

Sandcastle Tools
Welcome
See Also Send Feedback

[This is preliminary documentation and is subject to change.]

Welcome to the Sandcastle Tools help file. The Sandcastle tools are used to create help files for managed class libraries containing both conceptual and API reference topics. API reference topics are created by combining the XML comments that are embedded in your source code with the syntax and structure of the types which is acquired by reflecting against the associated .NET Framework assembly. Conceptual topics are created by converting XML documents that you author containing Microsoft Assistance Markup Language (MAML).

Sandcastle was originally created by Microsoft back in 2006. The last official release from Microsoft occurred in June 2010. Until October 2012, it was hosted at the Sandcastle project site on CodePlex. In October 2012, Microsoft officially declared that they were ceasing support and development of Sandcastle. The Sandcastle tools have been merged into the <u>Sandcastle Help</u> File Builder project and all future development and support for them will be handled at its project site. The Sandcastle tools themselves remain separate from and have no dependency on the help file builder. As such, they can be used in a standalone fashion with your own scripts and build tools if that is your preference.

🗷 Tip

See the <u>Installation Instructions</u> for information about the required set of additional tools that need to be installed, where to get them, and how to make sure everything is configured correctly. The guided installer also provides information on the necessary tools and walks you through the installation steps.

■ See Also

Other Resources

Getting Started Installation Instructions Version History Making a Donation

Send comments on this topic to <u>Eric@EWoodruff.us</u>

[v2.7.3.0] Sandcastle Tools



[This is preliminary documentation and is subject to change.]

This topic contains the following sections.

• <u>Microsoft Public License (Ms-PL)</u>

Microsoft Public License (Ms-PL)

This license governs use of the accompanying software. If you use the software, you accept this license. If you do not accept the license, do not use the software.

1. Definitions

The terms "reproduce," "reproduction," "derivative works," and "distribution" have the same meaning here as under U.S. copyright law.

A "contribution" is the original software, or any additions or changes to the software.

A "contributor" is any person that distributes its contribution under this license.

"Licensed patents" are a contributor's patent claims that read directly on its contribution.

2. Grant of Rights

(A) Copyright Grant - Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royaltyfree copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create.

(B) Patent Grant - Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free license under its licensed patents to make, have made, use, sell, offer for sale, import, and/or otherwise dispose of its contribution in the software or derivative works of the contribution in the software.

3. Conditions and Limitations

- (A) No Trademark License This license does not grant you rights to use any contributors' name, logo, or trademarks.
- (B) If you bring a patent claim against any contributor over patents that you claim are infringed by the software, your patent license from such contributor to the software ends automatically.
- (C) If you distribute any portion of the software, you must retain all copyright, patent, trademark, and attribution notices that are present in the software.
- (D) If you distribute any portion of the software in source code form, you may do so only under this license by including a complete copy of this license with your distribution. If you distribute any portion of the software in compiled or object code form, you may only do so under a license that complies with this license.
- (E) The software is licensed "as-is." You bear the risk of using it. The contributors give no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this license cannot change. To the extent permitted under your local laws, the contributors exclude the implied warranties of merchantability, fitness for a particular purpose and non-infringement.

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools

Getting Started

See Also Send Feedback

[This is preliminary documentation and is subject to change.]

The topics in this section will help you install all of the required tools, become familiar with the Sandcastle Help File Builder, and create your first project.

- Installation Instructions
- Walkthrough: Enabling and Using XML Comments
- Documenting Web Projects
- The following article is outdated now but does contain some useful information: <u>Taming Sandcastle: A .NET Programmer's Guide to</u> <u>Documenting Your Code</u>
- Sandcastle Wall Chart

∃ See Also

Other Resources

Welcome Version History Frequently Asked Questions (FAQ)

Send comments on this topic to <u>Eric@EWoodruff.us</u>

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools
Installation Instructions
See Also Send Feedback

[This is preliminary documentation and is subject to change.]

This topic describes the additional tools that are required to use the Sandcastle tools along with some general configuration information.

- Guided Installation
- <u>Requirements</u>
- <u>Recommended Additions</u>
- Optional Items
- <u>See Also</u>

Guided Installation

Due to the number of products that need to be installed to get a working build environment, a guided installation package has been created to simplify the process. The guided installer contains most of the necessary parts and various optional components. For those parts that cannot be distributed such as the help compilers, it contains instructions on where to get them and how to install them if you need them.

Download the latest Sandcastle Help File Builder Guided Installer

When using the guided installation, you can skip the **Requirements** section below as it takes care of all of the required steps. See the **Recommended** and **Optional** sections for additional tools not included in the guided installation that you may find useful.

Requirements

The latest version of the Sandcastle tools can always be found at the <u>Sandcastle Help File Builder</u> CodePlex project site. In addition, you will need the following:

• The <u>.NET Framework 4.0</u>.

In order to use the Sandcastle tools, the following additional tools are required based on the types of help files that you want to produce. Each must be installed prior to building a help file of that particular type:

- The <u>HTML Help Workshop</u> for building HTML Help 1 (.chm) help files.
- The MS Help 2 compiler for building MS Help 2 (.HxS) help files. This must be downloaded and installed as part of the <u>Visual Studio</u> <u>2005 SDK v4.0</u> or the <u>Visual Studio 2008 SDK v1.0</u>. Note that the latest version of the Visual Studio 2008 SDK does not contain the Help 2 compiler so you must download the 1.0 version. The latest release can be installed after it if you need it.
- The Help Library Manager and Help Library Agent for installing and viewing MS Help Viewer (.mshc) help files. These are installed as part of Visual Studio.

The tools can be installed in any order. However, be aware that if installing the Visual Studio 2005 or 2008 SDK to obtain the Help 2 compiler that both contain outdated versions of Sandcastle. This can cause the DXROOT environment variable used to locate the Sandcastle tools to be incorrect. After installing either SDK, do the following:

- 1. Check the <u>user</u> environment variables. If you see a copy of the DXROOT environment variable there, delete it.
- 2. If there is a <u>user</u> copy of the PATH environment variable, remove any path to the SDK version of Sandcastle that it contains. If that is all it contains, it can be deleted.

3. Check the <u>system</u> environment variables to ensure that a DXROOT environment variable does exist and that it points to the correct location of the Microsoft Sandcastle tools (**not** the help file builder). Usually this will be *C:\Program Files (x86)\Sandcastle* unless you installed it in an alternate location.

🗹 Note

You may need to reboot the system in order for any environment variable changes to take effect.

Recommended Additions

The following tools are optional but you will find them useful and it is highly recommended that they be installed as well.

- To view MS Help 2 files without registering them, you can use the <u>H2</u> <u>Viewer</u> (free for personal use).
- To view MS Help Viewer files with a full table of content and index support rather than in the default browser, you can use the <u>H3 Viewer</u> (free for personal use).
- To register MS Help 2 files both for testing and as part of your product installation, download the shareware tool <u>H2Reg.exe</u> from <u>The Helpware Group</u>.
- If you are going to document website projects, install the <u>Custom Web</u> <u>Code Providers</u> included as part of the Sandcastle tools installer. These make it easy to document website projects by extracting the XML comments files to a location of your choosing when the website is deployed.

Optional Items

The following items are not required to use Sandcastle or the help file builder but you may find them useful.

- If you are documenting .NET 1.1 applications and need the Help 2 compiler tools, you can download the <u>Visual Studio .NET Help</u> <u>Integration Kit 2003</u>.
- For a Visual Studio add-in used to create XML comments automatically, check out SubMain's <u>GhostDoc</u>.
- If you need to convert HTML additional content topics to MAML, you can use the **HTML to MAML Converter** utility included with the Sandcastle tools installer.

■ See Also

Other Resources

Getting Started

Send comments on this topic to <u>Eric@EWoodruff.us</u>

[v2.7.3.0] Sandcastle Tools



 Sandcastle Tools
 Walkthrough: Enabling and Using XML Comments
 See Also Send Feedback

[This is preliminary documentation and is subject to change.]

This walkthrough will describe the steps needed to enable XML comments file output in your Visual Studio projects, provides information on where to get more information on using XML comments to decorate your code, and describes how to open help file builder projects from within Visual Studio.

- Enabling XML Comments File Generation
- Decorating Your Code
- <u>See Also</u>

Enabling XML Comments File Generation

In order to create a help file that contains reference content (API documentation), you must enable XML comments in your Visual Studio projects so that an XML file is generated to contain them when the project is built.

🗹 Note

If you are trying to document a website project, see the <u>Documenting</u> <u>Web Projects</u> topic for some necessary information as they are unlike other Visual Studio projects and require additional steps in order to generate XML comments files.

🗹 Tip

If not using the Visual Studio IDE, the various managed language compilers support a */doc* command line option that will produce the XML comments file. See your language compiler command line option documentation for details.

All Projects Except Managed C++ Projects

- 1. In the Solution Explorer, right click on the project and select **Properties**.
- 2. Select the **Build** property page.

☑ Note

The XML comments filename is a configuration option. As such, you can either select the **All Configurations** option at the top of the page to set the XML comments options for all configurations at the same time or select each configuration individually and repeat the next steps for each one.

3. In the **Output** section, check the checkbox next to the **Xml documentation file** text box and specify a name for the XML file. Although not required, a common convention is to name the XML comments file after the related assembly (except with a *.xml*

extension). The assembly name can be found on the **Application** property page.

4. If you have a solution with multiple projects that need to be documented, repeat the above steps for each project in the solution. It is recommended that you give each project's XML comments file a unique name.

If documenting a managed C++ project, the procedure differs and you need to follow these steps instead.

