10Tec DocMounter 3.0 Manual

Table of Contents

Acknowledgments

We would have never created DocMounter without a good text editor component. Our partner company, AlterNET Software (https://www.alternetsoft.com/), provided such a component called SyntaxEdit for this development. Due to this component, we can edit the XML-like source of documentation using syntax highlighting and code folding. This great editor also gave us the ability to implement a built-in customizable spell checker. Now we are producing help solutions without typos and major grammar mistakes directly from DocMounter without the need to use an external editor with a spell checker like Microsoft Word.

DocMounter Concept

Purpose

DocMounter is a tool used to document .NET assemblies. You can write descriptions for the members of your assemblies and create conceptual topics explaining the principles of your products in one DocMounter project.

DocMounter allows you to generate the results in the following 3 different formats:

1) An MSHC help file in the MSHelp3 format you can view with 10Tec Help Viewer or Microsoft Help Viewer.
2) A manual in the HTML format you can use to convert into a PDF or any other suitable format.
3) An accompanying XML description file for your .NET assemblies in the Microsoft XML Documentation Comments format (brief and full versions).

Terms

We use the following two main terms related to topics in a DocMounter project:

- Member topic (MT).
  This is a page describing a member of your assembly (property, method, class, etc.) It is linked to the corresponding member and is marked is obsolete automatically if the member is removed.
- Conceptual topic (CT).
  A page that describes a concept or contains other useful information about your assemblies. These pages aren't linked to any members and exist regardless of them.

Documentation tags

The documentation is written with the help of a special set of tags you insert in normal text to provide special features, such as hyperlinks to members, tables, images, etc. This set of DocMounter tags is based on the standard Microsoft XML Documentation Comments tags.

DocMounter uses the Microsoft documentation compiler for managed class libraries called Sandcastle to produce topic pages in the HTML format for the MSHC file. Sandcastle implies that the conceptual documentation is written using a special Microsoft Assistance Markup Language (MAML). You can use it to write rich CTs in DocMounter - most of the MAML features are supported. However, the XML Documentation Comments format used to document classes in Visual Studio, uses another set of tags. To avoid thinking about what tags can be used in CTs or MTs, we designed our special set of DocMounter tags you can use in topics of both types. It is a minimal set of tags but providing you with most features used in a fully functional documentation.

MSHC files is the main, but not the only file type you can produce with DocMounter. If you use the standard DocMounter tag set to write your documentation, you will be also able to build an equivalent manual document from your project. If you build only MSHC files using DocMounter, you can also use MAML tags in your CTs to provide more features.

Main Benefits of DocMounter

1) DocMounter lets you separate your code and its description. As a result, your code isn't bloated because of documentation in your source files - what you generally get in Visual Studio. This also allows the developer and technical writer to work on the product development and its documentation at the same time independently.
2) Visual Studio does not allow you to create conceptual documentation. In DocMounter, you can create conceptual topics in addition to topics describing members and create links from one kind of topics to another in one visual environment.
3) DocMounter allows you to build not only an MSHC file with a tree structure of conceptual topics, but also a printable manual in which all the conceptual topics are placed one-by-one.
4) DocMounter produces documentation similar to pages from Microsoft help.
5) The DocMounter interface is intuitive and simple, and all help authoring tasks can be done a visual way. At the same time, DocMounter provides all main features you may need to write help solutions.
6) You write both conceptual and member topics in one unified interface using the same principles and tag set. You can also use a handy visual Insert Tag dialog instead of writing tags manually.
7) Both conceptual topics and member topics are automatically compiled into full-featured help pages that use the same visual and functional template.
8) If assembly members you are documenting have been changed (renamed, parameter list was changed, etc.), DocMounter automatically updates the project to insert the updated members. All obsolete members aren't removed and can be easily found using a special find command. This allows you to copy the descriptions from the obsolete members into the new ones and remove the obsolete members from the project manually when you no longer need them.
9) If you already documented your assemblies using XML Comments in Visual Studio, you can easily switch to DocMounter to continue documenting your assemblies due to the Import Member Descriptions function. You can import member descriptions from the XML comment files accompanying assemblies or even from the description attributes stored directly in your assemblies!
10) Though DocMounter is based on the external Sandcastle help compiler, it adds some cool additional features the original version of Sandcastle does not have. Among them are the abilities to insert pictures into member topics and to make cross-reference links between member topics and conceptual topics.

