type **THndGeneratorInfo**

Information and actions over the current generation

- BOMOutput: Boolean;
Add BOM to currently generated file
- ForceOutputEncoding: Boolean;
Force output encoding for the currently generated file
- HelpNDocVersion: string;
Current HelpNDoc version
- OutputDir: string;
Output directory of the current generator
- OutputFile: string;
Output file name for the current generator
- function GetAssetsList: TStringDynArray;
Returns a list of template assets
- function GetCustomSettingValue(const aCustomSetting: string): Variant;
Returns the user-defined value of a custom setting.
- function GetGeneratedFiles: TStringDynArray;
List of files that have been generated so far.
- property CurrentBuildId: string;
Current build Id being executed.
- property CurrentFile: string;
Currently generated file
- property CurrentTopic: string;
Current topic ID which is being worked on.
- property LogInfoCount: Integer;
How many INFO log occurred while generating.
- property LogWarningCount: Integer;
How many WARNING log occurred while generating.
- property LogErrorCount: Integer;
How many ERROR log occurred while generating.
- property OutputDirLib: string;
Output sub-directory where library items are generated. Always a valid HTML path
- property TemplateInfo: THndTemplateInfo;

Information about the currently used template.

Var HndJsSearchEngine

Methods to manage the JavaScript search engine

- `procedure AddSearchData(const aSearchData: string; const aAssociatedTopicId: string; const nMinWordLength: Integer = 3);`
Add search data and associate it to a specific topic.
- `procedure ClearSearchData;`
Clear the current search data.
- `function GetJsData: string;`
Get the JavaScript search data.

Type THndKeywordsAttachMode

Specify how the moved keyword will be attached: - `hkamAdd`: Adds a node at the same level as the existing node. - `hkamAddChild`: Adds a child node to the existing node.

- `hkamAdd`
- `hkamAddFirst`
- `hkamAddChild`
- `hkamAddChildFirst`
- `hkamInsert`

Type THndKeywordsInfo

Minimal keyword information. Used by the `THndKeywordsInfoArray`

- `Id: string;`
- `Caption: string;`
- `ParentId: string;`

Type THndKeywordsInfoArray

Array of minimal keyword information

Var HndKeywords

Properties and methods for keywords

- `function CreateKeyword: string;`
Create a new keyword. The new keyword will be placed at the bottom of the list.
- `procedure DeleteAllKeywords;`
Delete all the keywords in the project, except the root project keyword.
- `function DeleteKeyword(const aKeywordId: string): Boolean;`

Delete a specific keyword and its children.

- `function GenerateUniqueCaption(anInitialCaption: string; aParentId: string; const aFilteredItems: array of string): string;`
Generates a unique caption within the specified parent.
- `function GetKeywordByCaption(aCaption: string; aParentId: string): string;`
Returns the ID of the keyword with the specified case-insensitive caption with the specified parent.
- `function GetKeywordCaption(const aKeywordId: string): string;`
Get the caption of a specific keyword.
- `function GetKeywordDirectChildrenCount(const aKeywordId: string): Integer;`
Returns the number of children for the specified keyword.
- `function GetKeywordDirectChildrenList(const aParentId: string): THndKeywordsInfoArray;`
Returns a list of all the children keywords.
- `function GetKeywordLevel(const aKeywordId: string): Integer;`
Returns the level of the specified keyword.
- `function GetKeywordList(const aIncludingProjectKeyword: Boolean): THndKeywordsInfoArray;`
Returns a list of all the keywords.
- `function GetKeywordNext(const aKeywordId: string): string;`
Get the next keyword in line. Could be a child or the next keyword.
- `function GetKeywordNextSibling(const aKeywordId: string): string;`
Returns the next sibling of a specific keyword.
- `function GetKeywordParent(const aKeywordId: string): string;`
Returns the parent keyword of a specific keyword.
- `function GetKeywordPreviousSibling(const aKeywordId: string): string;`
Get the previous sibling keyword.
- `function GetProjectKeyword: string;`
Returns the root project keyword.
- `function MoveKeyword(const aKeywordId: string; const aReferencedKeywordId: string; const oAttachMode: THndKeywordsAttachMode): Boolean;`
Move the keyword to a new position in reference to nReferencedKeywordsId.
- `function MoveKeywordLeft(const aKeywordId: string): Boolean;`
Move the specific keyword left in the hierarchy.
- `function MoveKeywordRight(const aKeywordId: string): Boolean;`
Move the specific keyword right in the hierarchy.

- `function SetKeywordCaption(const aKeywordId: string; const sNewCaption: string): string;`
Defines the specific keyword' caption.

Var HndKeywordsMeta

Access to topics meta data

Type THndLibraryItemAttachMode

Specify how the moved library item will be attached: - `hlamAdd`: Adds a node at the same level as the existing node and makes the new node last. - `hlamAddFirst`: Adds a node at the same level as the existing node and makes the new node first. - `hlamAddChild`: Adds a child node to the existing node and makes the new node last. - `hlamAddChildFirst`: Adds a child node to the existing node and makes the new node first. - `hlamInsert`: Inserts a node at the same level just before the existing node.

- `hlamAdd`
- `hlamAddFirst`
- `hlamAddChild`
- `hlamAddChildFirst`
- `hlamInsert`

Type THndLibraryItemsInfo

Minimal library item information. Used by the `THndLibraryItemsInfoArray`

- `Id: string;`
- `Caption: string;`
- `Extension: string;`
- `Kind: Integer;`
- `Source: Integer;`

Type THndLibraryItemsInfoArray

Array of minimal library item information

Var HndLibraryItems

Properties and methods for library items

- `function CreateItem: string;`
Creates a new unspecified item to the library.
- `function DeleteItem(const anItemId: string): Boolean;`
Delete a specific library item.
- `function GetItemByCaption(const aCaption: string): string;`

Returns the item with the given caption.

- `function GetItemCaption(const anItemId: string): string;`
Gets the caption of a specific item.
- `function GetItemContent(const anItemId: string): TMemoryStream;`
Get the content of the item as a stream. Caller must free the stream after using it.
- `function GetItemContentAsText(const anItemId: string): string;`
Get the content of an item as a text.
- `function GetItemContentChecksum(const anItemId: string): string;`
Returns the checksum (MD5) of a library item content. Useful for comparison purposes.
- `function GetItemExtension(const anItemId: string): string;`
Get the item's file extension.
- `function GetItemIsLocked(const anItemId: string; out aLockUser: string; out aLockDateTime: TDateTime): Boolean;`
Returns True if a library item is locked, as well as lock data
- `function GetItemKind(const anItemId: string): Integer;`
Gets the kind of the specific item (1 => picture, 2 => movie, 3 => document, 4 => system variable, 5 => variable, 6 => HTML, 7 => snippet, 8 => folder, 9 => image map, 10 => bar code, 11 => counter).
- `function GetItemList(const aIncludingKinds: array of Integer): THndLibraryItemsInfoArray;`
Returns a list of items filtered by `aIncludingKinds`. For example `GetItemList([1,2])` lists only images and movies. Leave blank to list everything.
- `function GetItemParent(const anItemId: string): string;`
Returns the parent item of a specific item.
- `function GetItemSource(const anItemId: string): Integer;`
Gets the source of a specific library item (1 => included, 2 => external file, 3 => URL).
- `function GetItemSourceContent(const anItemId: string): TMemoryStream;`
Get the source content of the item as a stream. Caller must free the stream after using it.
- `function GetItemSourceContentChecksum(const anItemId: string): string;`
Returns the checksum (MD5) of a library item's source content. Useful for comparison purposes.
- `function GetItemsWithSameContent(const anItemId: string; const aFilteredItems: array of string): THndLibraryItemsInfoArray;`
Returns the items with the same content as the specified one.
- `function GetItemUrlFile(const anItemId: string): string;`
Gets the URL File of a specified library item.
- `function GetItemUrlFileAbsolute(const anItemId: string): string;`
Gets the URL of a specified library item: use project path if relative.

- `function GetItemUrlLink(const anItemId: string): string;`
Gets the URL Link of a specified library item.
- `function GetProjectItem: string;`
Returns the root project library item.
- `function MoveItem(const anItemId: string; const aReferencedItemId: string; const oAttachMode: THndLibraryItemAttachMode): Boolean;`
Move the library item to a new position in reference to aReferencedItemId.
- `function SetItemCaption(const anItemId: string; const aCaption: string): string;`
Sets the caption of a specific item.
- `function SetItemContent(const anItemId: string; const aContent: TStream): Boolean;`
Sets the content stream of a specified item.
- `function SetItemContentFromBytes(const anItemId: string; const aContent: TBytes): Boolean;`
Sets the content of the item from an array of Bytes.
- `function SetItemContentFromFile(const anItemId: string; const aContentFile: string): Boolean;`
Set the content of the item from an existing file.
- `function SetItemContentFromText(const anItemId: string; const aContentText: string): Boolean;`
Sets the content as text of a specific item.
- `function SetItemExtension(const anItemId: string; const anExtension: string): Boolean;`
Set the item's file extension.
- `function SetItemKind(const anItemId: string; const aKind: Integer): Boolean;`
Sets the kind of a specific item. See `GetItemKind` for kind value
- `function SetItemSource(const anItemId: string; const aSource: Integer): Boolean;`
Sets the source of a specific library item (1 => included, 2 => external file, 3 => URL).
- `function SetItemSourceContent(const anItemId: string; const aContent: TStream): Boolean;`
Sets the source content stream of a specified item.
- `function SetItemUrlFile(const anItemId: string; const anUrlFile: string): Boolean;`
Sets the URL File of a specified library item.
- `function SetItemUrlLink(const anItemId: string; const anUrlLink: string): Boolean;`
Sets the URL Link of a specified library item.

Var HndLibraryItemsEx

Additional properties and methods for library items

- `function FilterCaptionString(const aCaption: string): string;`
- `function GetItemContentGraphic(const anItemId: string; anEditorDisplayWidth: Integer = 0; anEditorDisplayHeight: Integer = 0): TGraphic;`
- `function SetItemContentGraphic(const anItemId: string; const aGraphic: TGraphic): Boolean;`
- `procedure SetVariableValueByName(const aVariableName: string; const aVariableValue: string);`

Var HndLibraryItemsMeta

Access to library items meta data

Type THndProjectSettings

Various project settings

- `Title: string;`
- `Author: string;`
- `DefaultTopic: string;`
- `Version: string;`
- `Copyright: string;`
- `Summary: string;`
- `Comment: string;`
- `Language: Integer;`
- `Charset: Integer;`

Var HndProjects

Properties and methods for projects

- `function BackupProject(const aBackupLocation: string): Boolean;`
Backup the project to the specified location
- `procedure CloseProject;`
Closes the currently opened project.
- `function CopyProject(const aNewProjectName: string; const OpenNewOne: Boolean): Boolean;`
Copy the project to a new location and open the new one if needed.
- `function DeleteProject: Boolean;`
Physically delete the currently opened project.
- `function GetProjectAuthor: string;`
Returns the author of the project.

