

Welcome

Welcome to the Sandcastle XML Comments Guide. This is intended to be a reference that you can use to find out all that you need to know about XML comments as used with Sandcastle to produce help files containing API reference content. If you have any information, tips, updates, or corrections that you would like to see added to the guide, feel free to submit them to the author using the **Send Feedback** link at the top of the page or the e-mail link in the page footer.

▲ Conventions

The XML comment elements are separated into four categories:

- Section These elements are used to define the content for the different sections of the documentation of a type or member and are always top-level elements that appear at the root of the XML comments block. Examples are summary and remarks.
- Block These elements format text within the top-level section elements. They are typically used to add structure to the text inside those elements. Examples are code and list.
- Inline These elements are typically used inside the other section and block elements to provide formatting such as the c element or links to other API topics such as the see element.
- Miscellaneous These elements are typically top-level or mixed use elements that do not fit one of the other categories such as include and inheritdoc.

For each XML comments element, information will be given about its placement and usage along with one or more examples. In the syntax examples, optional attributes and parameters will be enclosed in square brackets ([]). If an attribute allows you to specify a value from a defined list of values, those values will be shown separated by a pipe character ([)). The **See Also** section will contain links to example API members so that you can see how a topic containing the element is rendered in an actual help file.

Certain elements or attributes they utilize are implemented by Sandcastle or are only usable with a particular third-party tool such as the Sandcastle Help File Builder. Such extensions to the standard XML comments elements and attributes will be noted in the affected topics.

Since the syntax of the XML comments themselves are language neutral with the exception of the comment characters that precede them, all example code is shown only in C# to reduce unnecessary clutter. Substitute your language's XML comments characters (i.e. triple apostrophe (''') for VB) and the XML comments will work the same way.

▲ See Also

Other Resources

Walkthrough: Enabling and Using XML Comments
Usage Notes
Making a Donation
Version History

[v2014.5.31.0] Sandcastle XML Comments Guide



License Agreement

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, royalty-free 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.

[v2014.5.31.0] Sandcastle XML Comments Guide



Walkthrough: Enabling and **Using XML Comments**

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

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.



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**.
- Expand the Configuration Properties category and then expand the C/C++ sub-category and select the Output Files option below it.



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.

- 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> element to each public type and to the public and protected members of those types. There are many other elements available that will help improve the quality of your documentation. See the other sections in this guide for more information

The XML comments elements and their format is consistent across all languages. The only variable is the comment characters that precede them. Here are some examples.

```
''' <summary>
''' This is an example of triple apostrophe XML (
''' delimiters supported by Visual Basic.
''' </summary>
```

```
C#, C++, and F#

/// <summary>
/// This is an example of triple slash XML commer
/// form of XML comments delimiter in C#, C++, ar
/// </summary>

NOTE: The multi-line XML comments delimiters show

/** <summary>
    * This is an example of using the multi-line XN
    * case, the common pattern " * " at the start
```

```
* is ignored by the compiler and will not appea
* </summary>
    */

/**
<summary>This example does not use a common lead;
the summary comments using the multi-line XML con

/** <summary>A single-line summary using the mult
•
```

■ Next Steps

Refer to the other sections in this guide for more information about the various XML comments elements and how to use them.

▲ See Also

Other Resources

Welcome

Usage Notes

Namespace Comments

Section Elements

Block Elements

Inline Elements

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



Usage Notes

The following are some general usage notes and things to keep in mind when authoring XML comments in your code.

 XML comments support all of the common HTML tags such as a (anchor), b (bold), img (image), p (paragraph), etc. As such, you can use HTML markup to add additional formatting to your XML comments if necessary.

Caution

Not all help output formats support HTML elements in XML comments or may use a different URL format from other help formats. For example, MS Help Viewer's URL format differs from the other help formats which may result in broken links if literal HTML anchor or image links are used. As such, try to use XML comments elements whenever possible. For example, to link to API topics, use see elements and for conceptual topics, use conceptualLink.