Installation and System Requirements

The DocMounter package

DocMounter requires the .NET Framework 4.7.2 or higher to be installed on your pc.

DocMounter is supplied as a portable application. After you unzipped the DocMounter package, run TenTec.DocMounter.exe to launch the tool.

Sandcastle compiler

Sandcastle is a Microsoft documentation compiler for managed class libraries. DocMounter uses this external tool to generate HTML files for member and conceptual topics while building MSHC help files.

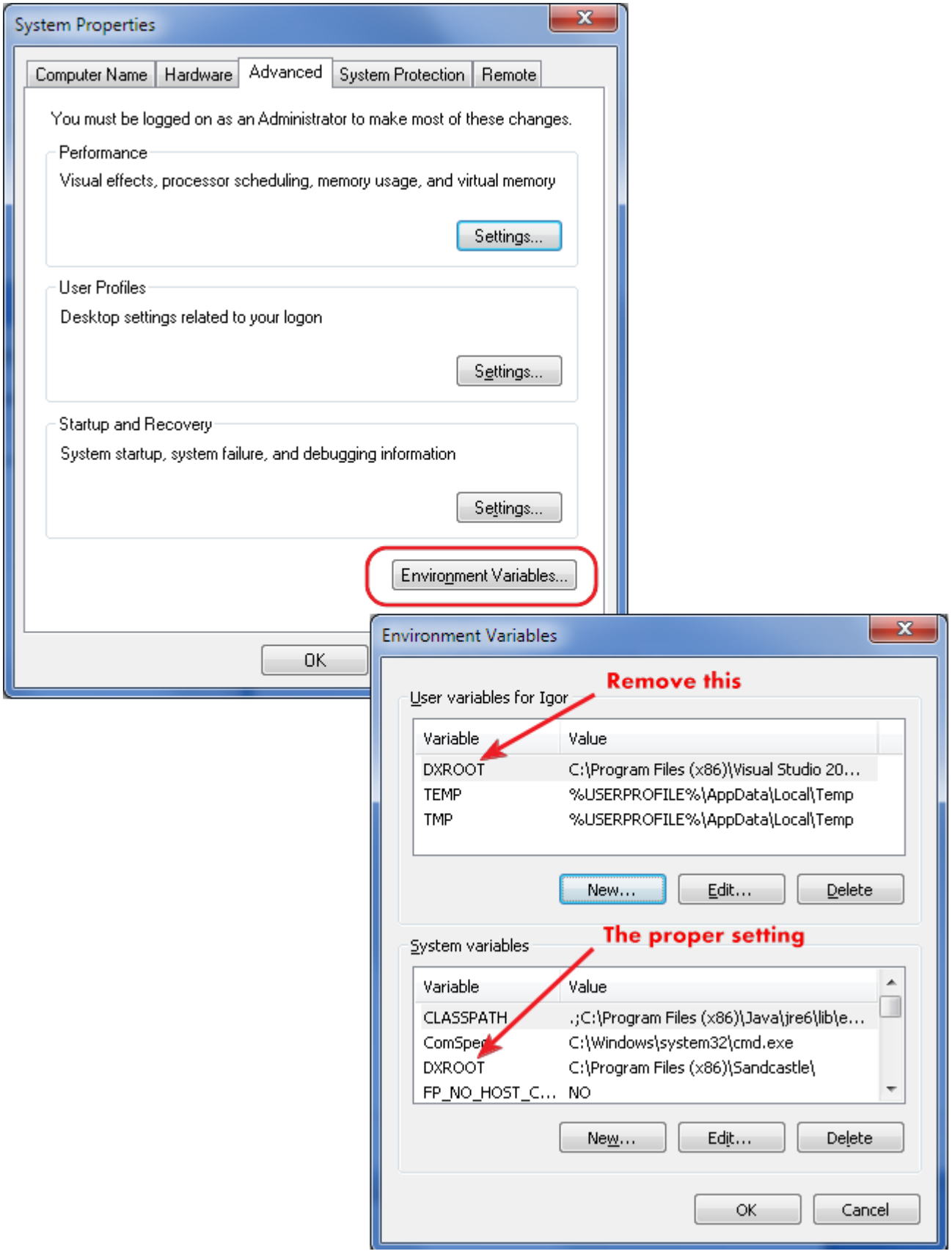
Originally Sandcastle was hosted at Codeplex. This portal has been closed and you can no longer download Sandcastle installations from the original source. Microsoft and other companies may have cached official Sandcastle installations on their servers, but these may not be the latest version ever released – June 2010 Release (Version 2.6.1062.1). We do not recommend using earlier versions of Sandcastle because DocMounter may fail while calling Sandcastle tools from older releases.