- `function GetProjectBusy: Boolean;`
Project is currently busy: creating, loading or closing.
- `function GetProjectCharset: Integer;`
Returns the project current charset.
- `function GetProjectCharsetAsHtml: string;`
Returns the project current charset as HTML charset value.
- `function GetProjectClosing: Boolean;`
Project is currently closing.
- `function GetProjectComment: string;`
Return the current project's comment.
- `function GetProjectCopyright: string;`
Returns the project copyright.
- `function GetProjectCreating: Boolean;`
Project is currently being created.
- `function GetProjectCssContent: string;`
Returns the CSS content of the current project. Can only be returned when generating an HTML related format.
- `function GetProjectDefaultTopic: string;`
Returns the Id of the default topic.
- `function GetProjectId: string;`
Returns the currently open project id.
- `function GetProjectIsOpen: Boolean;`
Returns True if a project is open.
- `function GetProjectLanguage: Integer;`
Returns the project current language as a Local ID (LCID).
- `function GetProjectLanguageCode: string;`
Returns the project language code (eg: en-us)
- `function GetProjectLanguageCodeAsHtml(const ShortFormat: Boolean): string;`
Returns the project language code as HTML (eg: en)
- `function GetProjectModified: Boolean;`
Indicates whether or no the current project has been modified since last save.
- `function GetProjectName: string;`
Returns the current project name (or file name).
- `function GetProjectNeverSaved: Boolean;`
Indicates whether the project as already been saved or not.
- `function GetProjectOpenning: Boolean;`
Returns True if the project is currently opening.
- `function GetProjectSummary: string;`

Returns the project summary.

- `function GetProjectTitle: string;`
Gets the title of the specified project.
- `function GetProjectVersion: string;`
Return the current project's version number.
- `function ImportTableOfContentsForProject(const aOtherProjectName: string): string;`
Import the table of contents of another proeject
- `function LockResource(const aResourceType: HndResourceType; const aResourceId: string; const doForceLock: Boolean): Boolean;`
Lock a specific resource
- `function MoveProject(const aNewProjectName: string): Boolean;`
Move the project to a new location.
- `function NewProject(const aProjectName: string = ''): string;`
Creates a new project and returns its unique id.
- `function OpenProject(const aProjectName: string; const doApplyMigrations: Boolean): string;`
Open an existing project and returns its unique id.
- `procedure SaveProject;`
Saves the project to a file.
- `procedure SetProjectAuthor(const anAuthor: string);`
Defines the project author.
- `procedure SetProjectCharset(const aCharSet: Integer);`
Defines the project charset.
- `procedure SetProjectComment(const aComment: string);`
Set the current project comment.
- `procedure SetProjectCopyright(const aCopyrightValue: string);`
Defines the project copyright.
- `function SetProjectDefaultTopic(const aDefaultTopic: string): string;`
Defines the project's default topic.
- `procedure SetProjectLanguage(const aLanguage: Integer);`
Defines the project language.
- `procedure SetProjectModified(const IsModified: Boolean);`
Mark the current project as being modified since last save.
- `procedure SetProjectNeverSaved(const IsProjectNeverSaved: Boolean);`
Mark the project as being never saved.
- `procedure SetProjectSummary(const aSummary: string);`
Set the project summary.

- `procedure SetProjectTitle(const aProjectTitle: string);`
Sets the title of the specified project.
- `procedure SetProjectVersion(const aProjectVersion: string);`
Sets the version of the current project.
- `procedure UnlockResource(const aResourceType: HndResourceType;
const aResourceId: string);`
Unlock a specific resource
- `function VacuumProject: Boolean;`
Clean the project and lower its file size by removing unneeded information.

Var HndProjectsEx

Additional properties and methods for projects.

- `function GetProjectSettings: THndProjectSettings;`
- `procedure SetProjectSettings(const aProjectSettings:
THndProjectSettings);`

Type THndProjectDateTimeFormat

Date and Time format for a project

- `CurrentDateTime: string;`
- `Date: string;`
- `DateLong: string;`
- `Day: string;`
- `DayLong: string;`
- `Month: string;`
- `MonthLong: string;`
- `Time: string;`
- `TimeLong: string;`
- `Year: string;`
- `YearLong: string;`

Var HndProjectsMeta

Access to project meta data

Var HndProjectsMetaEx

Additional methods to access project meta data

- `function GetProjectDateTimeFormat: THndProjectDateTimeFormat;`
Returns project date / time formats
- `procedure SetProjectDateTimeFormat(const aDateTimeFormat:`

[THndProjectDateTimeFormat](#));

Set project date / time formats

Var HndSecrets

Properties and methods for Secrets

- `procedure DeleteSecret(const aKey: string);`
Delete a secret and its value
- `function GetSecret(const aKey: string; const aPassword: string = ''): string;`
Get a secret's value. Value will be decrypted using the password specified, or HelpNDoc's internal password. Returns an empty string is the password is incorrect
- `procedure SetSecret(const aKey: string; const aValue: string; const aPassword: string = '');`
Set a secret value. Value will be encrypted using the password specified, or HelpNDoc's internal password (less secured)

Type THndStatusInfo

Information about a specific status

- `Id: string;`
- `Caption: string;`
- `Color: TColor;`
- `Order: Integer;`

Type THndStatusInfoArray

Array of THndStatusInfo

Var HndStatus

Properties and methods for Statuses

- `function CreateStatus: string;`
Create a new status
- `procedure DeleteAllStatus;`
Delete all statuses
- `function DeleteStatus(const aStatusId: string): Boolean;`
Delete a specific status
- `function GetStatusByCaption(const aCaption: string): string;`
Returns a status ID based on its caption
- `function GetStatusCaption(const aStatusId: string): string;`
Get the caption of a status

- `function GetStatusColor(const aStatusId: string): TColor;`
Get the color of a status
- `function GetStatusCount: Integer;`
Return the number of available statuses.
- `function GetStatusFirst: string;`
Returns the first status in the statuses list.
- `function GetStatusList: THndStatusInfoArray;`
Get a list of statuses.
- `function GetStatusNext(const aStatusId: string): string;`
Get the next status.
- `function GetStatusOrder(const aStatusId: string): Integer;`
Returns the order of a specific status.
- `function GetStatusPrevious(const aStatusId: string): string;`
Returns the previous status.
- `procedure MoveStatusAfter(const aStatusId: string; const aAfterStatusId: string);`
Move a status after another one.
- `procedure MoveStatusBefore(const aStatusId: string; const aBeforeStatusId: string);`
Move a status before another one.
- `procedure MoveStatusFirst(const aStatusId: string);`
Move the status first in the list.
- `procedure MoveStatusLast(const aStatusId: string);`
Move the status last in the list.
- `procedure SetStatusCaption(const aStatusId: string; const aCaption: string);`
Set the caption of a status.
- `procedure SetStatusColor(const aStatusId: string; const aColor: TColor);`
Set the color of a status.

Var HndStyles

Properties and methods for Styles

Var HndTags

Properties and methods for Tags

- `function CreateTag(aTagName: string): string;`
Create a new tag. Returns the created tag name.
- `procedure DeleteAllCustomTags;`

Delete all custom tags for the current project.

- `procedure DeleteCustomTag(aTagName: string);`
Delete a specific custom tag.
- `function GetTagCount(const doIncludeSystemTags: Boolean): Integer;`
Returns the number of tags, including or not system tags.
- `function GetTagListAll: TStringList;`
Returns a list of all system and custom tags.
- `function GetTagListCustom: TStringList;`
Get a list of custom tags.
- `function GetTagListSystem: TStringList;`
Get a list of system tags.
- `function IsSystemTag(aTagName: string): Boolean;`
Returns True if the specified tag is a system tag

Var HndTemplates

Properties and methods for Templates

- `function GetDefaultTemplateFor(const aTemplateKind: string): THndTemplateInfo;`
Get the default template info for the specified kind.
- `function GetTemplateCategoryHierarchy: THndTemplateHierarchyArray;`
Returns the full category hierarchy of available templates.
- `function GetTemplateFromName(const aTemplateName: string; const aTemplateKind: string): THndTemplateInfo;`
Returns a specific template based on its name and kind.
- `function GetTemplateList: THndTemplateInfoArray;`
Returns a list of available templates on specified kind, or all kinds if none specified.
- `procedure UpdateTemplateList;`
Retrieves the templates from the hard drive.

Var HndTemplatesEx

Additional properties and methods for Templates

- `function CopyAndMergeTemplateToFolder(const aTemplateInfo: THndTemplateInfo; const aFolder: string): Boolean;`
Copy the template to the specific folder, merging its info file with existing template in that folder.
- `function GetTemplateInfoFromPath(const aTemplatePath: string): THndTemplateInfo;`
Get template info from its path.
- `function GetTemplateKindFromPath(const aTemplatePath: string):`

```
string;
```

Returns the kind of template available at the specified path.

- `function GetTemplateRawNameFromPath(aTemplatePath: string): string;`

Returns the template directory name from path.

Type **THndTopicsAttachMode**

Specify how the moved topic will be attached: - `htamAdd`: Adds a node at the same level as the existing node and makes the new node last. - `htamAddFirst`: Adds a node at the same level as the existing node and makes the new node first. - `htamAddChild`: Adds a child node to the existing node and makes the new node last. - `htamAddChildFirst`: Adds a child node to the existing node and makes the new node first. - `htamInsert`: Inserts a node at the same level just before the existing node.

- `htamAdd`
- `htamAddFirst`
- `htamAddChild`
- `htamAddChildFirst`
- `htamInsert`

Var **HndTopics**

Create, edit and manage topics within the current project.

- `procedure CopyTopicToClipboard(const aTopicId: string; const isCut: Boolean);`
Copy the specified topic to clipboard.
- `procedure CreateMultipleTopics(const aTopicList: TStrings; const aParentTopic: string = ''; const aGeneratedIds: TStrings = nil);`
Creates multiple child topics of the specified parent based on a tabular lists.
- `function CreateTopic: string;`
Create a new topic. The topic will be placed at the bottom of the topic list.
- `procedure DeleteAllTopics;`
Delete all the topics in the project, except the parent topic.
- `function DeleteTopic(const aTopicId: string): Boolean;`
Delete a specific topic and its children. Project topic can't be deleted.
- `function GenerateUniqueHelpContext(const aBaseHelpContext: Integer; const aFilteredTopics: array of string): Integer;`
Generates a unique help context among all the topics except the filtered ones.
- `function GenerateUniqueHelpId(const aBaseHelpId: string; const aFilteredTopics: array of string): string;`
Generates a unique help id among all the topics except the filtered ones.

- `function GetCurrentTopic: string;`
Returns the ID of the currently edited topic.
- `function GetProjectTopic: string;`
Returns the root project topic.
- `function GetTopicCaption(const aTopicId: string): string;`
Get the caption of a specific topic.
- `function GetTopicContent(const aTopicId: string): TMemoryStream;`
Returns the content of the specified topic.
- `function GetTopicContentChecksum(const aTopicId: string): string;`
Returns the checksum (MD5) of a topic's content. Useful for comparison purposes.
- `function GetTopicContentAsHtml(const aTopicId: string): string;`
Get the topic's content as HTML. Only available when generating an HTML related documentation.
- `function GetTopicCount: Integer;`
Returns the number of topics available in the current project.
- `function GetTopicCreationDateTime(const aTopicId: string): TDateTime;`
Returns the date and time when the topics was created.
- `function GetTopicDescription(const aTopicId: string): string;`
Returns the description of the topic
- `function GetTopicDirectChildrenCount(const aParentId: string): Integer;`
Returns the number of direct children a topic has.
- `function GetTopicDirectChildrenList(const aParentId: string): THndTopicsInfoArray;`
Returns a list of direct children of the specified topic.
- `function GetTopicFirst: string;`
Returns the first topic in the project.
- `function GetTopicFooterKind(const aTopicId: string): Integer;`
Returns the topic's kind of footer text: normal, custom, hidden.
- `function GetTopicFooterText(const aTopicId: string): string;`
Returns the custom topic footer text.
- `function GetTopicFooterTextCalculated(const aTopicId: string): string;`
Returns the footer text based on the header kind: project caption, custom text or empty text for hidden footer.
- `function GetTopicHeaderKind(const aTopicId: string): Integer;`
Returns the topic's kind of header text: normal, custom, hidden.
- `function GetTopicHeaderText(const aTopicId: string): string;`
Returns the custom topic header text.