Managed C++ Projects

- 1. In the Solution Explorer, right click on the project and select **Properties**.
- 2. Expand the **Configuration Properties** category and then expand the **C/C++** sub-category and select the **Output Files** option below it.

☑ Note

The XML comments file option is a configuration option. As such, you can either select the **All Configurations** option at the top of the dialog box to set the XML comments options for all configurations at the same time or select each configuration individually and repeat the next step for each one.

- 3. In the **Output Files** options, change the **Generate XML Documentation Files** option to **Yes (/doc)**.
- 4. By default, the comments file is named after the project target with a *.xml* extension and is placed in the target folder. If you want to change the name, select the **XML Document Generator** category below the **Configuration Properties** category and change the filename using the **Output Document File** property.
- 5. If you have a solution with multiple projects that need to be documented, repeat the above steps for each project in the solution. If you explicitly specify a name for the comments file, it is recommended that you give each project's XML comments file a unique name.

Once the above has been done, Visual Studio will create the XML comments file each time the project is built.

Decorating Your Code

In addition to setting the project option to create the file, you must also add XML comments to your source code. At a minimum, you should add a **<summary>** tag to each public type and to the public and protected members of those types. There are many other tags available that will help improve the quality of your documentation.

For a comprehensive list of XML documentation elements and a description of their attributes and usage, see the **Sandcastle XML Comments Guide** installed as part of the Sandcastle tools. Basic information on XML comments and how to use them can also be found at either <u>XML</u> <u>Documentation Comments (C# Programming Guide)</u> or <u>Documenting Your</u> <u>Code with XML (Visual Basic)</u>.

Although it is not mentioned in the XML tag documentation, you can use HTML markup within your XML comments to provide additional formatting, embed images, etc.

■ Next Steps

Now that you have enabled XML comments in your project and added some comments to your code, you are ready to use the Sandcastle tools to generate documentation.

■ See Also

Other Resources

Getting Started

Documenting Web Projects

Taming Sandcastle: A .NET Programmer's Guide to Documenting Your Code

Sandcastle Wall Chart

Send comments on this topic to <u>Eric@EWoodruff.us</u>

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools
 Documenting Web Projects
 See Also Send Feedback

[This is preliminary documentation and is subject to change.]

This topic contains the following sections.

- Documenting Web Application and Web Service Application Projects
- Documenting Website and Web Service Website Projects
- <u>Automating the Process</u>
- Usage Notes and Known Issues
- How The Code Providers Work
- <u>Conclusion</u>
- <u>See Also</u>

A question that comes up with some regularity is how to produce a help file from the XML comments in the code for a web project using Sandcastle. There are two types of web project: a *web application* project and a *website* project (both include a web service and an AJAX-enabled variant of the project type). A web application project is the simplest to document as it generates a single assembly with a fixed name much like any other non-web project. A website project, however, requires some extra effort to document and has a number of limitations:

- It does not necessarily generate a single assembly.
- It only generates assemblies when you actually publish the project.
- The assembly names may vary from build to build depending on the options you use when publishing the website.
- There is no project-level option to produce XML comments files. Instead, you have to add a section to the *Web.config* file to produce them.

- Even after adding the option to produce XML comments files, there are limitations:
 - If you specify a single comments filename, only the comments from the last assembly compiled will be saved. The VB.NET compiler has a /doc+ option to work around this issue. However, the C# compiler does not so this makes it impossible to fully document a C# website project.
 - The comments files produced using the VB.NET /doc+ option are not placed in the folder to which the project was published. Instead, they are left in the system's temporary ASP.NET folder for the related .NET version (i.e. *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET files* so you must locate and retrieve them after each build.

To work around the limitations present when documenting website projects, you can use the custom code providers made available with the Sandcastle tools installer (one for C# and one for VB.NET). By using these code providers, you can specify a folder to which the XML comments files should be copied. This saves you from having to locate and copy them manually and solves the issue of the C# compiler not being able to output one comments file per assembly. Below are instructions for documenting each type of web project and information about how the code providers work.

Documenting Web Application and Web Service Application Projects

As noted, web applications are the easiest to document and do not require the use of the custom code providers. These projects are created in Visual Studio by selecting **File** | **New** | **Project** and then selecting **ASP.NET Web Application** or **ASP.NET Web Service Application** from the **Web** category of your preferred language. Note that these options are only available in Visual Studio 2005 after you install Service Pack 1.

You must enable the XML comments option in your application projects so that an XML comments file is produced when the projects are built. This is done as follows:

- 1. In the Solution Explorer, right click on the project and select **Properties**.
- 2. Select the **Build** property page (C#) or the **Compile** property page (VB.NET).
- 3. The XML comments filename is a per configuration option. As such, you can either select the **All Configurations** option at the top of the page to set the XML comments options for all configurations at the same time or select each configuration individually and repeat the next step for each one.
- 4. For C#, in the **Output** section, check the checkbox next to the **XML documentation file** text box and specify a name for the XML comments file. Although not required, a common convention is to name the XML comments file after the related assembly except with a *.xml* extension. The assembly name can be found on the **Application** property page.

For VB.NET, just check the **Generate XML documentation file** checkbox. It names the file after the assembly automatically.

5. If you have a solution with multiple web application or web service application projects that need to be documented, repeat the above steps for each project in the solution. If using the Sandcastle Help File Builder, be sure to give each project's XML comments file a unique name as they are copied to a common location for the build. Identically named files would overwrite each other and the documentation in the overwritten files would not appear in the help file.

Once the above has been done, Visual Studio will create the XML comments file each time the project is built. It is placed in the .*bin* folder along with the assembly. These can be used in conjunction with your preferred documentation tool to produce a help file.

Documenting Website and Web Service Website Projects

These projects are created in Visual Studio by selecting **File** | **New** | **Web Site** and then selecting **ASP.NET Web Site** or **ASP.NET Web Service** from the template dialog box. In order to create XML comments files, you must add the custom code providers to the *Web.config* file and publish the website. This is done as follows:

- 1. If your project does not yet contain one, add a *Web.config* configuration file.
- 2. Add the following <system.codedom> section to the <configuration> section as shown in the example below. You only need to add the <compiler> element for the language that you are using but it will not hurt anything if they are both present.

```
Example Code Provider Configurations
<configuration>
  <system.codedom>
    <compilers>
      <!-- For C# -->
      <compiler language="c#;cs;csharp" extensic
        compilerOptions="/docpath:C:\Publish\Doc
        type="EWSoftware.CodeDom.CSharpCodeProvi
              EWSoftware.CodeDom, Version=1.1.0.
              PublicKeyToken=d633d7d5b41cbb65">
        <!-- NOTE: Change value to "v3.5" for .N
        <providerOption name="CompilerVersion" v</pre>
      </compiler>
      <!-- For VB.NET -->
      <compiler language="vb;vbs;visualbasic;vbs
        compilerOptions="/docpath:C:\Publish\Doc
        type="EWSoftware.CodeDom.VBCodeProviderw")
              EWSoftware.CodeDom, Version=1.1.0.
              PublicKeyToken=d633d7d5b41cbb65">
        <!-- NOTE: Change value to "v3.5" for .N
```

```
<providerOption name="CompilerVersion" v
</compiler>
</compilers>
</system.codedom>
</configuration>
```

Instead of a /doc option, the custom code providers use a custom /docpath option in the compilerOptions attribute that specifies the fully qualified folder name into which the XML comments files are placed. The folder will be created by the code provider if it does not exist. Any other compiler options that you might need can also be specified in the attribute value separated by a space.

3. In order to use the code providers, the ASP.NET compiler must be able to find them. You can either copy the EWSoftware.CodeDom.dll assembly into your project's .\bin folder or you can register it in the Global Assembly Cache (GAC) so that it is available to any project. To do so, use the gacutil command line tool as follows replacing the path to the assembly to match the installed location on your system:

gacutil -i \DotNet\Web\WebCodeProviders\Bin\Rele

To remove the assembly from the Global Assembly Cache, use the **gacutil** command line tool as follows:

Copy

gacutil -u EWSoftware.CodeDom

You can find a copy of the tool in the Visual Studio installation folder in the .*SDK**v*2.0*bin* subfolder.

Once the above has been done, you can publish your website. Right click on the project in the Solution Explorer, select **Publish Web Site**, set the options, and click OK to publish it. When finished, look in the location you

specified in the /docpath option and you will find one XML comments file for each assembly that was built for your website. These can be used in conjunction with the assemblies and your preferred documentation tool to produce a help file.

As an alternative to publishing the website, you can use a web deployment project. Add-ins are available at the following locations:

- Visual Studio 2005 Web Deployment Add-In
- Visual Studio 2008 Web Deployment Add-In

The <u>Sandcastle Help File Builder</u> version 1.8.0.0 and later supports wildcard documentation sources. As such, just add one entry with the path set to [*appPath*]*bin**App_*.dll* where [*appPath*] is the path to the published website and another entry with the path set to [*docPath*]*App_*.xml* where [*docPath*] is the path to the XML comments files.

Since the code providers are not needed after the website is published, you can delete the system.codedom section from the published *Web.Config* file and delete the assembly from the .*bin* folder if you added it to the project for publishing.