The current version of DocMounter was developed and tested with the latest official release of Sandcastle (Version 2.6.1062.1). You can find and download it from the 10Tec Company website.

The installation of Sandcastle itself is trivial: you launch the MSI installer and follow traditional installation wizard steps. To apply the Sandcastle styles patch, unpack the contents of the styles patch zip archive to the corresponding folders in the Sandcastle installation directory. The default location of Sandcastle is "C:\Program Files\Sandcastle\" in a 32-bit Windows or "C:\Program Files (x86)\Sandcastle\" in a 64-bit Windows.

DocMounter automatically finds Sandcastle using the system environment variable DXROOT created while installing the latest Sandcastle. Pay attention to the fact that DXROOT must be a system but not a user environment variable. Before using Sandcastle from DocMounter, we strongly recommend that you check this in the system properties. The fact is that installations of other tools, such as older Visual Studio SDKs, may have installed outdated versions of Sandcastle and created the corresponding user environment variable DXROOT pointing to "C:\Program Files\Visual Studio 2005 SDK\2007.02\VisualStudioIntegration\Tools\Sandcastle\" or a similar directory. In this case you must delete the user DXROOT environment variable and leave only the system environment variable called DXROOT.

To do this, open the **System Properties** dialog in your OS and click the **Environment Variables...** button on the **Advanced** tab. The **Environment Variables** dialog will be opened. Remove the DXROOT variable from the **User variables for <username>** group if it is there:


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your logon
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Environment Variables
Remove this
User variables for Igor
Variable
Value
DXROOT
C:\Program Files (x86)\Visual Studio 20...
TEMP
%USERPROFILE%\AppData\Local\Temp
TMP
%USERPROFILE%\AppData\Local\Temp
New...
Edit...
Delete
The proper setting
System variables
Variable
Value
CLASSPATH
.;C:\Program Files (x86)\Java\jre6\lib\e...
ComSpec
C:\Windows\system32\cmd.exe
DXROOT
C:\Program Files (x86)\Sandcastle\
FP_NO_HOST_C...
NO
New...
Edit...
Delete
OK
Cancel

How to Start

To document a library or libraries, create a new DocMounter project first. This is done by choosing the **New Project** command in the **File** menu. An **Open Assemblies** dialog allowing you to select the assemblies you are going to document will be displayed. Select the assemblies and click the **Open** button. If you already created XML comment files for your assemblies, DocMounter will also suggest importing of member descriptions from these files. After a new project has been created, you see the topic tree at the left part of the DocMounter window. The root nodes of this tree are namespaces defined in your assemblies. Expand them to find members from your assemblies and write descriptions for them in the editor pane at the right. After that you can build a help file or manual from your project.