- `function GetTopicHeaderTextCalculated(const aTopicId: string): string;`
Returns the header text based on the header kind: topic title, custom text or empty text for hidden header.
- `function GetTopicHelpContext(const aTopicId: string): Integer;`
Get a specific topic's help context.
- `function GetTopicHelpId(const aTopicId: string): string;`
Get a specific topic's help id.
- `function GetTopicIconIndex(const aTopicId: string): Integer;`
Get the icon index for a specific topic.
- `function GetTopicIndex(const aTopicId: string; const isZeroBased: Boolean): Integer;`
Returns the index of the topic from its parent.
- `function GetTopicIndexHierarchy(const aTopicId: string; const isZeroBased: Boolean): THndIntegerArray;`
Returns an array representing the topic's index hierarchy. Ex: [1,0,3,4] for 1.0.3.4.
- `function GetTopicIsLocked(const aTopicId: string; out aLockUser: string; out aLockDateTime: TDateTime): Boolean;`
Returns True if a topic is locked, as well as lock data
- `function GetTopicKind(const aTopicId: string): Integer;`
Returns the kind of the specific topic (0: normal; 1: empty; 2: Link to URL; 3: Link to file).
- `function GetTopicLast: string;`
Returns the last topic in the project.
- `function GetTopicLevel(const aTopicId: string): Integer;`
Returns the level of the specified topic.
- `function GetTopicList(const aIncludingProjectTopic: Boolean): THndTopicsInfoArray;`
Returns a list of topics and their associated ID as object.
- `function GetTopicListWithCaption(const aCaption: string; const isPartialCaption: Boolean): THndTopicsInfoArray;`
Returns a list of topics with a specific caption.
- `function GetTopicModificationDateTime(const aTopicId: string): TDateTime;`
Returns the date and time when the topic was last modified.
- `function GetTopicNext(const aTopicId: string): string;`
Get the next topic in line. Could be a child or a sibling topic.
- `function GetTopicNextSibbling(const aTopicId: string): string;`
Get the next sibling topic.
- `function GetTopicOrder(const aTopicId: string): Integer;`
Returns the order of the topic based on sibling topics.

- `function GetTopicParent(const aTopicId: string): string;`
Returns the parent topic of a topic.
- `function GetTopicPrevious(const aTopicId: string): string;`
Returns a list of topics and their associated ID as object.
- `function GetTopicPreviousSibling(const aTopicId: string): string;`
Get the previous sibling topic.
- `function GetTopicStatusId(const aTopicId: string): string;`
Get the status id of a topic
- `function GetTopicUrlFile(const aTopicId: string): string;`
Returns the topic's URL file property.
- `function GetTopicUrlLink(const aTopicId: string): string;`
Returns the topic's URL link property.
- `function GetTopicVisibility(const aTopicId: string): integer;`
Returns the topic's visibility.
- `function MoveTopic(const aTopicId: string; const
aReferencedTopicId: string; const oAttachMode:
THndTopicsAttachMode): Boolean;`
Move the topic to a new position in reference to nReferencedTopicsId.
- `function MoveTopicDown(const aTopicId: string): Boolean;`
Move the specific topic bellow the next sibling.
- `function MoveTopicLeft(const aTopicId: string): Boolean;`
Move the specific topic left in the hierarchy.
- `function MoveTopicRight(const aTopicId: string): Boolean;`
Move the specific topic right in the hierarchy.
- `function MoveTopicUp(const aTopicId: string): Boolean;`
Move the specific topic above the previous sibling.
- `procedure PasteTopicFromClipboard(aParentId: string);`
Paste the topic from clipboard as a child of the parent specified or as a child of the project topic.
- `function SetCurrentTopic(const aTopicId: string): Boolean;`
Defines the currently selected topic.
- `procedure SetTopicCaption(const aTopicId: string; const
sNewCaption: string);`
Defines the specific topic's caption.
- `procedure SetTopicContent(const aTopicId: string; const
aContentStream: TStream);`
Set the topic's content.
- `procedure SetTopicDescription(const aTopicId: string; const
aDescription: string);`
Set the topic's description.

- `procedure SetTopicFooterKind(const aTopicId: string; const aFooterKind: Integer);`
Sets the topic footer kind.
- `function SetTopicFooterText(const aTopicId: string; const aFooterText: string): string;`
Sets the custom text for the topic footer.
- `procedure SetTopicHeaderKind(const aTopicId: string; const anHeaderKind: Integer);`
Sets the topic header kind.
- `function SetTopicHeaderText(const aTopicId: string; const anHeaderText: string): string;`
Sets the custom text for the topic header.
- `function SetTopicHelpContext(const aTopicId: string; const aHelpContext: Integer): Integer;`
Set a specific topic's help context. Returns the corrected context.
- `function SetTopicHelpId(const aTopicId: string; const aHelpId: string): string;`
Set a specific topic's help id. Returns the corrected string.
- `procedure SetTopicIconIndex(const aTopicId: string; const nIconIndex: Integer);`
Set the icon index of a specific topic.
- `function SetTopicKind(const aTopicId: string; const aNewKind: Integer): Integer;`
Set the topic kind (0: normal; 1: empty; 2: Link to URL; 3: Link to file)
- `procedure SetTopicStatusId(const aTopicId: string; const aStatusId: string);`
Set the topic status ID
- `procedure SetTopicUrlFile(const aTopicId: string; const anUrlFile: string);`
Set the topic's URL file.
- `procedure SetTopicUrlLink(const aTopicId: string; const anUrlLink: string);`
Set the topic's URL link.
- `procedure SetTopicVisibility(const aTopicId: string; const aVisibility: Integer);`
Set the visibility for the topic

Var HndTopicsEx

Additional properties and methods for Topics

- `function FindTopicWithHelpContext(const aHelpContext: Integer): string;`

Returns the topic with the specific Help Context.

- `function FindTopicWithHelpId(const aHelpId: string): string;`
Returns the topic with the specific Help Id.
- `function GetTopicCaptionHierarchy(const aTopicId: string; const aIncludingProjectTopic: Boolean): TStringDynArray;`
Returns the captions of all parent topics of the specified topic
- `function GetTopicDirectChildrenCountGenerated(const aParentId: string; const doExcludeHiddenInToc: Boolean): Integer;`
Returns the number of direct generated children a topic has.
- `function GetTopicDirectChildrenListGenerated(const aParentId: string; const doExcludeHiddenInToc: Boolean): THndTopicsInfoArray;`
Returns a list of direct generated children of the specified topic.
- `function GetTopicIndexGenerated(const aTopicId: string; const isZeroBased: Boolean; const doExcludeHiddenInToc: Boolean): Integer;`
Returns the index of the topic from its parent. Considering only generated topics.
- `function GetTopicIndexHierarchyGenerated(const aTopicId: string; const isZeroBased: Boolean; const doExcludeHiddenInToc: Boolean): THndIntegerArray;`
Returns an array representing the topic's index hierarchy. Ex: [1,0,3,4] for 1.0.3.4. Considering only generated topics
- `function GetTopicIsGenerated(const aTopicId: string; const doExcludeHiddenInToc: Boolean; const doCheckParentVisibilityToo: Boolean): Boolean;`
Returns True if a topic is generated: visible and exported in current build.
- `function GetTopicIsGeneratedForBuild(const aTopicId: string; const aBuildId: string; const doExcludeHiddenInToc: Boolean; const doCheckParentVisibilityToo: Boolean): Boolean;`
Returns True if a topic is generated for a specific build: visible and exported in that build.
- `function GetTopicListGenerated(const doExcludeHiddenInToc: Boolean; const aIncludingProjectTopic: Boolean): THndTopicsInfoArray;`
Returns a list of generated topics and their associated ID as object.
- `function GetTopicNextGenerated(const aTopicId: string; const doExcludeHiddenInToc: Boolean): string;`
Returns the next generated topic.
- `function GetTopicNextSibblingGenerated(const aTopicId: string; const doExcludeHiddenInToc: Boolean): string;`
Returns the next generated sibling topic.
- `function GetTopicPreviousGenerated(const aTopicId: string; const doExcludeHiddenInToc: Boolean): string;`
Returns the previous generated topic.

- `function GetTopicPreviousSiblingGenerated(const aTopicId: string; const doExcludeHiddenInToc: Boolean): string;`
Returns the previous generated sibling topic.
- `procedure SynchronizeAllHelpIds;`
Overwrite all topic's help ids based on caption.

Var HndTopicsKeywords

Handle relationship between topics and keywords

- `function AreTopicAndKeywordAssociated(const aTopicId: string; const aKeywordId: string): Boolean;`
Indicates whether the specified topic and keyword are associated.
- `function AssociateTopicWithKeyword(const aTopicId: string; const aKeywordId: string): Boolean;`
Link a specific topic to a specific keyword.
- `function DissociateKeywordFromAllTopics(const aKeywordId: string): Boolean;`
Check all topics and if they are associated with this keyword, dissociate them.
- `function DissociateTopicFromAllKeywords(const aTopicId: string): Boolean;`
Check all keywords and if they are associated with this topic, dissociate them.
- `function DissociateTopicFromKeyword(const aTopicId: string; const aKeywordId: string): Boolean;`
Remove the association between the specified topic and keyword.
- `function GetKeywordsAssociatedWithTopic(const aTopicId: string): TStringDynArray;`
Get a list of keyword ids associated with a specific topic.
- `function GetTopicsAssociatedWithKeyword(const aKeywordId: string): TStringDynArray;`
Get a list of topic ids associated with a specific keyword.

Var HndTopicsKeywordsEx

Additional properties and methods to handle relationship between topics and keywords

- `function GetGeneratedTopicsAssociatedWithKeyword(const aKeywordId: string): TStringDynArray;`
Get a list of generated topic ids associated with a specific keyword.

Var HndTopicsMeta

Access to topics meta data

Type THndTopicsPropertyInfo

Information about a specific property

- `key: string;`
- `value: string;`
- `topicid: string;`

Type **THndTopicsPropertyInfoArray**

Array of THndTopicsPropertyInfo

Var **HndTopicsProperties**

Handle relationship between topics and properties

- `procedure DeleteAllTopicCustomProperties(const aTopicId: string);`
Delete all custom properties for a specific topic
- `procedure DeleteTopicCustomProperty(const aTopicId: string; const aCustomPropertyName: string);`
Delete a custom property for a specific topic
- `function GetTopicCustomPropertiesList(const aTopicId: string): THndTopicsPropertyInfoArray;`
Returns a list of custom properties for this topic
- `function GetTopicCustomPropertyExists(const aTopicId: string; const aCustomPropertyName: string): Boolean;`
Returns true if a custom property with that name exists
- `function GetTopicCustomPropertyValue(const aTopicId: string; const aCustomPropertyName: string): string;`
Returns the value of a custom property
- `function RenameTopicCustomProperty(const aTopicId: string; const aOldCustomPropertyName: string; const aNewCustomPropertyName: string): Boolean;`
Rename a custom property
- `procedure SetTopicCustomPropertyValue(const aTopicId: string; const aCustomPropertyName: string; const aValue: string);`
Set the value of a custom property

Var **HndTopicsTags**

Handle relationship between topics and tags

- `function AreTopicAndTagAssociated(const aTopicId: string; const aTagName: string): Boolean;`
Indicates whether the specified topic and tag are associated.
- `function AssociateTopicWithTag(const aTopicId: string; const aTagName: string): Boolean;`

Link a specific topic to a specific Tag.

- `procedure DissociateAllForTag(const aTagName: string);`
Remove any association with the specified tag.
- `procedure DissociateAllForTopic(const aTopicId: string);`
Remove any association with the specified topics.
- `function DissociateTopicFromTag(const aTopicId: string; const aTagName: string): Boolean;`
Remove the association between the specified topic and tag.
- `function GetTagsAssociatedWithTopic(const aTopicId: string): TStringDynArray;`
Get a list of tag Ids associated with a specific topic.
- `function GetTopicsAssociatedWithTag(const aTagName: string): TStringDynArray;`
Get a list of topic Ids associated with a specific tag.