Automating the Process

Using the **aspnet_compiler** tool, it is possible to automate the publishing of a website project. The tool is located in the .NET Framework folder. For .NET 2.0 through 3.5, this is usually

C:\Windows\Microsoft.NET\Framework\v2.0.50727. To run it, issue a command similar to the following (lines wrapped for display purposes):

C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspne -f -v /MyWebSite

-p C:\Path\To\MyWebsite C:\Publish\MyWebSite

The -f option tells it to delete any existing content in the target folder (*C:\Publish\MyWebsite* in the example above). The -v [path] option tells it the virtual path of the website to compile (i.e. the path you use to load it in the web browser). The -p [path] option tells it the physical location of the website related to the virtual path to compile. Using a script, you could for example publish the website, build the help file, replace the *Web.config* in the published website with one that has had the system.codedom section removed, and delete the code provider assembly if it was copied locally. This is left as an exercise for the reader. The web deployment add-ins noted above support post-build events so they may provide better support for such tasks.

Usage Notes and Known Issues

The following are some known issues and things to be aware of when using the custom code providers.

- If any part of the XML comments file path contains spaces, enclose it in " entities (i.e. compilerOptions="/docpath:"C:\My Path With Spaces"").
- Do not use a relative path in the /docpath option. Always use a fully qualified path. When using the project's publish option, the ASP.NET compiler always runs in the .NET Framework folder so your XML comments files will end up in a location relative to it rather than your project's folder if you do.
- You can specify any folder except one that is under the location to which you publish the website. The act of publishing the website clears the destination folder and, since the comments files are produced first, you will lose them if you place them under it.
- The code providers will delete any "App_*.xml" files from the specified folder on the first use. This prevents lots of files from accumulating in the comments file folder that contain duplicate information when not using fixed names. It also means that you should publish each website's comments files to a unique folder so that they are not lost if you are publishing and documenting multiple projects.
- Unless you are using fixed names when publishing, the assembly and comments filenames will change each time you publish the website. As such, use wildcards (App_*.dll and App_*.xml) to specify the files to use in your preferred documentation tool.
- When using the VB.NET custom code provider, the commonly imported namespaces such as System and Microsoft.VisualBasic are not included automatically for some unknown reason. The custom code provider will add a /imports command line option automatically for most of the

common namespaces to work around the problem. If publishing fails due to undefined types when using the custom code providers, simply add a /imports: [Namespace] option to the compilerOptions attribute where "[Namespace]" is the missing namespace. You can generally identify namespaces that may cause problems by checking Web.Config for <namespace> elements in the <pages> configuration section. See below for a list of the common namespaces added automatically by the VB.NET custom code provider. Another workaround to this issue is to add Imports statements to the code for each of the required namespaces rather than relying on the automatically included references.

 Related to the item above, when the custom code providers are active in the *Web.Config* file, you may see lots of errors indicated by the red squiggly underline related to missing types while viewing VB.NET code. During development, you can comment out the custom code providers so that this does not happen. As noted, the other workaround is to add an Imports statement to the code that includes the missing namespace that contains the types.

How The Code Providers Work

The custom code providers are actually quite simple. Both the C# and VB.NET custom code providers are identical with the exception of their base class and an extra set of compiler options passed by the VB.NET version. The C# version is shown below.

Example Code Provider

```
public class CSharpCodeProviderWithDocs : CSharpCod
{
    public override CompilerResults CompileAssembly
      CompilerParameters options,
      params CodeCompileUnit[] compilationUnits)
    {
        CodeProviderHelper.ReplaceDocPathOption(opt
        return base.CompileAssemblyFromDom(options,
            compilationUnits);
    }
    public override CompilerResults CompileAssembly
      CompilerParameters options, params string[] f
    {
        CodeProviderHelper.ReplaceDocPathOption(opt
        return base.CompileAssemblyFromFile(options
    }
    public override CompilerResults CompileAssembly
      CompilerParameters options, params string[] s
    {
        CodeProviderHelper.ReplaceDocPathOption(opt
        return base.CompileAssemblyFromSource(optio
    }
}
```

The derived version simply overrides three key methods and, prior to calling the base implementation, calls the CodeProviderHelper.ReplaceDocPathOption method which

takes care of updating the compiler options to produce a unique XML comments file for the assembly in the specified folder. It is shown below.

Replace Doc Path Option Method

```
internal static void ReplaceDocPathOption(
 CompilerParameters options,
 string[] additionalOptions)
{
    Match m;
    string docPath, docFile;
    // Replace /docpath with /doc
    if(!String.IsNullOrEmpty(options.CompilerOption
    ł
        m = reDocPathOpt.Match(options.CompilerOpti
        if(m.Success)
        {
            docPath = m.Groups[1].Value.Replace("\"
                String.Empty);
            if(!Directory.Exists(docPath))
            {
                Directory.CreateDirectory(docPath);
                docFilesPurged = true;
            }
            else
                if(!docFilesPurged)
                {
                    // Purge the comments files fro
                    // on the first call.
                    foreach(string file in Director
                      docPath, "App_*.xml"))
                        File.Delete(file);
                    docFilesPurged = true;
                }
```

```
docFile = Path.Combine(docPath, Path.Ge
        Path.ChangeExtension(options.Output
        ".xml")));
        options.CompilerOptions = String.Format
        "{0} /doc:\"{1}\"", reDocPathOpt.Re
        options.CompilerOptions, String.Emp
      }
   }
}
// Append the other options
if(additionalOptions != null && additionalOptio
        options.CompilerOptions = String.Concat(
            options.CompilerOptions, " ", String.Jo
            additionalOptions));
}
```

A regular expression is used to locate the /docpath option. If found, it extracts the folder name and either creates it if it does not exist or deletes all existing *App_*.xml* files in it the first time it is called. It then combines the path with the output assembly's name and a *.xml* extension and replaces the /docpath option with a /doc command line option. If any additional options are passed to the method, they are also appended to the command line options. The VB.NET code provider passes in an extra /imports command line option to specify the following default import namespaces to work around the issue noted earlier:

- System
- System.Collections
- System.Collections.Generic
- System.Collections.ObjectModel
- System.Configuration
- System.Data
- System.Web
- System.Web.Configuration

- System.Web.UI
- System.Web.UI.HtmlControls
- System.Web.UI.WebControls
- System.Web.Util
- System.Xml
- Microsoft.VisualBasic

Conclusion

By using the custom code providers and the instructions above, you can now easily produce XML comments files for website projects in order to create help files for them.
■ See Also

Other Resources

Getting Started Walkthrough: Enabling and Using XML Comments Sandcastle Help File Builder

Send comments on this topic to <u>Eric@EWoodruff.us</u>

[v2.7.3.0] Sandcastle Tools



[This is preliminary documentation and is subject to change.]

Select a topic below to learn more about Sandcastle and its various parts.

Sandcastle Tools

- General Information
- <u>Architecture</u>
- TODO: More information will be added as time permits.

⊠ Note

Author Credit: The information and images in the General Information and Architecture topics was originally created by Dave Sexton as part of the wiki content for the Sandcastle Styles project on CodePlex.

■ See Also

Other Resources

Getting Started

Send comments on this topic to <u>Eric@EWoodruff.us</u>

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools
General Information
See Also Send Feedback

[This is preliminary documentation and is subject to change.]

Sandcastle is not a single executable program but is actually a set of command-line programs and XSL transformations that are executed in a specific sequence to produce a set of files that can be used to create a help file. It is extremely flexible and offers many configurable features and the build process can be changed depending upon the desired output format and presentation style.

As the tools are command-line based, they can be used in various ways such as in batch files, MSBuild projects, and PowerShell scripts. One downside to its command-line based nature and the flexibility it offers with its various configuration files and presentation styles is a steep learning curve. Using the tools in their native form can be a daunting process so a good front end that hides the various details can greatly simplify the process of using them.

An example GUI and some command line scripts are provided with the Sandcastle tools but they provide limited functionality and do not expose enough of the configuration options to make them useful beyond a few simple examples. It is highly recommended that you use an open source front end such as the <u>Sandcastle Help File Builder</u> that provides both a standalone GUI as well as Visual Studio integration. It provides the usual project management options and also many more features that can be used to produce better looking help files and to author conceptual help topics. It also hides the complexity of the Sandcastle tools and lets you focus on the task of authoring the help content.

Sandcastle produces two types of help topic, each using a different build process. The results can be combined into a single help file.

• **Conceptual Content** - This topic is an example of conceptual content. It is written using Microsoft Assistance Markup Language (MAML) and is used to create topics for general conceptual information, walkthroughs, how-tos, etc. For more information about MAML and how to use it, see

the **Sandcastle MAML Guide** that is available for installation along with the Sandcastle tools.

• API Reference Content - Reference content is generated by using reflection to generate information about one or more managed assemblies, the types it contains, and their members and combining that information with the related XML comments extracted from the source code. This produces topics with a format similar to those you see on MSDN. For more information about XML comments and how to use them, see the Sandcastle XML Comments Guide that is available for installation along with the Sandcastle tools.

As noted above with regard to API reference content, Sandcastle is primarily for use in documenting managed code assemblies that utilize the standard .NET Framework, Silverlight, the .NET Portable framework, or the .NET for Windows Store Apps framework. However, if you can produce a reflection information file for other API types such as a JavaScript library, it is possible to document them as well. For example, plug-ins exist for the <u>Sandcastle Help</u> <u>File Builder</u> that allow you to document JavaScript libraries and XML schemas. Currently, there is no means of documenting unmanaged native code assemblies with Sandcastle. However, if information about the unmanaged API could be converted to a reflection information file, it would be possible to produce a help file.

It is important to note that Sandcastle itself does not generate the end result, a compiled help file such as a Help 1 (CHM), MS Help 2 (HxS), or an MS Help Viewer (MSHC) file. Instead, it relies on third-party tools such as the Help 1 compiler in the HTML Help Workshop, the Help 2 compiler in the Visual Studio SDK, or the third-party front ends to compress the content into an MS Help Viewer archive. The HTML output by Sandcastle is ready to use for website output but it lacks a front end to display the table of content in any meaningful way and does not provide for searching or a keyword index. Third-party tools such as the Sandcastle Help File Builder can fill these gaps for you.