A DocMounter project must be saved before the first build operation. DocMounter will suggest doing this automatically for an unsaved project. We also recommend that you check the default paths to the output folders in your project properties and correct them if required. You can do this in the **Project Properties** dialog that can be opened from the **Project** menu or with the corresponding toolbar button. DocMounter works with relative folder paths, and they are specified using the location of your project file as the base path. This is one more reason why you must save project before the first build of help solution.

To build an output file, use the commands from the **Build** menu. For example, if you want to build an MSHC help file, choose the **Build and View Help** command from the **Build** menu or click the corresponding toolbar button (or simply hit F5). Your first MSHC file will be created and displayed in 10Tec Help Viewer. You will see the log for the build process in the **Output** pane displayed automatically when the build process starts.

DocMounter Documentation Tags

Object	MT/ CT tag	Tag format	1) Is an XML Comments tag? 2) MAML equivalent.
		Topic structure	
Normal text section	CT only	<section> <title>*Optional title*</title> <content> *Section content* </content> </section>	XML Comments tag: no MAML: <section>
		Paragraphs	
Paragraph	MT/ CT	<para>*Paragraph text*</para>	XML Comments tag: yes MAML: <para>*text*</para>
		Cross-references	
Member field reference	MT/ CT	<see cref="*member reference*" /> or <see cref="*member reference*">label</see> *"label" is allowed only in MT.* *"member reference" contains full path to the type including its namespace and starts with "T:", "M:", etc. or a special "Overload:" string to point to the page with the list of all overloaded versions.*	XML Comments tag: yes MAML: <codeEntityReference> *T:System.String* </codeEntityReference>
Conceptual topic reference	MT/ CT	<see-topic id="*topic_id*" title="*Topic title*" /> *(The "title" attribute is only for people; it indicates the referenced topic and may differ from the effective title)*	XML Comments tag: no MAML: <link xlink:href="*guid*" />
		Contents	
Code	MT/ CT	<code[language="*lang*"]> *Statements* </code> *(The "language" attribute is optional; typical values for "lang" are "C#" and "VB")*	XML Comments tag: yes MAML: <code language="*lang*"> *Statements* </code>
Image	MT/ CT	 *("filename" is a relative file name from the Media folder of the project; it is also an inline tag placed directly in text – insert paragraph by yourself if you need the image on a separate line)*	XML Comments tag: no MAML: <mediaLink> <image xlink:href="*6be7079d-a9d8-4189-9021-0f72d1642beb*"/> </mediaLink> *(but in MAML GUID and a special media content file should be used to reference images – DocMounter does not need that)*
Parameter	MT only	<paramref name="*name*" />	XML Comments tag: yes MAML: -
Generic parameter	MT only	<typeparamref name="*name*" />	XML Comments tag: yes MAML: -

Lists & Tables			
Bulleted or numbered list	MT/ CT	<list type="bullet|number"> <item> <description>*Item 1*</description> <description>*Item 2*</description> <description>*Item 3*</description> </item> </list>	XML Comments tag: yes MAML: <list class="bullet|ordered|nobullet"> <listItem>*Item 1*</listItem> <listItem>*Item 2*</listItem> <listItem>*Item 3*</listItem> </list> *(items may contain nested tags)*
Definition list (2-column table)	MT/ CT	<list type="table"> <listheader> <term>*header term*</term> <description>*header descr*</description> </listheader> <item> <term>*Term 1*</term> <description>*Term 1 description*</description> </item> <item> <term>*Term 2*</term> <description>*Term 2 description*</description> </item> </list>	XML Comments tag: yes MAML: <table> <title>A *Simple Table*</title> <tableHeader> <row> <entry>*Header 1*</entry> <entry>*Header 2*</entry> </row> </tableHeader> <row> <entry>*Row 1, Cell 1*</entry> <entry>*Row 1, Cell 2*</entry> </row> <row> <entry>*Row 2, Cell 1*</entry> <entry>*Row 2, Cell 2*</entry> </row> </table> *(items can have nested tags)*
Multi-column table	CT	<table> <header> <cell>*header 1*</cell> <cell>*header 2*</cell> </header> <row> <cell>*Row 1, Cell 1*</cell> <cell>*Row 1, Cell 2*</cell> </row> <row> <cell>*Row 2, Cell 1*</cell> <cell>*Row 2, Cell 2*</cell> </row> </table>	XML Comments tag: no MAML: See <table> above
Formatting			
Bold text	MT/ CT	*text*	XML Comments tag: no MAML: <legacyBold>*text*</legacyBold> *(see also other MAML tags for specific items, such as <application>, <ui>, etc.)*