Type **TUIControlType**

Reference to a specific UI control: `uiMainForm`, `uiTreeToc`, `uiTreeKeywords`, `uiTreeLibrary`, `uiTopicEditor`

Type **THndSelectedExecMethod**

Method run for each selected node. Return True to break the loop: `function(aSelectedId: string): Boolean;`

Var **HndUI**

Methods and properties of HelpNDoc's user interface.

- `procedure CancelEditTree(aTree: TUIControlType);`
Cancel any edition being done on one of the trees (toc, library, keywords)
- `function EditSnippetContent(const aSnippetId: string; const aSelStart: Integer; const aSelLength: Integer; const aLinearPos: Integer): Boolean;`
Show the snippet content editor. Return true if edit was successful. If selection arguments are defined: selection is performed
- `procedure FocusFirstTopic(const doExpandAll: Boolean);`
Focus the first topic, right after the project topic
- `procedure ForEachSelectedKeywordDo(aSelectedExecMethod: THndSelectedExecMethod);`
Call the specified method for each keyword selected in the keywords tree
- `procedure ForEachSelectedLibraryItemDo(aSelectedExecMethod: THndSelectedExecMethod);`
Call the specified method for each library item selected in the library tree

- `procedure ForEachSelectedTopicDo(aSelectedExecMethod: THndSelectedExecMethod);`
Call the specified method for each topic selected in the table of contents tree
- `function GetControl(const aControlType: TUIControlType): TObject;`
Returns a specific UI control
- `function GetCurrentControl: TObject;`
Returns the currently active control
- `function GetCurrentFrame: TObject;`
Returns the currently visible frame in the UI (e.g. topic editor, topic options...)
- `function GetCurrentKeyword: string;`
Returns the currently selected keyword in the UI.
- `function GetCurrentLibraryItemId: string;`
Returns the currently selected library item in the UI.
- `function GetCurrentTopic: string;`
Returns the currently selected topic in the UI.
- `function GetTopicEditorObject: TObject;`
Returns the topic editor object, which can be manipulated using HndEditor methods.
- `procedure SaveCurrentFrame();`
Save the currently edited content to project file
- `function SetCurrentKeyword(const aKeywordId: string): Boolean;`
Select the specified keyword in the UI.
- `function SetCurrentTopic(const aTopicId: string): Boolean;`
Select the specified topic in the UI.
- `procedure SetCurrentTopicAndPlaceCursor(const aTopicId: string; aLinearPos: Integer);`
Select the specified topic in the UI and place the cursor where specified.
- `procedure SetCurrentTopicAndSelect(const aTopicId: string; aSelectionStart: Integer; aSelectionLength: Integer);`
Select the specified topic in the UI and select a specific part of that topic.
- `procedure SetCurrentTopicAndShowCheckpoint(const aTopicId: string; const aCheckpointName: string);`
Select the specified topic in the UI and show the checkpoint specified.
- `function SetCurrentTopicByHelpContext(const aHelpContext: Integer): Boolean;`
Select the topic with the specified help context
- `function SetCurrentTopicByHelpId(const aHelpId: string): Boolean;`
Select the topic with the specified help ID

Migrating scripts and templates

The HelpNDoc API evolves from time to time and this requires an update of older scripts and templates.

The following topics explain the changes required for each new versions of HelpNDoc:

- [Migrating scripts from V4 to V5](#)
- [Migrating scripts from V5.0 to V5.1](#)
- [Migrating scripts from V6 to V7](#)

Migrating scripts from V4 to V5

HelpNDoc version 5 includes several modifications to scripting methods. The following list helps migrate scripts (and templates) from earlier versions of HelpNDoc.

Older versions	HelpNDoc v5	Remarks
HndProjectsMeta.GetItemMetaStringValue('DefaultTopic')	HndProjects.GetProjectDefaultTopic()	Use this method to get the default topic
HndBuilds.GetBuildCustomValue...	HndBuildsMeta.GetItemMeta*Value	Where * can be: Bool, Int, List or String
HndBuildsEx.GetChmButtonVisibilityHex	HndBuildsMetaEx.GetChmButtonVisibilityHex	
HndBuildsEx.GetChmNavigationPaneStyleHex	HndBuildsMetaEx.GetChmNavigationPaneStyleHex	
HndTopics.GetTopicDirectChildrenCountVisible	HndTopicsEx.GetTopicDirectChildrenCountGenerated	
HndTopics.GetTopicDirectChildrenListVisible	HndTopicsEx.GetTopicDirectChildrenListGenerated	
HndTopics.GetTopicsVisible	HndTopicsEx.GetTopicsGenerated	
HndTopics.GetTopicListVisible	HndTopicsEx.GetTopicListGenerated	

HndTopics.GetTopicNextVisible	HndTopicsEx.GetTopicNextGenerated	
HndTopics.GetTopicNextSiblingVisible	HndTopicsEx.GetTopicNextSiblingGenerated	
HndTopics.GetTopicPreviousVisible	HndTopicsEx.GetTopicPreviousGenerated	
HndTopics.GetTopicPreviousSiblingVisible	HndTopicsEx.GetTopicPreviousSiblingGenerated	
HndTopics.SetTopicsVisible	HndTopics.SetTopicVisibility	HndTopics.GetTopicVisibility

User interface related operations

To speed-up global processing, all operations are done in memory and doesn't affect the user interface (UI). It might be required to update the UI. This is why the following methods must be considered based on the need of the script:

Non-UI version	UI version	Remarks
HndTopics.GetCurrentTopic	HndUI.GetCurrentTopic	UI version will return the currently edited topic
HndTopics.SetCurrentTopic	HndUI.SetCurrentTopic	UI version will set the currently edited topic

Migrating scripts from V5.0 to V5.1

The following modifications were made between HelpNDoc 5.0 to HelpNDoc 5.1. If upgrading from an earlier version, other modifications need to be applied: [Migrating scripts from V4 to V5](#)

HelpNDoc v5.0	HelpNDoc v5.1	Remarks
HndBuildsMetaEx.GetChmNav	HndBuildsMetaEx.GetChmNavigationP	The build id is now required. Note: this is

igationPaneStyleHex()	aneStyleHex(const aBuildId: string)	usually only used in the CHM documentation format
HndBuildsMetaEx.GetChmButtonVisibilityHex()	HndBuildsMetaEx.GetChmButtonVisibilityHex(const aBuildId: string)	The build id is now required. Note: this is usually only used in the CHM documentation format

Migrating scripts from V6 to V7

HelpNDoc 7.0 introduces a completely new script engine. It is still based on the Pascal language but some differences with previous versions are listed bellow. If upgrading from an earlier version, other modifications need to be applied: [Migrating scripts and templates](#)

HelpNDoc v6	HelpNDoc v7	Remarks
<pre>var int1, int2: integer; str1, str2: string;</pre>	<pre>var int1, int2: integer; var str1, str2: string;</pre>	Variables with different types must use the var keyword for each type
<pre>const int1, int2: integer; str1, str2: string;</pre>	<pre>const int1, int2: integer; const str1, str2: string;</pre>	Constants with different types must use the const keyword for each type
SetLength(ARRAY_OR_STRING, NEW_LENGTH)	ARRAY_OR_STRING.SetLength(NEW_LENGTH)	The SetLength() method can't be used for arrays anymore
Print 'VALUE'	Print('VALUE')	The Print() method requires parenthesis
<pre>procedure proc1(); begin proc2(); end; procedure proc2(); begin end;</pre>	<pre>procedure proc2(); begin end; procedure proc1(); begin proc2(); end;</pre>	If a procedure calls another one, it must be declared before that method. Alternatively, proc2 can be declared before proc1 as a forward method: procedure proc2;

		forward;
HndGeneratorInfo.AssetsList: TStringList	HndGeneratorInfo.GetAssetsList(): TStringDynArray	Method return type has changed from TStringList to TStringDynArray
HndGeneratorInfo.GeneratedFiles: TStringList	HndGeneratorInfo.GetGeneratedFiles(): TStringDynArray	Method return type has changed from TStringList to TStringDynArray
HndBuildsEx.GetBuildByName(const aBuildName: string)	HndBuilds.GetBuildWithName(const aBuildName: string)	Method has moved to another object and has been renamed

Code only template files

Template files which only contains code must now encapsulate code within the<% and %> delimiters.

Frequent errors in custom templates

Here are some possible errors that you might see with custom templates after you upgrade. Refer to the table above to update custom templates and remove those errors:

- [Error] aliases.pas.html(4, 5): Unknown name "nCurTopic"
- [Error] context.pas.html(4, 5): Unknown name "nCurTopic"
- [Error] keywords.pas.hhk(4, 2): Unknown name "nBlocLevel"
- [Error] topics.pas.html(4,5): unknown name sDefaultTopicId

Using the integrated web server

Most web-browsers won't allow the HTML documentation generated by HelpNDoc to [run on the local computer](#) due to security restrictions. That's why HelpNDoc includes a full featured web server to work around this limitation: local HTML documentation generated by HelpNDoc can be tested as if it was uploaded to a real web server.

Launching the web server from the generation window



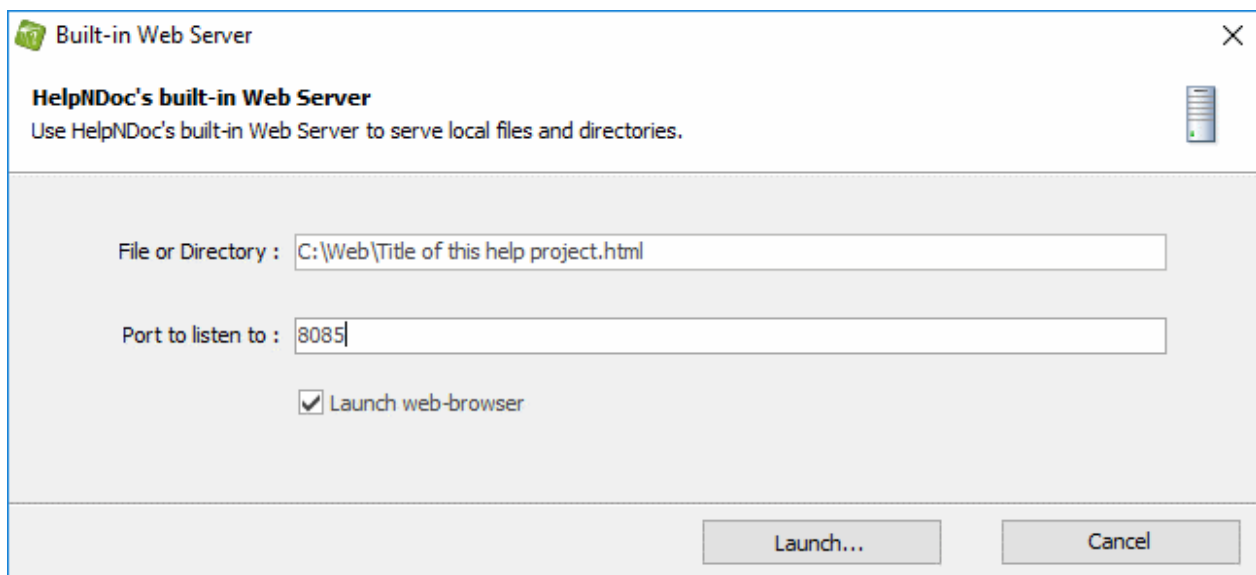
Once the HTML documentation generation process finishes, HelpNDoc will provide multiple. Using the "Launch web server" option will immediately start the integrated web server with the correct options, and launch the default web browser at the required URL to display the newly generated documentation for testing purposes.