∃ See Also

Other Resources

Sandcastle Tools

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools
Architecture
See Also Send Feedback

[This is preliminary documentation and is subject to change.]

This topic gives an overview of the Sandcastle architecture.

- Sandcastle Layers
- Modes of Operation
 - <u>Reference Mode</u>
 - <u>Conceptual Mode</u>
- <u>Presentation Styles</u>

■ Sandcastle Layers

At a high level, Sandcastle can be seen as a set of tools that transforms raw input into documentation output.



Inputs of various types are supported such as managed assemblies and raw XML documentation files, which are typically compiler-generated along with the assemblies, as well as MAML topics and additional content such as images and code snippets. With this input, Sandcastle performs a highly configurable set of transformations to produce web-ready HTML files as output.

Sandcastle's output can be used as-is by hosting the generated HTML files

and their dependencies (namely, CSS and images) on a web server. Alternatively, and more common, the generated HTML files can be passed as input to help compilers to produce compiled help in various forms. Compilers are additional tools (downloaded separately) that aggregate all of the HTML files produced by Sandcastle to create compiled help files. These are files that can be downloaded and viewed in a special desktop application such as the HTML Help Viewer or Microsoft Document Explorer. Typically, compiled help files are distributed with the products being documented, or at least made available as a separate download.

■ Modes of Operation

There are two modes of operation supported by Sandcastle: Reference and Conceptual. The differences between them can be found in both the input and output of Sandcastle. To provide a quick definition of each, consider reference documentation to provide information about the physical properties of a managed library or application, such as the actual namespaces, types, members and method arguments, and consider conceptual documentation to provide instructional information about the overall design and conceptual properties of a managed library or application, such as how to accomplish specific usage scenarios and providing an in-depth analysis of specifically complex APIs. More distinctions between the two modes can be found in their corresponding sections below.



🗹 Note

Although built separately, the outputs from the reference and conceptual builds are compatible and are typically combined into a single help file.

Reference Mode

To produce reference documentation, Sandcastle requires managed assemblies as input. The **MRefBuilder** command-line tool is used to analyze the physical properties of the assemblies, as well as their dependencies, ultimately producing an XML reflection file that becomes input for the next steps of the process. This XML reflection file contains all of the information that is required to document the physical properties of the input assemblies.

Sandcastle also accepts XML documentation files as input. XML documentation files are typically generated by code compilers from the triple-slash (C#) or triple-apostrophe (VB) comments found in source code files, at the same time that their corresponding assemblies are compiled. XML documentation files contain specific identifiers that allow Sandcastle to correlate XML documentation for each API to the same API in the XML reflection file that is produced by **MRefBuilder**. Additionally, reference mode supports shared inputs of various types such as images and code snippets to be included in the documentation.

The output of Sandastle in reference mode is created by the **BuildAssembler** tool with a specific configuration. It produces a set of HTML files that document the physical structure of the XML reflection file (generated by **MRefBuilder**) passed in as input. XML documentation comments that are discovered during the process are also injected into the HTML files in the appropriate places; e.g., contents of the <summary> elements appear at the top of the HTML file, while <remarks> contents appear after the syntax section.

Sandcastle also automatically generates a table of contents (TOC) and a keyword index based on the XML reflection file. Each auto-generated API topic also includes appropriate hyperlinks. Typically, pseudo-topics are automatically generated as well; e.g., a topic containing a list of all the types in a particular namespace, and another topic containing a list of all properties in a particular type, etc.

Conceptual Mode

To produce conceptual documentation, Sandcastle depends on a special XML schema called Microsoft Assistance Markup Language (MAML). The input to Sandcastle in conceptual mode must be a set of XML documentation files written against the MAML schema. XML companion files can be provided as input to specify metadata for each MAML topic; e.g., index keywords. Additionally, conceptual mode supports shared inputs of various types such as images and code snippets to be included in the documentation.

The output of Sandcastle in conceptual mode is created by the

BuildAssembler tool with a specific configuration. It produces a set of HTML files that are created by transforming the individual MAML files, where one HTML file is created per MAML file. The HTML files display the data in each MAML topic according to the document type of that topic; e.g., all **How-To** topics will have an introduction and at least one procedural section that contains an ordered list of steps for the reader to follow. Most, but not all MAML topics also contain a **See Also** section at the bottom. Special MAML document types, such as **Glossary**, use a very different type of transformation from the other document types.

It is up to the author to specify a table of contents (TOC) in conceptual mode. Likewise, a keyword index is not automatically generated, so it is the author's responsibility to specify relevant keywords for each MAML topic using an XML companion file.

Presentation Styles

The presentation styles in Sandcastle contain all of the aesthetic resources that are required to build HTML documentation in different styles. Each presentation style includes a set of XSLT files, XML files, CSS files, and images. The set of XSLT files is executed once for each topic that **BuildAssembler** processes to produce an individual HTML file as output. Each presentation style can have a different set of XSLT files (although usually similar in many ways); therefore, each presentation style also must provide a custom configuration for the **BuildAssembler** tool. The XML files contain look-up information, such as localizable textual content. The CSS files and images are referenced by the HTML files that are generated, so technically they are just static output.

∃ See Also

Other Resources

Sandcastle Tools

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



[This is preliminary documentation and is subject to change.]

The following are some common problems and solutions along with some known issues with Sandcastle and help files that you may encounter.

TODO: This will be filled in as time permits.

■ See Also

Other Resources

<u>Getting Started</u> <u>Links to Resources</u>

Send comments on this topic to Eric@EWoodruff.us [v2.7.3.0] Sandcastle Tools



Sandcastle Tools
Links to Resources
See Also Send Feedback

[This is preliminary documentation and is subject to change.]

This topics contains links to various resources that you may find useful.

- Information and Guides
- Discussion Groups, Blogs, Etc.
- Build and Project Management Tools
- Third-Party Build Components
- <u>See Also</u>

■ Information and Guides

The most recent version of this help file and other guides on MAML and XML comments can be download from the <u>Sandcastle Help File Builder</u> project on CodePlex.

Discussion Groups, Blogs, Etc.

- To discuss Microsoft's Sandcastle, suggest new features, report bugs in it, and request help using it, visit the <u>Sandcastle Help File Builder</u> CodePlex project.
- For discussion and questions about help systems in general, visit the <u>Visual Studio General Questions forum</u>.
- The retired <u>Microsoft Developer Documentation and Help System</u> <u>forum</u> may also provide information on older help systems.
- <u>The Sandcastle Blog</u> provides news about Sandcastle and blog entries describing how its various features work. That information will eventually be merged into this help file.

Build and Project Management Tools

- The <u>.NET 2.0 Framework</u>.
- The <u>.NET 3.5 Framework</u>.
- The <u>.NET 4.0 Framework</u>.
- The latest version of the Sandcastle tools (currently the April 2013 v2.7.3.0 release as of April 7, 2013). These are distributed along with the <u>Sandcastle Help File Builder</u>.
- The <u>HTML Help Workshop</u> for building HTML Help 1.x (.CHM) help files.
- The HTML Help 2.x compiler for building HTML Help 2.x (.HxS) help files. This must be downloaded and installed as part of the <u>Visual Studio 2005 SDK</u> or the <u>Visual Studio 2008 SDK</u>.
- If you are documenting .NET 1.1 applications and need the Help 2 compiler tools, you can download the <u>Visual Studio 2003 Help</u> <u>Integration Toolkit</u>.

Third-Party Build Components

- The <u>Sandcastle Help File Builder</u> contains the following custom build components:
 - The Code Block Component used to colorize code blocks and add optional line numbering and collapsible sections.
 - The Help Attributes Component used to insert additional HTML Help 2 attributes into reference topics.
 - The IntelliSense Component used to generate XML comments files for use as IntelliSense in Visual Studio.
 - The Show Missing Documentation Component used to add "missing documentation" notes to reference topics and to autodocument constructors.
 - A set of cached build data components that cache key information such as comment, reflection index information, and MSDN URLs across builds in order to speed them up.

■ See Also

Other Resources

<u>Welcome</u>

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools
Making a Donation
Send Feedback

[This is preliminary documentation and is subject to change.]

Several people have asked how to make a donation to support the Sandcastle Help File Builder project. If you would like to do so, you can make a donation of any amount you like by clicking on the PayPal donation button below. If you would prefer to donate via check or money order (USD), please send me an e-mail at Eric@EWoodruff.us and I will reply with my mailing address.

The above method of donating is secure. PayPal guarantees your privacy and security. I never receive details of your payment other than the amount, your name, e-mail address, and any optionally entered information.

Thanks to those of you that have made a donation. It is much appreciated!

[v2.7.3.0] Sandcastle Tools

Send comments on this topic to Eric@EWoodruff.us



[This is preliminary documentation and is subject to change.]

The topics in this section describe the various changes made to the Sandcastle tools over the life of the project.

Version History

Select a version below to see a description of its changes.

- <u>Version 2.7.3.0, April 2013</u>
- <u>Version 2.7.2.0, December 2012</u>
- <u>Version 2.7.1.0, October 2012</u>
- <u>Version 2.7.0.0, April 2012</u>

Send comments on this topic to <u>Eric@EWoodruff.us</u> [v2.7.3.0] Sandcastle Tools



Sandcastle Tools
 Version 2.7.3.0, April 2013
 See Also Send Feedback

[This is preliminary documentation and is subject to change.]

Version 2.7.3.0 was released on April 7th, 2013.