Adding a Custom Root Page to Help

A help file can have a root page used as an entry point to the documentation. As a rule, such a root page is opened automatically when the user opens the help file. DocMounter allows you to add a custom HTML page as the root page to the generated MSHC help file. The root page automatically becomes the page with the highest hierarchy in the Table of Contents. In other words, all top-level conceptual and namespace topics from a DocMounter project will be child pages of the root page.

To add a root page to your help solution, place an XHTML file with the name root_page.htm in the same directory in which the DocMounter project is located. Below is the template of a root page with some meta tags an MSHC root page must have:

```
<?xml version="1.0" encoding="utf-8"?>
<html>
<head>
  <title>ROOT PAGE TITLE</title>
  <meta name="Description" content="ROOT PAGE DESCRIPTION" />

  <meta name="Microsoft.Help.Locale" content="en-us" />
  <meta name="Microsoft.Help.TopicLocale" content="en-us" />
  <meta name="Microsoft.Help.SelfBranded" content="true" />

  <meta name="Microsoft.Help.TocParent" content="-1" />
  <meta name="Microsoft.Help.TocOrder" content="0" />
</head>
<body>

<!— ROOT PAGE CONTENTS -->

</body>
</html>
```

DocMounter will check whether root_page.htm exists during the help build process, and if so, will add it to the generated help file as the root page.

Building Manuals with DocMounter

Your DocMounter project is not only a source for building a help file, its conceptual topics can also be turned into a manual for end users. DocMounter allows you to 'glue' all conceptual topics into one long HTML document and provide it as is for your users or use it as a basis to create a well-formed printable document in a text editing application.

To build an HTML manual from your project, choose the **Build Manual** command from the **Build** menu or hit F6. DocMounter will combine topic in the project's conceptual part into the resulting HTML document and will save it in the dedicated folder for manual output specified in the project properties. This output folder will be opened automatically upon manual build completion.

DocMounter can build manuals of two kinds. It can save the contents of all CTs or only of those which do not have child nodes in the final HTML document. There can be cases when the latter kind is better than the former – for example, you may have TOC-like references to child nodes in parent nodes but omit them in a printable manual.

You can build either of these kinds of manual or even both during one build process. The names of the corresponding HTML documents are set on the **Manual** tab in the **Project Properties** dialog. If the corresponding document name is set (not empty), it is built when you issue the **Build Manual** command.

DocMounter use the <h1> HTML tag for topic titles of the first hierarchy level, <h2> for topics of the second hierarchy level and so on in the built HTML manual. There is an option **Shift All Headings 1 Level Up** to use <h1> for topics on the second hierarchy level, <h2> for topics on the third hierarchy level and so on. You will find the corresponding check box on the same **Manual** tab in the **Project Properties** dialog. As a rule, this option is used in the case if the conceptual part of your project has just one node used as the root for this part of the documentation and you do not want to use the only <h1> tag for it in the resulting document. In this case the title of the root topic will become a paragraph with a special style you can apply another format to if required.

The formatting of the generated HTML document is set in the Manual.css file that is also generated by DocMounter during the manual build process. This is a CSS set you can use as is or adjust to change the formatting of the manual. You will also find the Media subfolder in the manual output folder. All images you refer from CTs are placed in this folder.