Serving a previously served location

When a specific folder or file has been served using HelpNDoc's integrated web server, it will be remembered so that it can rapidly be served another time. This can be achieved as follows:

- From HelpNDoc's "Tools" ribbon tab, locate the "Extra" group
- Click the arrow of the "Launch web server" button to display a list of recently served folders and files
- Click one of those items to serve it again

Serving random files and directories



It might be useful to serve specific files or directories using HelpNDoc's built-in web server. This can be achieved as follows:

- From HelpNDoc's "Tools" ribbon tab, locate the "Extra" group
- Click the top part (without the arrow) of the "Launch web server" button to display the server configuration window
- Customize the web server's options (see bellow) and hit the "Launch" button to start the web server

The following options can be configured before launching the server:

- **File or directory:** what file or directory will be served. If a file is chosen, the root of the server

will be the directory where that file is placed

- **Port to listen to:** which port will be used by the web server to listen to incoming connections
- **Launch web-browser:** whether the default web-browser will be launched at the correct URL when the server is started

Backing up projects

It is very important to make frequent backup of HelpNDoc projects (as well as any important data) to be able to recover a previous version in case of a software or hardware failure. HelpNDoc includes a handy backup action:

1. Navigate to HelpNDoc's "Tools" ribbon tab
2. In the "Low level" group, click "Backup Project..."
3. Check the proposed backup location and file name and click "Save"

Note: Default backup saving path can be defined in [HelpNDoc's options](#).

Documentation formats specifics

Documentation formats can be very different from one another. The following topics explain some documentation format's specific information:

- [CHM files and programming languages](#) : How to open CHM help files using various programming languages ;
- [HTML help URL parameters](#) : How to customize the HTML help via URL parameters ;
- [Context sensitive HTML help](#) : How to open specific HTML documentation topics ;

CHM files and programming languages

HelpNDoc generates standard Windows CHM help files which can be opened from any Windows application. The following section explains how to open CHM help files using various programming languages. This includes:

- [.NET \(C#\) Integration](#)
- [Delphi integration](#)
- [Java integration](#)
- [Microsoft Access integration](#)
- [Visual Basic integration](#)
- [WinDev integration](#)

.NET (C#) integration

Opening a CHM help file from a .NET (C#) program

The .NET framework includes the `Help.ShowHelp` method to display the contents of a Help file.

See: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.help.showhelp>

Delphi integration

Opening a CHM help file from a Delphi program

Modern versions of Delphi include the `Vcl.HtmlHelpViewer` unit which implements the wrapper for Windows HTMLHelp function API. Check Embarcadero's web-site to learn how to use it: <http://docwiki.embarcadero.com/Libraries/en/Vcl.HtmlHelpViewer>

Older versions of Delphi requires a dedicate unit. See: [Older versions of Delphi](#)

Older versions of Delphi

Opening a CHM help file from a program built using and older version of Delphi

Courtesy of The [Helpware Group](#), the Delphi HH Kit is a free download for Delphi 2/3/4/5/6/7... It consists of two units and a document file. The first unit is a port of the C++ header file "HtmlHelp.h". The second one is a library of HTML help related functions. Two versions exists based on the Delphi compiler capabilities:

- [Download "The Kit" version 2.1](#) - Size: 32KB, 3-Dec-2009
 - Unicode version now compatible with Delphi 6/7/... Delphi 2010.
 - HH_Funcs.pas utility now compiles under D2010. ANSI and Unicode versions of funcs.
 - HH.pas - HTMLHelp() function defaults to either ANSI or Unicode depending on the version of Delphi.
- [Download "The Kit" version 1.09](#) - Size: 32KB, 27-Aug-2008
 - ANSI version for Delphi 2/3/4/5/6/...
 - Fix bug with Windows Vista compatibility.
 - Updated D6OnHelpFix.pas to fix small memory leak on shutdown.

Java integration

Opening a CHM help file from a Java program

Courtesy of Daniel Whitworth, the following Java code assumes the CHM help file is located in the same directory as the program being run:

```
File file = new File("help.chm");
try
{
    Runtime.getRuntime().exec("HH.EXE ms-its:" + file.getAbsolutePath() + ":///TOPIC_ID.html");
} catch (IOException e1)
{
    e1.printStackTrace();
}
```

The highlighted parts should be customized according to your needs:

- **help.chm**: This is the CHM help file you would like to open
- **TOPIC_ID**: This is the topic ID you would like to open

Microsoft Access integration

Opening a CHM help file from Microsoft Access

You can find a detailed PDF document [here](#), courtesy Dave Liske.

Visual Basic integration

Opening a CHM help file from a Visual Basic program

Courtesy of [David E. Liske](#), the HTML help VB class is a free download for Visual Basic 5/6. It includes the source code and the documentation on how to use it.

[Download HTML Help VB Class](#) (Version: 3.0h, Size: 80 KB)

WinDev integration

Opening a CHM help file from WinDev

The [WHelp function](#) can be used to display a file or a help page for the CHM format.

Syntax using topics' Help Id:

```
WHelp(<NameOfHelpFile> [, <HelpId>])
```

Syntax using topic's Help Context number:

WHelp(<NameOfHelpFile> , <HelpContextNumber>)

Where:

- *NameOfHelpFile* is the name of the CHM help file to open
- *HelpId* is the topic help id to display
- *HelpContextNumber* is the topic's help context to display

See the [How to manage your topic identifiers in HelpNDoc](#) step-by-step guide.

HTML help URL parameters

It is possible to customize the behavior of the HTML documentation format through its parameters.

Show a specific topic

It is possible to show a specific topic (contextual help) by using its Help Id. See the [Context sensitive HTML help](#) topic to learn more.

Choose which tab is shown

By default, the HTML help includes 3 tabs: Contents, Index and Search. To choose which tab is shown when the HTML documentation is opened, simply add the "?tab=" URL parameter. Here are the available values:

URL parameter	Tab
no parameter	Show the Contents tab by default
?tab=index	Shows the Index tab
?tab=search	Shows the Search tab

Example:

`https://www.MY-SERVER.com/index.html?tab=index`

Search for a specific term

It is possible to run the search engine on a specific term when the page is shown. Simple add the "?search=" URL parameter.

Example:

```
https://www.MY-SERVER.com/index.html?search=help
```

Combining URL parameters

Behaving as normal URL parameters, it is possible to combine them altogether. For example, showing the "Search" tab while searching for the "help" term can be achieved as follows:

```
https://www.MY-SERVER.com/index.html?tab=search&search=help
```

Context sensitive HTML help

HelpNDoc's generated HTML documentation **can be used as a context sensitive help source** by using either a topic's Help ID or Help Context number

Using Help IDs

It is possible to show a specific topic using its [Help ID](#). As an example, HelpNDoc's online topic "Change topic properties" (whose Help Id is "Changetopicproperties") can be shown as follows:
<https://www.helpndoc.com/documentation/html/Changetopicproperties.html>

The pattern is *http://www.WEB-SITE.com/HELP-ID.html*

Where:

- "www.WEB-SITE.com" is the full address of your web site
- "HELP-ID" is the unique help id of the topic to show

Help IDs are unique topics identifiers (alpha-numeric) which you can customize. Learn more on [how to get and define topic's Help Ids](#).

Using Context numbers

The default HTML template also generate redirects to access topics via their [Help Context numbers](#). As an example, HelpNDoc's online topic "Change topic properties" (whose Help Context number is "78") can be shown as follows:

<https://www.helpndoc.com/documentation/html/context/78.html>

The pattern is *http://www.WEB-SITE.com/context/HELP-CONTEXT.html*

Where:

- "www.WEB-SITE.com" is the full address of your web site
- "HELP-CONTEXT" is the unique help context of the topic to show

Help context numbers are unique topics identifiers (numeric) which you can customize. Learn more on [how to get and define topic's Help Ids](#).

See the [How to manage your topic identifiers in HelpNDoc](#) step-by-step guide.

License key management

Once you've [purchased the full version of HelpNDoc](#), you need to activate it using a license key. How this is done depends on the licenses you've purchased:

- [Named licenses](#)
- [Floating licenses](#)

Named licenses

Named licenses can only be used by one specific person and installed on her own computer. To validate the correct use of the license, HelpNDoc needs to be activated on the computer where it will be used. To activate HelpNDoc, simply enter the license key you've retrieved from the customer's section.

Warning: Activating HelpNDoc requires an Internet access to connect to the license server. Once activated, HelpNDoc will check the license server from time to time (usually, every 90 days). If it can't connect to the license server, it will keep working for a certain period of time (usually, 14 days). Once this period is over, it won't run anymore and will require access to the license server. See [Grace period](#) topic for more information.

How to: Activate HelpNDoc

Once you've purchased a named license, you will be able to download the full version of HelpNDoc from your customer's section and request a license key.

When you launch your licensed copy of HelpNDoc, it will automatically ask you to enter that license key and connect to the license servers to validate it. Once this is done, HelpNDoc will be activated on that computer and you won't need to enter the license key again. It will periodically check with the license server to make sure the key is still valid.

How to: Update you license

Once your license has expired, you can choose to purchase an update: the license key won't change but you'll need to connect to the license server to update its content. This can be done by using the "*license -f*" command line option while connected to the Internet. As an example, using Windows' command prompt:

```
> C:\Program Files\IBE Software\HelpNDoc 9\hnd9.exe license -f
```

How to: Upgrade your edition of HelpNDoc

If you've placed an order to upgrade e.g. from the Standard Edition to the Professional Edition of HelpNDoc, your license key will not be valid anymore. You can simply activate your new edition of HelpNDoc after requesting a new license key.

Command line usage

It is possible to manage HelpNDoc's license from the command line. See [Usage from the command line](#) to learn more.

See the following step-by-step guides:

[How to activate a named license of HelpNDoc](#)

[How to deactivate a named license of HelpNDoc](#)

Grace period

Once an instance of HelpNDoc is activated, it will connect to the license server from time to time (usually, every 90 days) to check that the license is still valid. If it can't connect to the license server, it will keep working for a certain period of time, the grace period (usually, 14 days).

To fix that, make sure that the computer running HelpNDoc is connected to the Internet and that the following web sites are accessible and white-listed on your router / proxy / firewall:

- <http://www.ibe-software.com> and <https://www.ibe-software.com>
- <http://www.wyday.com> and <https://www.wyday.com>

Floating licenses

Floating licenses can be shared between anyone within a company, with the limit of one person per purchased license using it at the same time.

Floating license server

A floating license server is required to enforce this rule. It can be installed on any Windows, Linux or MacOS computer which is accessible from running instances of HelpNDoc (same network). The floating license server directory includes a file named `UserManual.html` which contains the latest information about how to operate the floating license server. The following resources are also useful:

- [How to install and activate the floating license server](#) to learn how to install the floating license server and activate it using your license key
- [How to connect HelpNDoc to the floating license server](#) to learn how to connect to the floating license server from an instance of HelpNDoc

Warning: Activating the floating license server requires an Internet access to connect to the license server. Once activated, the floating license server will check the license server from time to time. If it can't connect to the license server, it will keep working for a certain period of time (configurable with a maximum of 90 days). Once this period is over, it won't run anymore and will require access to the license server.

If an online activation is not possible, it is possible to proceed with an offline activation by exchanging XML activation files with HelpNDoc's support team.

The HelpNDoc software (the client) will only need to connect to the floating license server, it won't need to be connected to the Internet.

Configuring the floating license server

The file `TurboFloatServer-config.xml` is used to configure the floating license server. It must be edited before the floating license server is run and contains many comments to explain all available configuration options.