- <u>MRefBuilder Tools</u>
- BuildAssembler Tools
- Build Components
- <u>Presentation Styles</u>

■ MRefBuilder Tools

Added the Windows Phone 8.0 framework to the framework definition file.

BuildAssembler Tools

- Reworked the BuildAssembler execution process to allow for parallel executon of component code. Components are still initialized and topics built sequentially for the time being but parallelism can be utilized now and is used in certain components when they are initialized.
- Reworked the BuildAssembler message logger to allow for parallel executon of component code without contention for the console when logging messages.
- Added a warning count to BuildAssembler which is displayed at the end of the build.

Build Components

🗹 Note

A number of these are breaking changes. However, most people will be unaffected unless they have created custom build components or syntax generators. These changes were necessary in order to make the API more accessible and to make it easier to derive build components from existing ones without having to reimplement a lot of the internal supporting classes.

- Deleted CodeReference, LiveExampleComponent, and WdxResolveConceptualLinksComponent as they appear to be for internal Microsoft use and do not appear to have much use for non-Microsoft users.
- Deleted the classes in the BuildComponents namespace from the BuildComponents project as they were unused. These were replaced by the like-named classes in the Microsoft.Ddue.Tools namespace.
- Deleted IntellisenseComponent2 and replaced IntellisenseComponent with the version from the Sandcastle Help File Builder. Added the IntelliSenseComponent configuration dialog which can be used by front end tools such as the Sandcastle Help File Builder to edit the component's configuration.
- Moved InheritDocumentationComponent into the main BuildComponents project and deleted the CopyComponents assembly as it was the only one in it.
- Moved the syntax writer classes into the SyntaxComponents assembly with their related classes.
- Moved all of the snippet related classes into their own namespace.
- Moved all of the target related classes into their own namespace.
- The reference link target classes were moved into their own assembly (*BuildComponents.Targets.dll*). This assembly is versioned independently of the other Sandcastle assemblies so that reference

target caches do not have to be rebuilt with each new release of the Sandcastle tools.

- Moved all of the command related classes into their own namespace.
- Made all of the supporting classes public.
- Made all reference link target classes serializable.
- Cleaned up and refactored the code where necessary in almost all of build components.
- Added a new CopyCommand abstract base class in BuildAssembler to make creating new copy commands possible.
- Added a new IndexedCache abstract base class for use by the CopyFromIndexComponent and its subcomponents for the index data. This also allows derived components to implement alternate caching mechanisms.
- Made TargetDictionary an abstract base class and added InMemoryTargetDictionary as the basic implementation. This allows alternate implementations that use a different storage mechanism for the target data.
- Reimplemented how reference link targets are loaded and stored. This allows for sharing target data independent of link type across multiple instances of ResolveReferenceLinksComponent2 reducing memory requirements and initialization time when building multiple output formats. The new implementation also allows caching of the information.
- Reworked the target dictionaries to allow for loading of target information in parallel to speed initialization up a bit.
- Renamed the IndexedDocumentCache to InMemoryIndexedCache and reworked it to support filtering by namespace to reduce the amount of reflection data that it needs to load and also added support for loading the index files in parallel.
- In CopyFromIndexComponent, added a virtual method to create the index caches, added code to dispose of them when done, and exposed the context via a protected property.

- Added a **Dispose** method to **CloneComponent** to properly dispose of all components in each branch.
- Simplified the FileCreatedEventArgs and updated the HxFGeneraterComponent to make it responsible for figuring out where to put its files.
- Added support for MSDN content ID caching to the ResolveReferenceLinksComponent2. The implementation allows derived classes to utilize backing stores other than the serialized binary file used by default as long as the cache type implements IDictionary<string, string>. It also allows sharing the cache across multiple instances of the component.
- Regenerated the MTPS Content Service classes. A decision was made to stay with the web service as the authentication issues have well-known solutions (see *BuildAssembler.exe.config*). Switching to a service type may introduce other authentication issues and since the web service works well enough, it was left as-is.
- Added support for defining which namespace targets to load in ResolveReferenceLinksComponent2. This prevents loading unnecessary namespaces saving some memory and initialization time.
- ReferenceLinkComponent2 and CopyFromIndexComponent now report cache usage at the end of the build with diagnostic messages. These can be used to help tune the cache sizes if necessary especially in derived caching components.
- Added support for a renderReferenceLinks attribute to the SyntaxComponent configuration's syntax element. By default the syntax component will not render links to types in the syntax section (this is not supported anymore by MS Help Viewer 2.0). By adding this attribute to the configuration and setting it to true, you can turn them back on again if wanted.
- Added a condition to the SyntaxComponent's Apply method to skip group, project, and namespace pages in which a syntax section is of no use. This allows removal of the IfThenComponent wrapper around it in the configuration files which wasn't using an up-to-date condition anyway.

- Updated the Visual Basic and Visual Basic Usage syntax generators to omit the line continuation character by default. It can be enabled by adding the includeLineContinuation attribute to the generator element and setting it to true.
- Removed the use of RegexOptions.Compiled in ExampleComponent which greatly improves its overall performance. The significant time and increased memory penalties outweighed any performance benefits over the course of the build.
- Updated DisplayComponent to output the content as a diagnostic message using the WriteMessage method rather than just dumping it to the console.
- Updated ValidateComponent to include the key in its warning messages.
- Moved the MSHelpAttrComponent and ShowMissingComponent from the Sandcastle Help File Builder into the Sandcastle BuildComponents project.
- Fixed up various issues in the BuildAssembler projects found by FxCop (too many to list).

Presentation Styles

- Changed the title element in the presentation styles so that it shows the qualified member name. This is required to provide unique titles for index entries. This does have the side-effect of causing member titles in the MS Help Viewer TOC to be qualified with the class name as Help Viewer no longer supports a TOCTitle metadata attribute like Help 2 did. Help for base framework classes exhibits the same behavior so this appears to be by design.
- Fixed the VS2010 style so that links work properly in Help 2.
- Fixed *VS2010**Branding**ps-body*.*xslt* so that it doesn't add a "./" relative URL prefix to non-relative image URLs.
- Commented out the templates in *VS2010\Branding\ps-body.xslt* that were stripping style attributes. These appear to be for use with non-self-branded content which doesn't apply anymore. Allowing the style attributes through fixes the problem of the "nobullet" list style not working.
- Updated all presentation styles so that fields, properties, events, and methods include a return type link even if no returns or value XML comments element is present. This makes it consistent with MSDN and allows for the removal of the type links within the Syntax section itself as the type links for the return value and parameter values are available elsewhere on the page.
- Deleted the unused reference and schema build configuration and XSL transformation files from all presentation styles.
- Removed the duplicate items from *shared_content_mshc.xml* leaving only the overridden path elements. The MS Help Viewer configuration files where updated to include the *shared_content.xml* file ahead of it.
- Updated the example and presentation style configuration files to reference InheritDocumentationComponent in the BuildComponents assembly.

- Added the ShowMissingComponent resource items to the *reference_content.xml* file in each presentation style.
- Added support for the markup element in main_sandcastle.xsl in each presentation style. This allows a block of HTML or other elements such as include to pass through as-is to the topic. This allows build components to insert content before the TransformComponent runs and prevents it being removed as unrecognized content by the XSL transformations. The conceptual transformations already allow this so this makes the conceptual and reference builds consistent in supporting a markup element.
■ See Also

Other Resources

Version History

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



Sandcastle Tools
 Version 2.7.2.0, December 2012
 See Also Send Feedback

[This is preliminary documentation and is subject to change.]

Version 2.7.2.0 was released on December 16th, 2012.

- <u>MRefBuilder Tool</u>
- BuildAssembler Tool and Components
- <u>Production Transforms</u>
- <u>Presentation Styles</u>
- <u>Extras</u>

■ MRefBuilder Tool

- Fixed MRefBuilder so that it does not exclude System.Xml and System.Data from the framework assembly cache.
- Updated the *Frameworks.xml* file to reference the December 2011 Silverlight Toolkit for Silverlight 5 rather than the older April 2010 version for Silverlight 4.
- Merged changes from ComponentOne that prevent additional cases that cause a crash when documenting obfuscated assemblies.

BuildAssembler Tool and Components

- Added support for topic key and message parameters in the message logging methods in BuildAssembler.
- Updated all build components to include the topic key in warning and error messages when available.
- Updated TransformComponent to raise a ComponentEvent with a TransformedTopicEventArgs parameter that contains the topic key and the transformed topic document. This allows components that ran earlier in the stack to perform tasks after the topic has been converted to HTML. This eliminates the need for a posttransform component as the component that ran earlier can handle the post-transform processing itself.

Production Transforms

- Updated *CreatePrototypeToc.xsl* with a rootNamespaceContainer parameter that can be set to true to have it generate a root namespace container node like the transformation for the other presentation styles.
- Updated ApplyVSDocModel.xsl and ApplyPrototypeDocModel.xsl to include assembly version number information in the API member data.

Presentation Styles

- Fixed up the casing on the *scripts*\ and *styles*\ folder references so that they are all consistent and will not cause casing issues on UNIX based web servers.
- Fixed VS2005 *CommonUtilities.js* so that the <see langword="XXX" /> elements work properly. The prior Help Viewer 2.0 fix broke their handling in the other help formats.
- Moved Shared\Content\syntax_content.xml and the files in Shared\Transforms\ into each of the presentation styles so that they are all self-contained. This will make cloning a presentation style to customize it easier. There was already a significant amount of duplication anyway so merging the few files that were shared does not make much difference in terms of maintenance.
- Removed the unused files in *Shared**Scripts*.
- Updated all presentation styles to include assembly version number information in the API topics.
- Updated all presentation styles to handle the title attribute on code elements.
- Added a condition to hide the language selector in the Prototype presentation style if there is only one language.
- Fixed handling of the preliminary XML comment element in all presentation styles so that it renders the content if specified.
- Added a transformation template in all presentation styles to get the code language so that it is consistent in all places that need it for the language filter.
- Added logo support to all presentation style XSL transformations. The VS2010 style now supports all of the logo placement options to match the other styles.
- Updated the XSL transformations so that they do not output an empty Abstract Help 2 metadata element for certain cases such as when it only contains a non-breaking space character.