Pay attention to the fact that DocMounter does not use Sandcastle to generate HTML manuals. The application processes the contents of CTs and DocMounter tags in them into the corresponding HTML tags using its own algorithm that differs from what Sandcastle does. This means that if you are going to use DocMounter to build both MSHC help and HTML manual from your project, you should not use the extended set of MAML tags in CTs because they will not be processed. Use only the DocMounter tags listed in the DocMounter Documentation Tags topic of this guide.

Generating XML Documentation for Assemblies

.NET Assemblies can be supplied with XML documentation files. The traditional way to create them is to write special XML documentation comments directly in the source code files and generate XML documentation files from Microsoft Visual Studio during compilation. DocMounter provides an alternative way – you can keep your code clean and describe assembly members in a DocMounter project, then generate corresponding XML documentation files from DocMounter.

To generate XML documentation files, use one of the following commands from the **Build** menu: **Build Brief XML Documentation** or **Build Full XML Documentation**. Both commands generate XML documentation files for the assemblies from the current project. The difference between them is that full XML documentation files contain all topic sections from the DocMounter project (summaries, remarks, exceptions, see also) but brief XML documentation files contain only summaries. Brief XML documentation files can be useful if you have very long descriptions of members and do not want to show them in the Visual Studio IntelliSense tooltips.

The XML documentation files are generated in the folders specified in the **Brief XML Documentation** and **Full XML Documentation** properties on the **Folders** tab of the **Project Properties** dialog.

Tips and Notes

Topic contents

Paragraphs

.NET product documentation is enough specific, and you should adhere to the following rules to get the best look of your topics in the result files:

- Use brief 1-paragraph summaries for all members (classes, methods, properties, etc.). This also concerns exceptions you describe for a member. Don't use the <para> tag for these fields.
- Generally a table cell or list item has one paragraph of text, and in this case the <para> tag isn't needed. However, if you create two paragraphs in one of these items using <para>, wrap all other items in the table or list with the <para> tags to have the same padding for all items.

Special XML characters

The "<" and ">" characters are special in XML and are used to mark the DocMounter tags in topic texts. These characters should be coded with their "<", ">" XML equivalents. The same concerns the ampersand character – use "&" instead. All other special characters, including quotes and apostrophes, can be inserted "as is".

Using Sandcastle MAML tags in conceptual topics

Though DocMounter provides you with a set of its standard tags to write both conceptual and member topics, you can still use most of the Sandcastle MAML tags to enhance your conceptual topics. For instance, the MAML <legacyUnderline> inline tag can be used to make some text underlined. But note that all these non-DocMounter tags aren't processed when you build a standalone manual as the Sandcastle compiler isn't used in this process.

Conceptual topic Id's

To implement cross-reference links to conceptual topics, unique topic Id's are used. You can see and change them for conceptual topics in the **Conceptual Topic Properties** dialog. Theoretically you could use any value in this field, but our experience shows that Sandcastle properly works only with Id's in the form of GUID. If you specify your custom id not in this form, you get a warning message like this while building topics:

```
Info: BuildAssembler: Building topic 6371b6a0-3483-4877-b9dc-b1e3f32b8f78
Warn: ResolveConceptualLinksComponent: Invalid conceptual link target 'MainTopicId'.
```

This warning tells us we have an improper link to the topic with Id "MainTopicId" in the topic with Id "6371b6a0-3483-4877-b9dc-b1e3f32b8f78". The link will be properly resolved in the final MSHC file, but it will display the topic Id in square brackets instead of the topic title. In our example, it will be an underlined hyperlink with the text "[MainTopicId]".

Id (GUID) values are case-insensitive.

Project settings

Paths in DocMounter projects

Internally DocMounter works only with absolute paths when it produces output, but all the folder locations you can set in the **Project Properties** dialog can be specified as relative paths. The base path used to calculate production absolute paths is the project file location folder. Thus, the project must be saved first before you can build any contents.