Using the log file

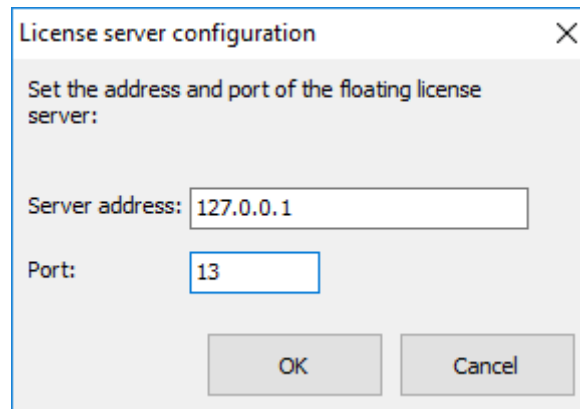
While troubleshooting the floating license server, the file `tfs-log.txt` can be used to get useful information about what happens while it is running. Make sure that you set the log level to "notification" in the `TurboFloatServer-config.xml` file to get as much information as possible:

```
<log file="tfs-log.txt" level="notification"/>
```

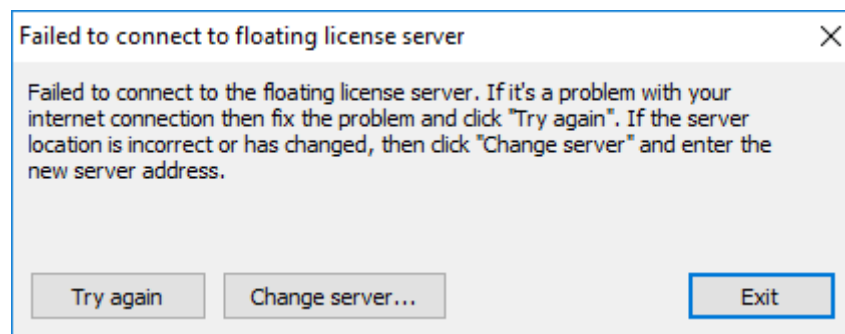
From the HelpNDoc instance

Floating licenses of HelpNDoc can be installed anywhere as long as it has network access to the

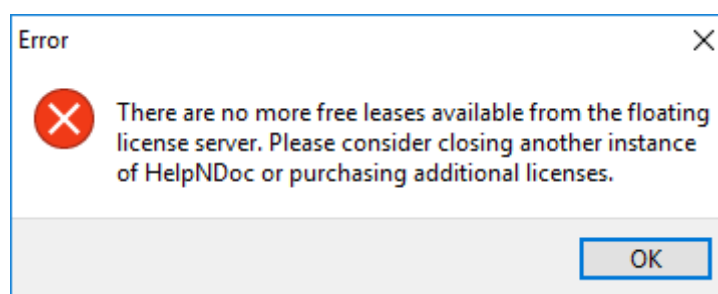
floating license server. When first run, it will request for the floating license server address and port:



This configuration will be saved for later use. If for some reason the server is not accessible, HelpNDoc will display a message with options to try again or change the server address or port:



Once connected to the server, it is possible that all leases are already taken: if for example you have purchased 3 floating licenses of HelpNDoc and you are trying to launch a 4th instance. When that happens, an error message will be shown. You'll need to close another instance of HelpNDoc or purchase additional floating licenses to launch another instance of HelpNDoc:



Troubleshooting

- [Check the floating license server's status](#) Check if the floating license server is activated and running
- [Check that the floating license server is listening to incoming connections](#) Make sure that the

floating license server is running and listening to connections

- [Check the connection to the server from HelpNDoc](#): Make sure that HelpNDoc can connect to the floating license server
- [How to handle dead leases](#): Deal with leases which haven't been correctly released
- [Listen to HTTPS communication](#): Run the floating license server behind an HTTPS server
- [Deactivating the floating license server](#): Deactivate a previously activated computer to activate a new one

See the following step-by-step guides:

[How to install and activate the floating license server](#)

[How to connect HelpNDoc to the floating license server](#)

Activating the floating license server

Before running and listening to incoming connections, the floating license server needs to be activated with your license key using the following command:

- **Linux:** `sudo ./turbofloatserver -a="LICENSE_KEY"`
- **Windows:** `.\TurboFloatServer.exe -a="LICENSE_KEY"`

Where:

- `LICENSE_KEY` is your private license key

Here are some errors which could occur during the activation process:

Error	Explanation
Failed to activate. Connection to the activation servers failed.	Make sure that the server is connected to the Internet and that the following domains are accessible from your server (e.g. white-listed on your Firewall / Proxy) for both HTTP and HTTPS protocols: wyday.com ; limelm.com ; ibe-software.com. If you are behind a proxy, make sure that the proxy address is correctly specified in the TurboFloatServer-config.xml file. If an online activation is not possible, proceed with an offline activation.
Failed to activate. The product key has already been activated on the maximum number of computers.	A floating license key can only be activated on one computer. To activate the floating license server on another computer, the previously activated computer needs to be deactivated first.

Check the floating license server's status

To check the floating license server's status, type the following command line:

- **On Windows:**

- o `.\TurboFloatServer.exe -v`

- **On Linux:**

- o `sudo ./turbofloatserver -v`

The output of that command could be:

- **Server activated and running:** "This floating license server instance is presumably activated. However, details about the activation cannot be displayed while the instance is running."
- **Server activated but NOT running:** "This floating license server instance is currently activated (with this product key: ABCD-EFGH-IJKL-MNOP-QRST-UVWX-YZ12) and will automatically re-verify its license in 90 days."
- **Server not activated:** "This floating license server instance is not activated."

Check that the floating license server is listening to incoming connections

Checking that the floating license server is listening to incoming connections on Windows

To make sure that the floating license server is running and listening to connections on the specified port (13 by default) on Windows, run the resource monitor as follows:

- From the Windows Start menu, type "resmon" without the quotes, right click on the "resmon" best match and click "Run as administrator"
- Once the "Resource Monitor" window is open, go to the "Network" tab
- Open the "Listening Ports" group
- Locate the "TurboFloatServer.exe" item and check that it is listening twice (IPv4 and IPv6) to the correct port (13), and that the "Firewall Status" column indicates "Allowed, not restricted"

Checking that the floating license server is listening to incoming connections on Linux

Once activated and running, we can test that the floating license server is listening to incoming connections on Linux. How this is done depends on your Linux distribution. On Ubuntu, you can use the following command:

- `sudo lsof -i:PORT_NUMBER`

Where:

- `PORT_NUMBER` is the port the server should be listening to. It is "13" by default, and can be changed from the "TurboFloatServer-config.xml" file

Check the connection to the server from HelpNDoc

When the floating license server is running correctly, we should be able to launch HelpNDoc, enter the server's address and port, and start using it.

If that doesn't work, it means that the computer where HelpNDoc is installed can't reach the floating license server. This could be caused by a software and/or hardware firewall on the server, the network infrastructure or the client. We recommend that you contact your IT team to get help with your specific network infrastructure.

You can also test the connection to the floating license server using the following PowerShell command line:

- `Test-NetConnection SERVER_ADDRESS -p PORT_NUMBER`

Where:

- `SERVER_ADDRESS` is the address of the floating license server's computer;
- `PORT_NUMBER` is the port used by the floating license server to listen to incoming connections. It is "13" by default and can be changed from the floating license server's "TurboFloatServer-config.xml" file

How to handle dead leases

If for some reason an instance of HelpNDoc crashes without releasing its lease, the license server will think that the lease is still in use and will now issue another lease when requested: this could lead to an error message indicating that "There are no more free leases available from the floating license server".

Leases are usually updated every 30 minutes (configurable on the server side) and dead leases will be eliminated then. If that happens often due to hardware or software failure, it is possible to speed up the process by lowering the lease time from the server. See the `<lease length="1800"/>` option from the floating license server's manual to learn more.

Listen to HTTPS communication

In addition to (or in place of) the default "raw" binary communication over the port you specified in the "bind" element, you can configure the floating license server to communicate over HTTPS. You can do this by "binding" the floating license server to an address and port for SCGI communication with an existing web server (e.g. Apache, NGINX, etc.).

This guide covers the very basics of the HTTPS server configuration. We assume you have experience running an HTTPS server.

Configure SCGI address and port

In the default "TurboFloatServer-config.xml" file you'll see an XML element like this:

```
<scgi just_scgi="false">
  <bind address="127.0.0.1" port="42"/>
</scgi>
```

There are 3 options to configure:

1. **The "just_scgi" attribute:** Set this to "false" and raw, unencrypted connections will still be allowed. Set this to "true" and only connections over SCGI (via an HTTPS server) will be accepted and the global bind element will be ignored.
2. **The "address" attribute in the "bind" element :** Set this to the address you want the HTTPS server that sits in front of the floating license server to connect to.

If the address is not present, the server accepts TCP / IP connections on all server host IPv6 and IPv4 interfaces if the server host supports IPv6, or accepts TCP / IP connections on all IPv4 addresses otherwise. Use this address to permit both IPv4 and IPv6 connections on all server interfaces. This value is the default.

If the address is ::, the server accepts TCP/IP connections on all server host IPv4 and IPv6 interfaces.

If the address is 0.0.0.0, the server accepts TCP/IP connections on all server host IPv4 interfaces.

If the address is a "regular" IPv4 or IPv6 address (such as 127.0.0.1 or ::1), the server accepts TCP/IP connections only for that IPv4 or IPv6 address.

3. **The "port" attribute in the "bind" element :** Set this to the port you want the HTTPS server that sits in front of the floating license server to connect to.

For security sake, we recommend always bind to a local address (like "127.0.0.1") so that all data that touches the floating license server must first go through an HTTPS server.

Configure the HTTPS server (Apache)

If you're using the Apache HTTPS server, modify your "httpd.conf" file to enable the "scgi" module (so Apache can communicate with the floating license server instance):

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
LoadModule ssl_module modules/mod_ssl.so
```

Next add a "VirtualHost" element that will accept HTTP connections and pass them along to the floating license server:

```
<VirtualHost yourhostaddress:443>
    # Pass *everything* to the floating license server via SCGI
    SetEnvIf Request_URI .* proxy-scgi-pathinfo
    ProxyPass / scgi://127.0.0.1:42/

    # ServerName gives the name and port that the server
    # uses to identify itself. This can often be determined
    # automatically, but we recommend you specify it
    # explicitly to prevent problems during startup.
    ServerName yourhostaddress:443

    # Turn SSL on
    SSLEngine on
</VirtualHost>
```

Replace "**yourhostaddress**" with the address name that the floating license server instance will be accessed from (i.e. the "public" address -- even if that "public" address is local network specific). Also, if you changed the SCGI "address" or "port" in the "TurboFloatServer-config.xml" file you'll also need to change the "ProxyPass" directive to use the address and port you changed them to.

Configure the HTTPS server (Nginx)

If you're using the NGINX HTTPS server, first make sure your NGINX has been compiled with [the SCGI module](#). You can do this by running "nginx -V" via command line and you should see something like this:

```
nginx version: nginx/1.15.7
built by gcc 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10)
built with OpenSSL 1.1.1a 20 Nov 2018
TLS SNI support enabled
```

configure arguments: `--sbin-path=/usr/local/sbin --with-http_ssl_module --with-http_v2_module --without-http_uwsgi_module --with-pcre-jit --with-threads`

If you **don't see** `--without-http_scgi_module` in the "configure arguments", then you know the NGINX instance was compiled with the SCGI module. It's compiled by default, so it should be available in every vanilla distribution of NGINX.

Next, modify your "nginx.conf" file to pass all communication to the address / port of your choosing to the floating license server instance via SCGI:

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    server_name yourhostaddress;

    # Maximum "request" size (i.e. max size the client sends us)
    client_max_body_size 10m;

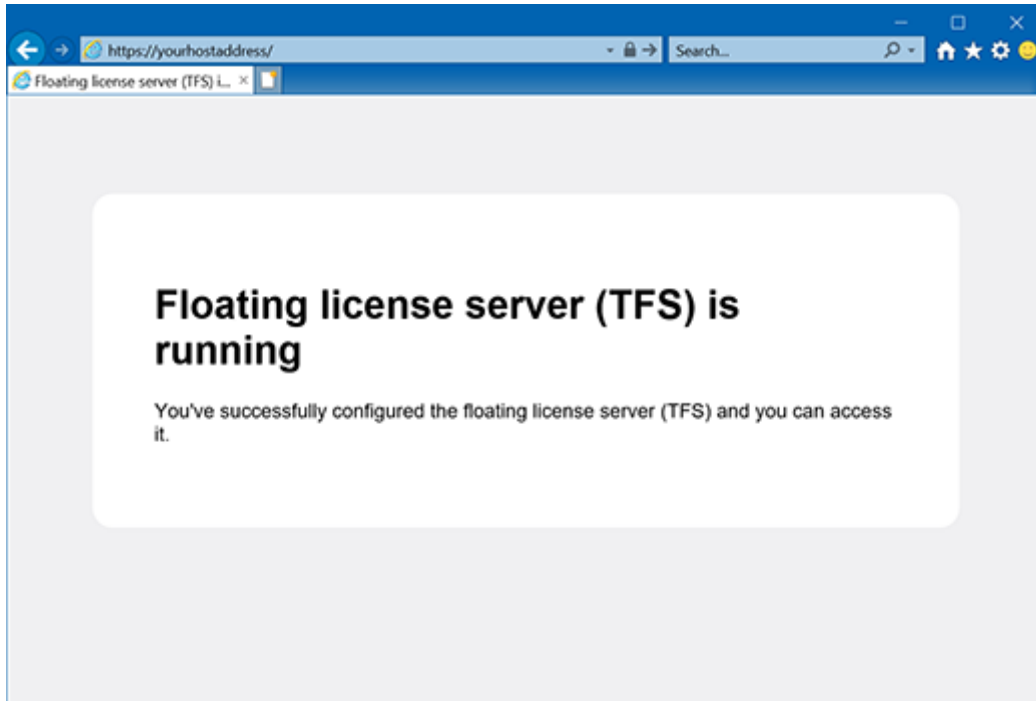
    # pass every connection to the Hosted floating license server on
    this machine.
    location / {
        scgi_param    SCGI                1;
        scgi_param    HTTPS                $https if_not_empty;
        scgi_param    REQUEST_METHOD      $request_method;
        scgi_param    CONTENT_LENGTH      $content_length;
        scgi_param    REMOTE_ADDR         $remote_addr;
        scgi_pass     127.0.0.1:42;
    }
}
```

Replace "**yourhostaddress**" with the address name that the floating license server instance will be accessed from (i.e. the "public" address -- even if that "public" address is local network specific). Also, if you changed the SCGI "address" or "port" in the "TurboFloatServer-config.xml" file you'll also need to change the "scgi_pass" directive to use the address and port you changed them to.

Test and fix access to the floating license server over HTTPS

After configuring the floating license server, then configuring the HTTPS server, start both of them up. If either one fails to start, read the error log and fix whatever problems are described. Once both are successfully started, you're ready to test if the floating license server is accessible via HTTPS. You can do this by opening a browser on your computer and typing the host address you've configured in your HTTPS server configuration.

So, in this example, "**yourhostaddress**" is the address that the HTTPS server is accessible from. Type, "https://**yourhostaddress**" into your browser, and if you've configured everything correctly (and both the floating license server and your HTTPS server are running) then this page should show:



If you see a certificate error page, then make sure you're using SSL certificates signed by a Certificate Authority. There are multiple options available, such as [the free Let's Encrypt certificate authority](#).

Deactivating the floating license server

To move the floating license server from one computer to another, it needs to be deactivated from the first computer before it can be activated on the second computer. To deactivate the floating license server instance, use the "-deact" command line switch:

- **Linux:** `sudo ./turbofloatserver -deact`
- **Windows:** `.\TurboFloatServer.exe -deact`

This will proceed with an online deactivation: it is done instantly but requires a working Internet connection.

To proceed with an offline deactivation (i.e. generate an offline activation request) then enter a file path using the "-deact" command line switch:

- **Linux:** `sudo ./turbofloatserver -deact="Location/To/OfflineReq.xml"`

- **Windows:** `.\TurboFloatServer.exe -deact="C:\Location\To\OfflineReq.xml"`

Then send the generated XML file to our support team at SUPPORT at IBE-SOFTWARE dot COM (replace at with @ and dot with .) so that we can mark your key as deactivated from that computer. Once done, you will be able to [activate the floating license server](#) on another computer.

FAQ and troubleshooting

Presents a list of Frequently Asked Questions (FAQ) as well as troubleshooting information.

Error and warning messages

Some common error and warning messages shown by HelpNDoc;

- [HelpNDoc shows an exception when launched](#)
- [Project already opened by another software](#)
- [Project is too old to open](#)

HelpNDoc shows an exception when launched

Symptoms

When launched, HelpNDoc shows an error message saying that an exception occurred.

Solutions

Software incompatibility: Sticky Password

The software "Sticky Password" is known to force many Windows applications to crash. Here is how to fix this problem if you are using this software:

- Navigate to the main Sticky Password window - Menu - Settings - Ignored Apps and try to add the executable file of HelpNDoc to the list. Usually, this is "C:\Program Files\IBE Software\HelpNDoc\HelpNDoc 9\hnd9.exe"
- If it doesn't help, on the same tab in the main Sticky Password window - Menu - Settings - Ignored Apps, try to disable autofill for the 32bit or the 64bit applications accordingly to the version of HelpNDoc that you are using.

Project already opened by another software

Message

This project is already opened by another software or instance of HelpNDoc. Please close the project first.

Explanation

This means that this HelpNDoc projects can't be opened or saved because either:

- Another instance of HelpNDoc has already opened this project without closing it
- It is used by another software or anti-virus
- It is marked as read-only or in a read-only folder
- The Windows user doesn't have rights to read and/or write to that location

Solutions

- Try rebooting your computer to make sure that no process is using that file anymore
- Make sure that only one instance of HelpNDoc has this project file opened
- Make sure that no other software has this project file opened
- White-list that file or folder in your anti-virus software
- Make sure that your Windows user has read / write access to that folder and project file
- Try to move the project file to a local simpler path such as c:\doc\doc.hnd to see if that is working better

Project is too old to open

Message

This project is too old to be opened with this version of HelpNDoc. Please open it and save it using HelpNDoc version x.9 to convert it first.

Explanation

The HND project file evolves from time to time in order to be able to support newer features: it is then automatically migrated to the newer version. However, when a project is too old, some migrations couldn't be applied and older versions of HelpNDoc need to be temporarily installed in order to migrate the project first.

Solution

Temporarily download and install older versions of the Free Personal Edition of HelpNDoc to proceed with the migration.

Older versions can be downloaded from: <https://www.helpndoc.com/whats-new>

The following versions are the one we recommend based on the requested migrations:

- Version 4.9.2 for projects created with HelpNDoc 3 and later: <https://dl.ibe-software.com/hnd/helpndoc-setup-4.9.2.132.exe>
- Version 3.9.2 for projects created with HelpNDoc 1 or 2: <https://dl.ibe-software.com/hnd/helpndoc-setup-3.9.1.648.exe>

Notes

- **Warning:** Make sure that you backup your projects first as it is not possible to go back to an earlier version otherwise
- Older versions can be installed in their own folder without un-installing newer versions as they are independent
- Once migration is complete, older versions can simply be un-installed from the computer

Help compilers

FAQ and troubleshooting about Help compilers.

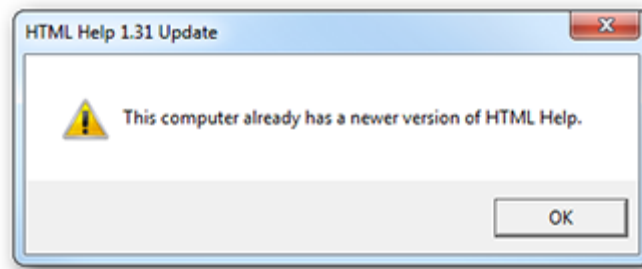
What compilers of libraries do I need to install?

HelpNDoc can generate the PDF, Word and HTML documentation by itself. However, to generate a CHM documentation, you will need to download and install the [Microsoft HTML Help Compiler](#).

Installing the Microsoft HTML Help Compiler displays a warning message?

Symptoms

When installing the Microsoft HTML Help Compiler on recent operating systems, you can receive a warning message saying that "This computer already has a newer version of HTML Help".



Solutions

- Discard the message as your system already has a valid and up-to-date help viewer. The compiler has correctly been installed despite of this message.

CHM and HTML help

FAQ and troubleshooting about CHM and HTML help.

The CHM viewer indicates that the page cannot be displayed

Symptoms

When viewing your CHM documentation, Microsoft's HTML Help Viewer is showing an error page saying either that:

- "The action has been canceled"
- "The page cannot be displayed"

Solutions

- Make sure your help file is not accessed from a network path or via a mapped networked drive. Try to copy the file locally and launch it again;
- Make sure your help file isn't in a path with symbols such as "#" (sharp). Once again, try to copy it locally before launching it;
- In some cases, you can have access to an "unblock" button in the properties page of the help file. Right click on the file then go to its properties and click the "unblock" button. This button is not available in all systems though.

CHM content is not displayed after Internet Explorer update

Symptoms

After an Internet Explorer update, when viewing your CHM documentation, Microsoft's HTML Help Viewer isn't showing anything in the topic's contents.

Solutions

The update process might have caused problems with some files registration. You can try to register them manually from the Start / Run prompt by entering each of these commands:

- `regsvr32 %systemroot%\system32\hhctrl.ocx` *<press the enter key>*
- `regsvr32 %systemroot%\system32\itss.dll` *<press the enter key>*

Despite modifying the navigation pane's width the CHM file is not updated

Symptoms

You change the navigation tab's width in the HelpNDoc's projects settings but when opening the CHM file, nothing has changed.

Solutions

The Microsoft HTML help viewer stores the help window's size and position for each individual help file as soon as it has been launched. Modifying the help settings after that won't have any effect as the help viewer will only read local configuration for that help file and ignore the file's settings set up using HelpNDoc.

A solution would be to erase the help viewer's configuration file, but be warned that this file contains all the configuration made to all the help files viewed on the system. So deleting this file will delete the configuration options for all the other files too.

This file is usually located there: `C:\Users\%username%\AppData\Roaming\Microsoft\HTML Help\hh.dat` where %username% is your Windows user name.

The search feature is not working in the CHM documentation

Symptoms

When trying to search within the CHM documentation, no results are found.

Solutions

- In your HelpNDoc project, click "Project options" in the "Home" ribbon tab and make sure the project language and charset are correct
- Make sure you are using a Windows installation setup with the same language as your HelpNDoc's project language
- Click the top part of the "Generate Help" button in the "Home" ribbon tab to access the "Generate documentation" dialog, then select your CHM build in the list on the left, then click

"customize" if the "Template settings" tab is not already visible on the right, then check the option "Use project charset for topics"

- Generate the CHM documentation again

Google Chrome shows an error when searching HTML documentation

Symptoms

When viewing a local (not uploaded to a server) HTML documentation, Google Chrome will show an error when trying to search within the documentation.

Solutions

HelpNDoc's HTML documentation generated using the default HTML template uses an AJAX call to retrieve the search data. This provides faster loading times for the overall documentation. However, when the HTML documentation is viewed locally, using the file:// protocol, Google Chrome will not allow the AJAX call.