- Merged changes from ComponentOne into the VS2005 style to prevent the unnecessary borders on the page headers in MS Help Viewer 1.0.
- Added support for the event XML comments element in all presentation styles. This was a custom element implemented by NDoc that never made it into Sandcastle. It renders a section called **Events** containing a table listing events that can be raised by the method and a description for how the event can be raised.

∃ Extras

- Added MAML snippet definitions for the common block and inline elements and a related page to the guided installer to copy them to the local snippets cache for the various Visual Studio versions.
- Updated the Sandcastle MAML guide to remove references to the Sandcastle Help File Builder components and features that have been moved into the Sandcastle build components and presentation style XSL transformations.
- Started to document the Sandcastle tools by adding this help file. The content and images in the general information and architecture topics was originally created by Dave Sexton as part of the wiki content on the Sandcastle Styles project site on CodePlex.

As of this release, it is still a work in progress. More information on the various tools and components will be added as time permits.

• Added the **Sandcastle XML Comments Guide** help file to provide a comprehensive set of documentation on the XML comments elements and how they are used with Sandcastle.

■ See Also

Other Resources

Version History

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



[This is preliminary documentation and is subject to change.]

Version 2.7.1.0 was released on October 2nd, 2012. This was the first official release after merging the Sandcastle tools with the Sandcastle Help File Builder project.

- <u>General Changes</u>
- <u>MRefBuilder Tool</u>
- XslTransform Tool
- Build Components
- <u>Presentation Styles</u>

■ General Changes

• Added /arg:project=Project to the *Generic.targets* file in the Reflection target to include a root namespace container in the manifest. This works around an issue with the web output from the example GUI failing if there is more than one namespace in the documented assemblies.

∠ Note

If you want to build website output, using the Sandcastle Help File Builder is recommended as it has better support for website output than the example GUI.

- Merged changes from Dany R. into the HTML to MAML Converter to support image alignment, topic ID and revision number metadata, and optional movement of leading topic text into the introduction element.
- Thanks to Alex Sherman and Sergey Zhurikhin for the Russian translations of the Sandcastle resource item files that have been added to the language pack.

■ MRefBuilder Tool

- Added code to Duplicator.VisitTypeReference() so that it doesn't get stuck in an endless loop when a type contains a nested type that itself implements a nested type from within the containing type (still with me?).
- Added a check condition in RecordExtensionMethods() to prevent a crash in an odd case where a potential extension method has no parameters.
- Added support for defining assemblies to ignore if unresolved. This is useful for ignoring assemblies that cannot be resolved for which you do not have a copy such as the Crystal Reports
 BusinessObjects.Licensing.KeycodeDecoder assembly. See the *MRefBuilder.config* file for an example definition (the ignorelfUnResolved child element of the resolver element).
- Added a new *Frameworks.xml* file for MRefBuilder to use in loading framework assembly information. This new style of framework definition is enabled by using the framework and version attributes on the platform element in the *MRefBuilder.config* file. This ensures that the reflection information is accurate for the selected framework type and version.

The framework and version attribute values correspond to one of the framework sets defined in the *Frameworks.xml* file. Using this method allows for automatic referencing of core framework assemblies as well as automatic binding redirection from older versions to the defined version for all assemblies in the framework's assembly set (i.e. automatically redirecting the RIA SDK dependencies from the Silverlight 4 versions to the Silverlight 5 versions when producing information for a Silverlight 5 project that uses the RIA SDK). In other words, you no longer need to specify the dependencies or define binding redirections for the standard framework assemblies.

Solution X Statement Tool

Added an UnauthorizedAccessException check to ignore temporary files that may be locked when attempting to delete them (i.e. virus scanners have them open).

Build Components

- Made a couple of fixes to BrandingComponent that allows the content to show up in Help Viewer 2.0 that comes with Visual Studio 2012.
- In the MSHCComponent:
 - Removed the ContentType metadata as it is output by the XSL transformations.
 - Changed SelfBranded to Microsoft.Help.SelfBranded for Help Viewer 2.0 support. Help Viewer 1.0 does not care which name you use for self-branded content but Help Viewer 2.0 requires the qualified name or it attempts to brand the content which causes some odd display issues.
 - Removed the VS2005 header table fix code as it was not needed anymore. It is already handled by the script used in the presentation style (see below).
- Added an extra slash to the ms-xhelp links generated by ResolveConceptualLinksComponent and ResolveReferenceLinksComponent2 so that they work properly with Help Viewer 2.0. Without it, Help Viewer 2.0 renders invalid links for some reason.

Presentation Styles

- Replaced conceptualLink elements with anchor links to the corresponding MSDN topics in the VS2010 style's resource item files.
- Removed the style attribute from VS2010\Branding\pscodesnippet.xslt as it was causing issues with FireFox and Chrome when viewing website output. The attribute appears to be unnecessary and its removal does not appear to affect the results.
- Added a case for using the currently defined target attribute value on an external link in VS2010\Branding\ps-instrumentedLink.xslt so that it honors the user-defined link target rather than always forcing it to _blank.
- Fixed a bug in *Presentation\VS2010\Branding\psinstrumentedLink.xslt* that caused local links to be rendered with an incorrect URL in Help Viewer 2.0.
- Removed the background attribute from the nsrBottom table element in the VS2005 style as it was breaking the language filter links at the top of the page in Help Viewer 2.0. The script sets the background so it was not needed anyway.
- Fixed various other issues in the VS2005 and VS2010 presentation styles to get them to work properly with Help Viewer 2.0.
- Fixed the VS2005 and VS2010 presentation styles so that F1 keywords work properly when looking up API members via F1 in Visual Studio with MS Help Viewer.
- Merged the collapsible section changes for MAML content from Microsoft into the Prototype style (Sandcastle work item #2269).
- The *dev10.mshc* file from Visual Studio 2010 is no longer required. Due to differences in the way Help Viewer 2.0 works and the fact that the file is not part of Visual Studio 2012, a temporary file of the same name has been included with a set of transformations necessary to get the content built with Sandcastle. A future release will probably get rid of it altogether in favor of integrating its various parts directly into

the presentation style.

■ See Also

Other Resources

Version History

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



[This is preliminary documentation and is subject to change.]

Version 2.7.0.0 was released on April 15th, 2012. This release was issued before officially being merged with the Sandcastle Help File Builder project. It fixed almost all of the known tool bugs and merged all changes from the Sandcastle Styles patch into the Sandcastle XSL transformations.

- General Changes
- <u>Schemas</u>
- <u>Command Line Library</u>
- <u>MRefBuilder Tool</u>
- <u>BuildAssember Tools</u>
- Build Components
- DBCSFix Tool
- <u>ChmBuilder Tools</u>
- <u>Presentation Styles</u>

□ General Changes

- Rearranged the source project folders to match the installed folder layout. This makes testing a development build much easier as you can point DXROOT at the .*Sandcastle\Main* folder and have it work like a release build.
- Created separate solutions for each tool so that they can be loaded, built, and tested individually. Reflector was used to get the source code for the tools that were missing source code and projects were added for them too.
- All tools are now built using .NET 4.0. This will allow taking advantage of newer features in the runtime added since .NET 2.0. This does not change the ability to run the tools against assemblies built with prior framework versions. That is still supported.
- Removed all *GlobalSuppressions.cs* files from all projects. Many of us do not have a version of Visual Studio that supports code analysis within the IDE. Instead, individual FxCop projects have been created for each project. Some clean up has been performed based on the initial scans. Warnings still exist in several projects and have been left for review until later.
- Signed all assemblies with a new key file and standardized the assembly attributes.
- Added *Data\BuildReflectionData.bat* and *Data\BuildReflectionData.proj* to properly rebuild the reflection data files.
- Updated the XSL stylesheet version number to 2.0 in all XSL transformations. A bug in .NET 4.0 prevents the <xsl:sort> function from working when using XSL version 1.1.
- Merged the Sandcastle Guided Installer, the Language Pack, Sandcastle MAML Guide, Web Code Providers, and the HTML to MAML Converter projects from Sandcastle Styles into the Sandcastle project. These are located under the .*Extras* folder.
- The Sandcastle Guided Installer has been rewritten as a WPF

application. It uses flow documents rather than HTML resources for the pages which are much easier to extend and work with in the code.

• Created a new setup project to install Sandcastle. This version includes the content from the .*Extras* folder as additional installable options. This installer also gives Sandcastle a presence on the Program Files menu with links to the included help files, the project web site, and the example GUI. The setup project is located in the .*Extras folder* too as it isn't part of the core Sandcastle code and uses the WiX tools.

∃ Schemas

- Added a new optional attribute to the reflection data schema for parameter elements. This is needed to properly document optional parameters that use OptionalAttribute and have no default specified using assignment.
- Updated the MAML schemas to include all of the new elements and attributes added by the Sandcastle Styles patch and adjusted a few elements to reflect how Sandcastle uses them. This allows for proper validation of MAML topics in Visual Studio.