Referenced non-documented assemblies

In many cases you need to document assemblies which use members defined in other assemblies they refer to, but there is no need to include those referenced assemblies in your DocMounter project. To make DocMounter and other related tools used internally (such as the Sandcastle help compiler) working properly in this case, you need to provide them with a way they can resolve links to those external assemblies automatically.

We guess that any standard .NET mechanism to resolve assembly references can be used for that (such as GAC, etc.), but it was not possible to check that for all possible cases. However, as we could see from our tests and all the projects we worked with, there is one way that always works for this situation. You simply place all referenced libraries in the folder(s) of your documented assemblies, and the links are resolved automatically.

UI helper tools

Hotkeys in the project tree

You can use the INSERT and DELETE keys to add/remove nodes in the tree. The INSERT key is used to add new conceptual topics. The DELETE key is used to remove the selected conceptual topic or obsolete member topic.

Reordering conceptual topics

The conceptual topics can be reordered using simple drag-n-drop. If you do it using the left mouse button, the dragged topic is placed before the topic you drop it on. If you do drag-n-drop using the right mouse button, an additional dialog pops up, and you can choose the place to move the dragged topic to – before/after or as a child node for the topic you drop on.

Copy/Paste for See Also

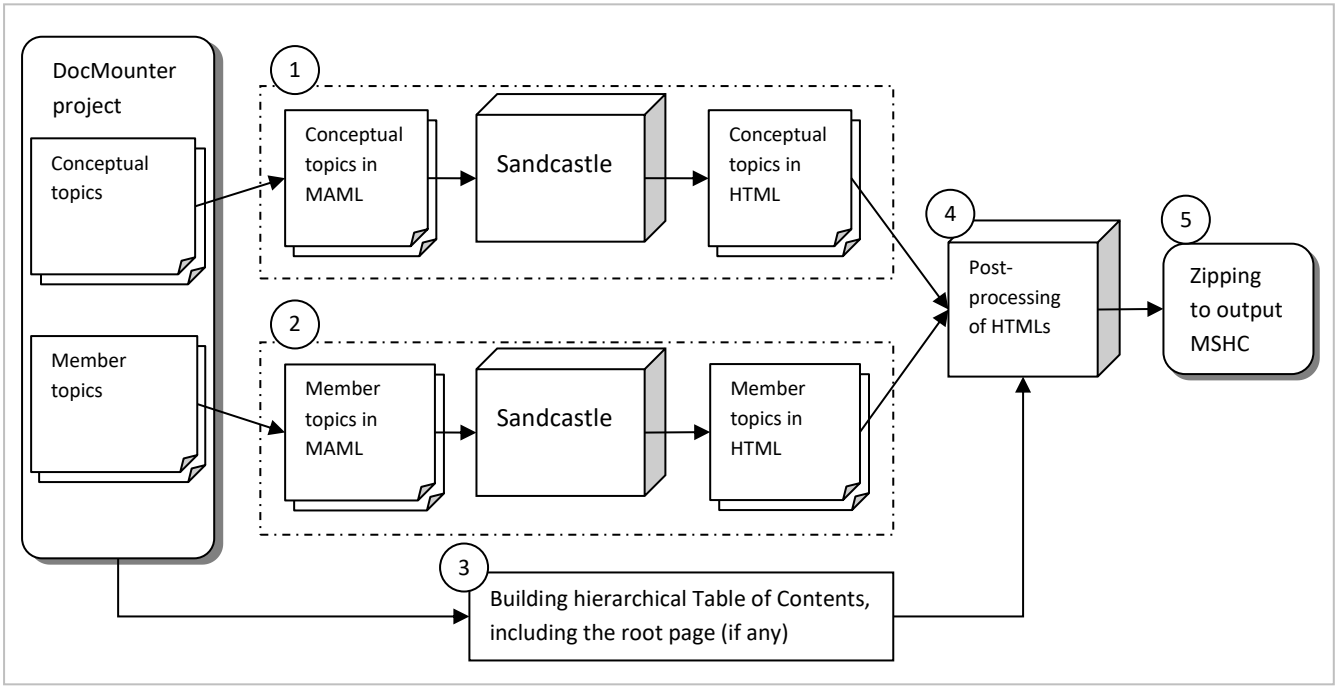
The **See Also** list allows you to copy/paste items between the topics. To do that, first select the required items in the list and press CTRL+C, then open this list for another topic and press CTRL+V. You can also quickly select all members by CTRL+A.