- To work around this limitation, Chrome can be launched with the "--allow-file-access-from-files" command line switch. As an example, run:
 - `chrome.exe --allow-file-access-from-files`
- Another possible solution is to serve the local documentation via a server such as Apache or IIS, and therefore viewing your documentation using the http:// protocol. Google Chrome won't have the same restriction in that particular case.

The HTML help is broken when hosted by CloudFlare

Symptoms

We are hosting our HTML documentation produced by HelpNDoc on the CloudFlare CDN but it appears broken with various JavaScript errors.

Solutions

CloudFlare's Rocket Loader technology rewrites part of your HTML to provide faster loading which can break some JavaScript. The solutions are:

- Turn off Rocket Loader in the Performance Settings of CloudFlare. See: [Why is JavaScript or jQuery not working on my site?](#)
- Write a custom HTML template and disable Rocket Loader for some scripts. See: [How can I have Rocket Loader ignore my script\(s\) in Automatic Mode?](#)

Missing files when generating a CHM file in the same directory as HTML

Symptoms

Your project contains an HTML build which is followed by a CHM build and while the CHM build is fine, the HTML one is missing files such as topics, library items...

Solutions

This is due to the fact that most of the temporary files generated by the CHM build overwrite existing files generated by the HTML build. And by default, the CHM build will delete those temporary files after successful generation: the HTML build is therefore missing some files.

The best solution is to generate each build into its own specific folder to avoid such unwanted interactions.

The HTML documentation is not loading or behaving incorrectly

Symptoms

The HTML documentation is not correctly loaded or its behavior is chaotic.

Solutions

HelpNDoc's responsive HTML 5 template uses advanced techniques to provide a greater user experience and lower download sizes. See: <https://www.helpndoc.com/feature-tour/produce-html-websites>

Unfortunately, most web-browsers won't allow this code to run on local HTML files (using the **file://** protocol) due to security restrictions. That won't happen when the HTML documentation is uploaded to a web server and viewed using the **http://** or **https://** protocols. And that's why we [included a local HTTP web server](#): HTML documentation can be tested locally as web browsers are tricked into thinking it has been uploaded to a real web server.

The solution is to **use HelpNDoc's included HTTP web server** or upload the whole documentation file to a real web server.

If browsing local HTML files is mandatory, we recommend the use of the "legacy framed HTML template" that ships with HelpNDoc. Be aware that some web-browsers might block it due to the same security restrictions though. Here is how to select a template for a specific build:

<https://www.helpndoc.com/step-by-step-guides/how-define-build-settings-helpndoc>

HelpNDoc's powerful template system can be used to create completely customized HTML documentation to fit specific requirements. See: [Working with templates](#)

Table of contents is empty or loading in default HTML template

Symptoms

When browsing the HTML documentation produced using HelpNDoc's default responsive template, the table of contents is empty, shows an infinite loading animation or an error message.

Solutions

First of all, make sure that the documentation is correctly browsed from a web server. See: [The HTML documentation is not loading or behaving incorrectly](#)

Some web servers such as older versions of Microsoft IIS or Amazon S3 must be configured to return the correct mime type for JSON, WOFF and WOFF2 files. Requesting such files will have the following results:

- The web server will return a "404 Not Found" error for those files: this will break the table of contents in your HTML documentation;
- The web server will return those files as "text/plain" mime type. The default HTML template will try to seamlessly work around that problem for JSON files, but can't do anything about font files.

In any case, we recommend that you web server is correctly configured to return the correct mime type for those files. Here is how this can be done for some servers:

Microsoft IIS

Edit your [web.config file](#) and add the following content:

```
<system.webServer>
  <staticContent>
    <mimeTypeMap fileExtension=".json" mimeType="application/json" />
    <mimeTypeMap fileExtension=".woff" mimeType="application/font-woff" />
    <mimeTypeMap fileExtension=".woff2" mimeType="application/font-woff2" />
  </staticContent>
</system.webServer>
```

Amazon S3

Specify the property Content-Type property for every JSON, WOFF, and WOFF2 files. See Amazon S3 documentation: <https://docs.aws.amazon.com/AmazonS3/latest/API/RESTObjectPUT.html>

HTML documentation hosted on GitHub are broken

Symptoms

While everything is working with the local web server, your HTML documentation hosted on GitHub and served as a [GitHub pages](#) has a broken or indefinitely loading table of contents.

Solutions

By default, GitHub assumes that you are hosting Jekyll (a static site generator) compatible pages: file names starting with a "_" (underscore) are ignored and not served by the GitHub web server. However, HelpNDoc generates files starting with a "_" (table of contents, keywords...) and those files are needed for the default HTML template to work correctly.

The solution is to include a `.nojekyll` file in the root path of your HTML documentation to turn off Jekyll on GitHub pages. See this [article from GitHub](#) for more information.

Windows reserved file names

Symptoms

When generating an HTML based documentation format, an error occurs saying that it is not possible to create a specific file.

Solutions

HelpNDoc uses the [each topic's unique help Id](#) as it's file name. However, some file names (such as CON, NUL, COM1, LPT1...) are not valid file names, and Microsoft Windows will not allow the file to be created.

From [Microsoft's documentation](#):

Do not use the following reserved names for the name of a file: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9. Also avoid these names followed immediately by an extension; for example, NUL.txt is not recommended.

The only solution is to change the problematic topic's Help Id and use a value which will be accepted as a valid file name by Microsoft Windows.

PDF documentation

FAQ and troubleshooting about PDF documentation.

Adobe Reader won't print with "drawing error" message

Symptoms

Printing a PDF document using Adobe PDF reader fails with the message saying "Drawing Error".

Solutions

Try updating the Adobe PDF Reader software to the latest version. It can be downloaded freely from [Adobe's servers](#)

Microsoft Word documents

FAQ and troubleshooting Microsoft Word documents.

Table of contents page numbers are wrong in Word documents

Symptoms

When viewing a Microsoft Word document generated by HelpNDoc, the page numbers displayed in the table of contents are wrong.

Solutions

HelpNDoc generates fields for the page numbers in the table of contents of Microsoft Word documents. They are automatically managed and update by Microsoft Word and they are usually updated when printing the document. It is possible to force the update of those fields by selecting the whole table of contents in Microsoft Word, then hit the F9 keyboard shortcut.

Qt Help

FAQ and troubleshooting about Qt Help documentation.

Modifications made to a Qt help file are not updated in assistant.exe

Symptoms

When generating a modified Qt Help file and viewing it using the Qt assistant.exe software, the latest changes are not displayed.

Solutions

This is a [known bug](#) of the Qt assistant.exe software which creates a cache version of the .qhc help file. The only solution is to manually delete this cache as follows:

- Note the name of the .qhc file that you are creating. E.g. `sample.qhc`
- Hit the Windows+R keyboard shortcut to show the "Run" dialog
- Type the path of the Qt Help assistant cache folder. Usually this is: `%USERPROFILE%\AppData\Local\QtProject\Assistant`
- Delete the .qhc file and folder with the same name of your generated documentation project.
E.g. `sample.qhc` and `.sample`

Sales and license information

FAQ and troubleshooting about sales and license information.

What is HelpNDoc's update policy?

By purchasing one of the full versions of HelpNDoc, you are entitled for free updates for a full version cycle with a one year safety period. This means that, no matter what, you will benefit from one year of free updates. And if by the end of that year we haven't reached a full version cycle - for example if you buy version 9.0, a full version cycle will go up to version 10.0 included - you will still get free updates until that version cycle has been reached.

How much does HelpNDoc costs

Some factors influence the price of HelpNDoc:

- The HelpNDoc edition acquired: Standard Edition, Professional Edition or Ultimate Edition
- The number of licenses needed: Volume discounts are available as well as Site and Global licenses
- Whether you need a named (per-seat) or floating (concurrent) user license
- Whether HelpNDoc will be used for Educational or Governmental purposes

The most up-to-date prices are available from the [HelpNDoc Store](#).

Do you provide a discounted Educational license ?

We do offer Educational discounts. Please [contact us](#) to receive further details.

Do you provide a government license ?

We do offer Governmental discounts. Please [contact us](#) to receive further details.

I need a special license: site license or global license?

Site license provide an unlimited number of licenses for a single location in the world. World

license provide an unlimited number of licenses for multiple locations throughout the world. HelpNDoc can be licensed site-wide or world-wide. Please [contact us](#) to receive further details.

What kind of payment devises and currencies do you accept?

We use MyCommerce as our payment handling partner. They are globally known for their security and efficiency in payment processing. They accept many currencies including US dollars, Euros, British pound, Australian dollar, Japanese yen, Canadian dollar, Swiss franc, Russian rouble, Brasilian real, Norwegian krona, Swedish krona, Polish zloty, Chinese renminbi yuan, Taiwan dollar, Indian rupee. You can pay using various payment methods including credit card, paypal, wire-transfer, check and cash. To learn more about MyCommerce and the ordering options, visit the [MyCommerce FAQ page](#).

How can I request a written quote before ordering?

To obtain a written quote, first choose the edition and license that you'd like to acquire from [HelpNDoc's Store page](#) and click the "Buy Now" button. Use the "Request Quote" tab and fill-in the requested information to receive a written quote.

Miscellaneous

Miscellaneous FAQ and troubleshooting.

HelpNDoc download problem

Symptoms

You downloaded the HelpNDoc installation program but can't launch it.

Solutions

- Check that the installation program's extension is correctly set as an .EXE file. Some programs such as CA Security Software can rename downloaded .EXE files to .EFW

Doc or DocX files can't be imported

Starting with HelpNDoc 8, *.doc documents can't be imported anymore. You can use an external application such as Microsoft Word to convert the legacy *.doc file format to the recommended *.docx format.

Starting with HelpNDoc 7, *.docx documents are imported by HelpNDoc without the need for any third-party software.

For earlier versions, the free "Microsoft Office Compatibility Pack for Word, Excel, and PowerPoint File Formats" must be installed on some systems in order to import Microsoft Word *.doc or *.docx files. It is automatically installed with some older versions of Microsoft Office but not with the latest versions. It can be downloaded from <https://www.helpndoc.com/downloads/office-converter/FileFormatConverters.exe>

Some panels are missing or HelpNDoc's Window is hidden

Symptoms

When launching HelpNDoc, either the main window can't be seen or some panels are missing.

Solutions

Use the "-reset" command line option to reset HelpNDoc's layout:

If you are using HelpNDoc 6.8 or later, you can try to reset HelpNDoc's user interface settings as follows:

- Close any opened HelpNDoc projects
- From HelpNDoc's "File" menu, click "Options"
- Click the "Reset" button at the bottom left of the newly opened "HelpNDoc Options" dialog
- Click "Reset Forms, Panels and Toolbars"
- Launch HelpNDoc again

If that doesn't fix the problem or if you are using an older version of HelpNDoc, please try to launch HelpNDoc with the "-reset" command line option to reset all its settings as follows:

- Use the WINDOWS + R keyboard shortcut to open the Run panel
- Indicate HelpNDoc's installation path with the -reset option. Usually it is "C:\Program Files\IBE Software\HelpNDoc 9\hnd9.exe -reset"
- Restart HelpNDoc normally from now on

Note: for older HelpNDoc 7, 6 or 5 versions, adapt the command line accordingly.

The table of contents is missing topics or behaving strangely

When the table of contents is [filtered](#), some filtered topics might be missing and the table of contents could behave strangely. Make sure that you [clear any filters first](#) to restore the default behavior. Some possible strange behaviors include:

- Newly added topics might not be visible, because they are filtered

- Moving a topic using the "Move" button could make it move intermittently, because it is actually moved based on currently filtered topics