Command Line Library

• Reworked the command line option classes to fix various FxCop warnings and to correctly implement support for required options.

■ MRefBuilder Tool

- Moved the content of the *CCI*\ and *Reflection*\ folders into the MRefBuilder project and made them members of the project so that they are compiled directly into *MRefBuilder.exe* to match prior releases of Sandcastle.
- Fixed *ExtensionMethodAddIn.cs* so that it doesn't add extension methods to enumerations and static classes (Sandcastle work item #8852).
- Fixed *ExtensionMethodAddIn.cs* so that it ignores unexposed namespaces and types. This prevents it from scanning unnecessary namespaces and types and stops a crash caused by it scanning compiler generated types created by the code contracts post-processing tool (Sandcastle work item #11066).
- Fixed

Method.ImplicitlyImplementedInterfaceMethods in *Nodes.cs* so that it recognizes interface member matches when the return type is generic (Sandcastle work item #22970).

- Fixed GetTemplateMember() and ParametersMatch() to properly check for template parameters when there are method overloads in which one uses a generic type and the other does not (i.e. Contains(T) and Contains(Guid)). This was Sandcastle work item #1908 and most likely fixed work item #7803 too as it looks similar and the test case works as expected now.
- Fixed TypeNode.Attributes so that it will not get stuck in an endless loop if a type's attribute references the type being parsed (Sandcastle work item #2253).
- Fixed TypeNode.NestedTypes so that it will not get stuck in an endless loop when a type contains a nested type that itself implements a nested type from within the containing type.
- Fixed ISExposedMember() and ISExposedType() in *ApiFilter.cs* so that they ignore unrecognized type and member visibility values (Sandcastle work item #2967 and #2969).

- Fixed IsExposedMember() so that it compares generic members using the Name<T> and Name{T} syntax so that it gets a match either way (Sandcastle work item #5593).
- Fixed ISExposedExpression() in *ApiFilter.cs* so that it doesn't exclude a type in an attribute expression as long as the hidden type has exposed members thus exposing the type.
- Fixed IsExposedNamespace(), IsExposedType(), and IsExposedMember() in *ApFilter.cs* so that they exclude members with names containing characters that are not valid in XML (i.e. obfuscated member names).
- Fixed *MemberDictionary.Contains()* so that when checking for matching members it compares generic template parameters by name to match members with generic parameters correctly. This fixes an issue where it treats overridden members as overloads when they contain generic template parameters. This was Sandcastle work item #4553 and most likely fixed work item #11303 too as it looks similar and the test case works as expected now.
- Fixed OrcasNamer.WriteTemplate() so that it uses the correct template parameter names which do not always match the base class's template parameter names (i.e. Collection<TControl> vs. Collection<T>). Sandcastle work item #5594.
- I am not sure if one of the other changes fixed it or if it was fixed already in the source code, but Sandcastle work item #2903 (Spurious references to parameterized class with inner class) is fixed as the test cases in it work as expected.
- Added a check for exposed members in unexposed types in **ISExposedType()** in *ApiFilter.cs*. Such cases effectively expose the type and it should be included whenever this check occurs for it. Without the check, it was incorrectly excluding types in several locations.
- Added CompilerGeneratedAttribute check to ISExposedType() in *ExternalDocumentedFilter.cs* as public members can sometimes be compiler generated (i.e. generated types for public fields that use the fixed keyword). This was a new issue I

found while testing the fixed keyword syntax issue.

- Added FixedBufferAttribute to the default *MRefBuilder.config* API filter as an exposed attribute so that it gets passed through. This is needed in order to properly document fixed members in the syntax section.
- Added OptionalAttribute to the default *MRefBuilder.config* API filter as an exposed attribute so that it gets passed through. This is needed in order to properly document optional parameters.
- Added code to write out the new optional attribute for optional parameters. This is needed to properly document optional parameters that use OptionalAttribute and have no default specified using assignment. The syntax components and XSL transformations have been updated to make use of it to document optional parameters and their values (Sandcastle work item #9627).
- Merged my changes into the AssemblyResolver class to provide support for assembly binding redirection. Added example configuration info for it to *MRefBuilder.config* (Sandcastle work item #1014).

BuildAssember Tools

- Added the proxy configuration settings that work around the HTTP 407 and HTTP 417 errors in comments to the *BuildAssembler.exe.config* file.
- Added support for specifying a verbosity level on the configuration element. This can significantly reduce the build assembler output and makes it easier to see warnings as it is much less chatty. The default verbosity is Info so that its default behavior matches prior releases.

Build Components

🛦 Important

Custom build components that override Dispose() will need to be updated to override protected void Dispose(bool disposing) rather than public void Dispose() as the Sandcastle source code release correctly implements the IDisposable pattern.

- Fixed ResolveReferenceLinksComponent2 so that it does not write out unnecessary parentheses on unresolved property elements (Sandcastle work item #1987).
- Fixed ResolveReferenceLinksComponent2 so that references to enumerated field types are redirected to the containing enumerated type so as to produce a valid link target (Sandcastle work item #2221).
- Fixed the C#, C++, VB.NET, F#, and J# syntax generator components so that they output numeric attribute values. In the event that an unrecognized attribute parameter type is encountered, the syntax generators write out the value as-is. Array parameters are also supported now but the reflection data does not contain the values. As such, it writes out the array type and a generic "{ ... }" placeholder where the values would go. The information is probably there but it would probably require changes to the CCI code to include it (Sandcastle work item #6779)
- Fixed the

XamlUsageSyntaxGenerator.WritePropertySyntax() so that it generates syntax for properties with abstract return types as long as there is a type converter for it (i.e. Brush) Sandcastle work item #5466.

 Added support for a duplicateWarnings attribute that can be added to the data elements of the index element on the CopyFromIndexComponent. When set to false, it suppresses duplicate index key warnings. This is useful for comments files where there can be duplicate keys but it isn't an issue. The default is true if not specified to maintain the behavior from past releases and report the warnings (Sancastle work item #11844).

- Added support for optional arguments to the C#, C++, VB.NET, F#, and J# syntax generator components. I did not enclose them in brackets as it looked rather odd when attributes were present and the assignment is a dead giveaway that it is an optional parameter. As such, it uses the standard syntax but I did update the XSL transformations to include "(Optional") after the parameter name in the Parameters section of the topic (Sancastle work item #9627).
- Added support for attributes on property getter/setter methods to the C#, C++, VB.NET, F#, and J# syntax generator components (Sandcastle work item #973).
- Made the unsafe code checks consistent across all syntax generators and added a check for the FixedBufferAttribute to the apiIsUnsafeExpression XPath expression.
- Added fixed keyword support to the C# and C++ syntax generators (Sandcastle work item #10107).
- Added interior_ptr<> support to the C++ syntax generator. Added a fixup regular expression to the Sandcastle help file builder too so that the comments are included in the member page (Sandcastle work item #2299).
- Merged my changes and bug fixes into the MSHCComponent. The changes include:
 - Support for a **sortOrder** option on the TOC elements to allow the sort order of the elements to be defined to set the proper placement of the TOC entries when parented to an entry outside of the help file and to parent the API content within a conceptual content folder.
 - Fixed a bug that caused a "duplicate key" error under certain conditions.
 - Fixed an incorrect XPath expression that was letting duplicate metadata through.

- Added my JavaScriptDeclarationSyntaxGenerator to the SyntaxComponents assembly. Modified *FixScriptSharp.xsl* to include a scriptSharp element to indicate to the syntax generator that it should apply the casing rules to member names. This allows the added syntax generator to work with normal JavaScript and Script# projects alike (Sandcastle work item #1999).
- Merged my changes and bug fixes into the ResolveConceptualLinksComponent (Sandcastle work item #2193):
 - Broken links use the None style rather than the Index style so that it is apparent that they do not work.
 - The inner text from the conceptual link is used if specified.
 - On broken links, when the showBrokenLinkText option is true and there is no inner text, the target value is displayed.
 - Conceptual link targets can include an optional anchor name from within the target such as "#Name".
 - Unnecessary whitespace is removed from the link text.
 - If the companion file contains a <linkText> element and no inner text is specified, its value will be used for the link text rather than the title. This allows for a shorter title or description to use as the default link text.

DBCSFix Tool

- Updated the file masks to use *.*htm*? to find both .*htm* and .*html* files (Sandcastle work item #1072).
- Fixed the incorrect check for LCID value (Sandcastle work item #1072).
- Fixed SubstituteAsciiEquivalents() so that it actually performs the substitutions for encodings other than Windows-1252 (Sandcastle work item #1072).
- Switched to using EnumerateFiles() which is more efficient for large projects.

ChmBuilder Tools

Switched to using EnumerateDirectories() and EnumerateFiles() which is more efficient for large projects.

Presentation Styles

This release added the new VS2010 presentation style developed by Don Fehr. Consider this a beta release as we work out the bugs. The following fixes and feature enhancements from the Sandcastle Styles patch and some unreported bugs from the discussion pages were applied to the presentation style files:

- Sandcastle work item #6858: The omitVersionInformation parameter is missing from VS2005\transforms\main_sandcastle.xsl
- Sandcastle work item #1943: ResolveArtLinksComponent is wrong in the configs
- Sandcastle work item #6785: Enumerated type members no longer contain a description in VS2005 style
- Sandcastle work item #2268: The sub-section toggles do not work in Hana and VS2005 because they are not given a unique ID
- Sandcastle work item #2603: "this" argument modifier not shown for extension methods
- Sandcastle work item #1965: XAML code samples cannot be hidden (VS2005 style)
- Sandcastle work item #4476: Cannot link to overloads page
- Sandcastle work item #938: Hana and VS2005 styles generate unnecessary Overloads pages and TOC entries
- Sandcastle Styles work item #6572: Unresolved types don't show up in overloaded method signatures in TOC entries and topic titles
- Sandcastle work item #2255: Remaining presentation Style Issues
- Sandcastle work item #1249: Conceptual: codeEntityReference is sensitive to whitespace
- Sandcastle work item #2034: Conceptual: XLinks in relatedTopics section requires explicit text
- Sandcastle work item #6788: Support for definition list type is missing.