In this operation, the items are copied into a special internal clipboard, and it is not the normal Windows clipboard. This allows you to copy/paste the 'see also' items using the familiar clipboard functionality and work with the standard text clipboard simultaneously and independently.

Build process

The MSHC Build Process

The following scheme outlines the general process of building an MSHC help solution with DocMounter:



DocMounter performs the following main steps when you issue the command to build an MSHC:

1. Conceptual topics are converted into files in the MAML format. These MAML files are compiled to HTML topics by Sandcastle.
2. Similarly to conceptual topics, member topics are also converted into the MAML format and compiled to HTML topics by Sandcastle.
3. DocMounter builds its internal hierarchical Table of Contents. If a root page was specified for the project, it becomes the root record in the TOC.
4. DocMounter inserts special HTML meta tags into the generated topic HTMLs. These meta tags are generally used by help viewers like 10Tec Help Viewer or Microsoft Help Viewer to display help contents. Among these meta tags are tags with topic keywords and tags to show the hierarchical Table of Contents (what DocMounter prepared on the previous stage). The topic HTMLs generated by Sandcastle are also slightly enhanced on this stage to provide best look.
5. The topic HTMLs and all related files, such as images and CSS styles, are packed into the output MSHC file.

Speeding up MSHC build process

The Sandcastle compiler used as a part of the MSHC build process resolves the references to the standard .NET types in your topics automatically. To do that, it loads special XML files from its Data\Reflection\ subfolder. The typical location of this folder is "C:\Program Files\Sandcastle\Data\Reflection\" in a 32-bit Windows or "C:\Program Files (x86)\Sandcastle\Data\Reflection\" in a 64-bit Windows. The size of this folder exceeds 400Mb for the used version of Sandcastle, and most of these files won't be used in your documentation. If you know what standard .NET types are used in your documentation, you can remove all unneeded XML files from the Reflection folder and significantly speed up building of some parts of your documentation. In the simplest case, if you are not interested in links to standard .NET types and/or want to get the fastest build process just to estimate the overall look of output topics, you can remove all XML files from the Reflection folder. But don't delete them at all – just store them in a safe place so you can place them back when you need that.

An example. If you document a class library that performs mathematical calculations or processes strings only, it is enough to have the only file System.xml in the Reflection folder as only the types from the System namespace will be used in your library.

It is also a good practice to have just this one System.xml in the Reflection folder when you start to learn DocMounter because in this case it will build MSHC files much faster.

Viewing results of build processes

DocMounter always tries to open the result of every build process upon successful completion as a visual confirmation of the work done and to provide you with the ability to view the results. If you build an MSHC file, it is opened in 10Tec Help Viewer. For other types of output, manuals and XML documentation files, the corresponding output folder is opened. You also have the ability to open the built help file or output folders for other kinds of output without launching the build process using the corresponding items of the **Tools** menu.

Limitations of the Demo Version

When you generate an MSHC help file with the demo version of DocMounter, the values of the project properties you can view/change on the **Topic Contents** tab in the **Project Properties** dialog are ignored. DocMounter inserts special watermark strings, such as “DOCMOUNTER DEMO”, into the generated pages instead of the actual property values.

The ability to add a root page to the built MSHC file is also not available in the demo version of DocMounter.