- Sandcastle work item #2258: Conceptual: ddue:copyright processing is broken
- Sandcastle work item #2264: ddue:list should support the "nobullet" style
- Sandcastle work item #2949: Sections containing nothing but an image or a list of link elements are not rendered
- Sandcastle work item #3652: MAML XSLT Generates Self-Closing Tag for Bookmarks
- Sandcastle work item #8879: Some pre elements are missing xml:space="preserve"
- Sandcastle Styles work item #11465: FireFox website output issues
- Sandcastle work item #2421 and #2435: Case of Presentation.css is not consistent in transformations. This causes case-sensitive web servers to miss the stylesheet.
- Sandcastle work item #2416: vs2005: summary from outer class ends up on inner class member page
- Sandcastle Styles work item #5362: Help 1 Sticky Language Filter
- Sandcastle work item #2274: Conceptual: Hana style, External link in a token not rendered in relatedTopics
- Sandcastle work item #2951: JavaScript syntax section is not shown in the Hana style
- Sandcastle work item #2260: Anchor tag styles need to be fixed
- Sandcastle work item #2261: Conceptual: All three styles, normalize the space on the abstract
- Sandcastle work item #2265: ddue:table handling needs fixing
- Sandcastle work item #2266: Rendering of ddue:definitionTable isn't consistent in presentation styles
- Sandcastle work item #2950: autoOutline displays "Related Topics" instead of "See Also"
- Sandcastle work item #2275: Conceptual: ddue:math should apply

templates to its inner text

- Sandcastle work item #2060: Conceptual: Formatting is not applied to some in-line MAML elements
- Sandcastle work item #2273: Support linkAlternateText and linkTarget in externalLink
- Sandcastle work item #1854: Conceptual: Don't Display Summary in Topic
- Sandcastle work item #2282: Prototype syntax sections have unnecessary leading whitespace
- Sandcastle work item #939: Version Builder Issues
- Sandcastle Styles work item #226453: Version Builder is broken in the June 2010 release
- Sandcastle work item #2257: Conceptual: Hana does not wrap code blocks in <div class="code">
- Sandcastle work item #998: Hana and Prototype styles need to add CHARSET attribute to topics
- Sandcastle work item #2259: Conceptual: Hana and Prototype are missing some namespace declarations
- Sandcastle work item 2267: ddue:schemaHierarchy and indent templates are missing from Prototype
- Sandcastle work item 2272: The Prototype See Also section doesn't have a #seeAlsoSection anchor
- Sandcastle work item #1613: seealso tags within an overloads tag not listed
- Sandcastle work item #6787: Add missing resource items to Hana and Prototype styles
- Sandcastle work item #1978: APIName Attributes Incorrect (Hana and Prototype)
- Sandcastle work item #974: Support External Hyperlink Target with Appropriate Default

- Sandcastle work item #6803: Support term element in bullet and number list types
- Sandcastle work item #6789: Support starting number on numbered lists
- Sandcastle work item #2948: Add address attribute support to several elements
- Sandcastle work item #2270: Conceptual: Support NamedUrlIndex keywords in metadata
- Sandcastle work item #2174: Conceptual: Pass-through HTML (using a markup element), supersedes Sandcastle work item #933
- Sandcastle work item #6786: Enumeration topics should show the value of each enumeration member
- Sandcastle work item #1835: Conceptual: Glossary Improvements
- Sandcastle work item #1795: Better implementation of the <note> tag
- Sandcastle work item #2256: Better implementation of the ddue:alert element
- Sandcastle work item #2189: mediaLink and mediaLinkInline should be handled as separate templates
- Sandcastle work item #2142: Conceptual: autoOutline Changes
- Sandcastle work item #2082: Auto-generated Bibliography (No dependency on build components)
- Sandcastle work item #6790: Add code contract support for VS2005 style
- Sandcastle work item #2271: List formatting doesn't look good (all three styles)
- Sandcastle work item #2947: Stylesheet updates needed to correct spacing and alignment
- Sandcastle work item #1950: Conceptual: Glossary term requires termClass attribute
- Sandcastle work item #2021: Conceptual: codeSection template

ignores valid code languages

- Ureported feature request: Replace conceptualLink elements in the reference content with anchor links to the actual MSDN pages
- Unreported feature request: Support conceptualLink elements in code XML comments to allow linking to conceptual topics from API member docs
- Unreported bug: VS2005 CommonUtilities.js fails under MS Help Viewer 2
■ See Also

Other Resources

Version History

Send comments on this topic to Eric@EWoodruff.us

[v2.7.3.0] Sandcastle Tools



Glossary Send Feedback

[This is preliminary documentation and is subject to change.]

$\underline{A} \mid \underline{B} \mid \underline{C} \mid \underline{D} \mid \underline{F} \mid \underline{G} \mid \underline{H} \mid \underline{I} \mid \underline{J} \mid \underline{K} \mid \underline{L} \mid \underline{M} \mid \underline{N} \mid \underline{O} \mid \underline{P} \mid \underline{Q} \mid \underline{R} \mid \underline{S} \mid \underline{T} \mid \underline{U} \mid \underline{V} \mid \underline{W} \mid \underline{X} \mid \underline{Y} \mid \underline{Z}$

A

additional content

An extra file included in the help file used to define extra content or used as a supporting file. This can be an HTML topic, an image, script, stylesheet, etc. In earlier releases of the help file builder, this was the way non-API content was included in the help file. The preferred method of including such content now is to use conceptual content. Although still supported, the additional content features of the help file builder are now considered deprecated.

See Also:

conceptual content

B

BuildAssembler.exe

This is a tool supplied with Sandcastle that is used to build the help topics for conceptual and reference builds. It is passed a configuration file containing a list of build components to use in transforming the topics into HTML and a manifest file that lists the topics to build.

See Also:

Sandcastle

C Code Block Component A custom build component that can be used to colorize code, add line numbering and/or collapsible section, and import code from working source code files. It can be used in both conceptual and reference builds. The component is integrated with the

Sandcastle Help File Builder and is also available as a standalone component from the same project site for use in other tools or your own build scripts. See Also: Sandcastle, Sandcastle Help File Builder

code snippets file

A file used to store commonly used code snippets. The snippets can be inserted into a conceptual topic using the CodeEntityReference element.

conceptual build

A build that uses Sandcastle to produce help file topics using information extracted from conceptual content topics and their related files.

See Also: reference build, Sandcastle

conceptual content

A topic file that contains conceptual content. These can be used to add usage notes, walkthroughs, tutorials, etc. to a help file. Conceptual topics use MAML rather than HTML to define their content.

See Also: additional content, MAML

D

dependency

See

<u>reference</u>.

documentation source

An individual assembly (executable or DLL), an individual XML comments file, a Visual Studio project file (C#, VB.NET, or J#), a Visual Studio

solution file containing one or more of the noted project types, or a wildcard item that results in a list of one or more of the previously mentioned file types. These are used to produce API documentation in a help project. At least one assembly must be specified as a documentation source in order to produce API help content. Documentation sources are managed using the **Documentation Sources** project node in the **Project Explorer** window.

G

Globally Unique Identifier, GUID

A unique value that is associated with each conceptual topic and image in a project. When inserting links to topics or images, the ID is used to refer to them. This allows you to alter the names or locations of the topic files without having to change the name or location in each topic that references them.

Μ

MAML

An acronym that stands for Microsoft Assistance Markup Language. Conceptual content topics are composed of MAML elements.

See Also:

conceptual content

R

reference

Also referred to as a *dependency*. This is an assembly that is referenced by one of the documented assemblies for base class information but is itself not documented as part of the help project. They are managed via the **References**

project node in the **Project Explorer** window. References can be GAC entries, assembly files, COM objects, or Visual Studio project files.

reference build

A build that uses Sandcastle to produce help file topics using information extracted from managed assemblies and their associated XML comments files.

See Also:

conceptual build, Sandcastle

S

Sandcastle

Sandcastle is a set of tools produced by Microsoft that can be used to build help files for .NET managed libraries, conceptual content, or a mix of both. Microsoft officially discontinued development in October 2012. The Sandcastle tools have been merged with the

Sandcastle Help File Builder project and are developed and supported there now as part of that project. See Also: Sandcastle Help File Builder

Sandcastle Help File Builder, SHFB

The <u>Sandcastle Help File Builder</u> is a standalone tool used to automate Sandcastle. It consists of a GUI front end that helps you manage and build help file projects. It uses a standard MSBuild format project file which can also be built from the command line using MSBuild or integrated into Visual Studio builds or other build scripts to produce a help file when your application projects are built. In addition, it provides a set of additional features beyond those supplied with Sandcastle that can improve your help file and make it easier to deploy. A Visual Studio integration package is also available for it that integrates the project management and build features into Visual Studio 2010 or later.

See Also: <u>Sandcastle</u>

Т

token, token file

A token is used as a replaceable tag within a topic and is represented using a **token** element. The inner text of the element is a token name. The tokens are defined in a separate token file. They are an easy way to represent common items that you use regularly such as a common phrase or external link. Send comments on this topic to Eric@EWoodruff.us [v2.7.3.0] Sandcastle Tools