- All XML and HTML within XML comments must be well formed.
 That means that all elements must have a corresponding closing element or must be self-closing, special characters such as "<" and ">" must be encoded appropriately (i.e. < and >), entities not recognized by the XML parser must be encoded in their numeric form (i.e. rather than for a non-breaking space).
- For larger blocks of XML that you want to insert literally such as code examples, you can use the <! [CDATA[]]> option to place them in the element. See the code element for an example.
- When referencing a generic type or a member with generic parameters as the target of an element attribute such as the cref attribute on a see element, use curly braces rather than angle brackets on the generic type parameters. For example:

```
/// <summary>
/// This class is derived from <see cref="Lis'
/// </summary>
/// <typeparam name="T">The element type of tl
/// <seealso cref="CustomDictionary{TKey, TVa.
public class CustomList<T> : List<T>
{
}
```

 Namespace comments cannot be attached directly to a namespace code element. Instead, you must use an alternate means to specify namespace comments. See the Namespace Comments topic for more information.

▲ See Also

Other Resources

Walkthrough: Enabling and Using XML Comments

Member ID Strings

Namespace Comments

Section Elements

Block Elements

Inline Elements

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



Member ID Strings

All cref attributes on documentation elements such as see or exception generate an ID string in the XML comments file. Normally, the ID string can be properly generated by the compiler based on a partial or fully qualified type or member name. However, there are occasions where this will not work properly. A common example is in C++ code where the compiler is not always able to generate IDs for forward referenced members. In such cases, an ID string can be used as a reference explicitly instead of letting the compiler figure it out.

Note

Author Credit: Most of the information in this topic was originally written by Microsoft as part of the .NET Documentation Guidelines document.

The format of the ID string is shown below.

Conv

type:fullname[(arglist)]`genericcount

type is one of the following. The final two are specific to the Sandcastle Help File Builder and Tools and are never generated by the compiler.

Type	Description
!	Used by the compiler to indicate an error such as being unable to resolve the member ID
N	Namespace
Т	Type (class, interface, structure, enumeration, etc.)
F	Field
E	Event

P Property
 M Method
 R Sandcastle generated. Represents the root namespace page. There will always be a single entry named R:Project_[HtmlHelpName] where "[HtmlHelpName]" is the value of your project's HTML Help Name property with spaces replaced by underscores. This suffix is required to keep the root namespace container page ID unique across all help files so that there are no duplicate IDs when generating MS Help Viewer output.
 O Sandcastle specific. Used to generate a link to the Overloads List page for an overloaded member.

fullname is the full name of the member from the root. Thus referencing StringBuilder would be System.Text.StringBuilder. The full name will include any member references as well so referencing the Append method of StringBuilder would be System.Text.StringBuilder.Append.

For properties that accept parameters and methods with parameters, the argument list is next. The argument list contains the parentheses as well. Each argument is specified as the full name of the associated type. For reference types, the type name is followed by an at (@) sign. For array types, the type name is followed by brackets ([]). Other symbols are possible but they are not supported by Visual Basic or C#.

For generic classes, the type is followed by a back tick (`) and the number of generic type parameters. For generic methods, there is one back tick (`) and a number for each generic parameter. Additional generic parameters are separated by a comma. Each number is the zero-based index within the parameter list of the parameter.

There are no spaces allowed in the ID string. If a name contains a dot (except to separate namespaces from types, types from other types, and types from members) then the pound (#) sign is used instead. In general, such a member is not possible. However, for referencing special methods like a constructor or explicitly implemented members, it is needed. An example of an explicitly implemented member ID is:

```
M:TestNamespace.TestClass.System#Collections#IEnu
```

The constructor for a type is always of the following form.

```
M:fullname.#ctor[(arglist)]
```

The destructor for a class is always of the following form.

```
M:fullname.Finalize
```

Operators are encoded as the formal name of the operator followed by their arguments. For conversion operators, the formal name is used (op_Explicit or op_Implicit) followed by the arguments, a tilde (~), and the return type. For example:

```
M:fullname.op_[binaryOpname](arg1, arg1)
M:fullname.op_[unaryOpname](arg)
M:fullname.op_Explicit(arg1, arg2)~returnType
M:fullname.op_Implicit(arg1, arg2)~returnType
```

Below are example IDs for each overloadable operator:

Operator	Usage	Member ID Example
+	x = y + z	M:TestDoc.Type1.op_Addition(TestDc
&	x = y & z	M:TestDoc.Type1.op_BitwiseAnd(Test
I	$x = y \mid z$	M:TestDoc.Type1.op_BitwiseOr(TestE
	X	M:TestDoc.Type1.op_Decrement(Test
/	x = y / z	M:TestDoc.Type1.op_Division(TestDo
==	if(x == y)	M:TestDoc.Type1.op_Equality(TestDo
٨	x = y ^ z	M:TestDoc.Type1.op_ExclusiveOr(Test

explicit	string x = (string)test1Object	M:TestDoc.Type1.op_Explicit(System
false	if(test1Object == false)	M:TestDoc.Type1.op_False(TestDoc.
>	if(x > y)	M:TestDoc.Type1.op_GreaterThan(Te
>=	$if(x \ge y)$	M:TestDoc.Type1.op_GreaterThanOr
implicit	string x = test1Object	M:TestDoc.Type1.op_Implicit(TestDoc
++	χ++	M:TestDoc.Type1.op_Increment(TestI
!=	if(x != y)	M:TestDoc.Type1.op_Inequality(TestE
<<	x = y << 2	M:TestDoc.Type1.op_LeftShift(TestDo
<	if(x < y)	M:TestDoc.Type1.op_LessThan(Test[
<=	$if(x \le y)$	M:TestDoc.Type1.op_LessThanOrEq
!	if(!x)	M:TestDoc.Type1.op_LogicalNot(Test
%	x = y % z	M:TestDoc.Type1.op_Modulus(TestDo
*	x = y * z	M:TestDoc.Type1.op_Multiply(TestDo
~	x = ~x	M:TestDoc.Type1.op_OnesCompleme
>>	x = y >> 2	M:TestDoc.Type1.op_RightShift(Test[
-	x = y >> 2 $x = y - z$	M:TestDoc.Type1.op_RightShift(Test[M:TestDoc.Type1.op_Subtraction(Tes
- true	•	
- true	x = y - z if(test1Object ==	M:TestDoc.Type1.op_Subtraction(Tes

+test1Object

▲ See Also

Other Resources

Walkthrough: Enabling and Using XML Comments

Usage Notes

[v2014.5.31.0] Sandcastle XML Comments Guide



Namespace Comments

XML comments cannot be applied directly to a namespace within your code. As such, you must take an alternate approach to specifying namespace comments.

- Your documentation build tool may provide support for defining project summary and namespace comments from within the development environment. For example, the Sandcastle Help File Builder provides a Summaries project property category that you can use to specify project summary and namespace summary comments. See your build tool's documentation for further information.
- You can manage the XML comments for the project summary and namespace summaries in an external XML comments file.
- Your documentation build tool may provide support for managing the namespace summary comments within your source code using a special hidden class called NamespaceDoc to which you attach the comments in each namespace.

▲ See Also

Other Resources

Walkthrough: Enabling and Using XML Comments

Usage Notes

Section Elements

Block Elements

Inline Elements

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



External XML Comments Files

Project summary and namespace comments can be maintained manually by creating a standalone XML comments file to contain the information.



Project summary comments will appear on the Namespace List topic in the compiled help file.

Start by creating an XML comments file like the following and give it a unique name that will not conflict with any other XML comments file in the related projects.

```
Sample Project/Namespace XML Comments File
<?xml version="1.0"?>
<doc>
  <assembly>
    <name>_NamespaceComments_</name>
 </assembly>
  <members>
    <member name="R:Project_CustomControlsHelp">
      <summary>
        Project summary comments go in here.
      </summary>
    </member>
    <member name="N:">
      <exclude />
      <summary>
        Exclude the global namespace from the hel
      </summary>
    </member>
    <member name="N:CustomControls.Internal">
```

A Important

The project summary comments will be placed in the member with the ID starting with "R:Project_". The ID must be suffixed with a unique value such as your help project's name (exclude any spaces in the suffix). This is required so that the root namespace container has a unique name that will not cause any conflicts when building MS Help Viewer output.

Next, do one or more of the following:

- To specify project summary comments, add them to the <summary> tag in the R:Project_[HtmlHelpName] node. As noted above, substitute a unique value for the ID's suffix. If you do not have any project summary notes, you may remove this node from the file.
- To exclude a namespace from the help file, add a <member>
 node, set it's name attribute to the namespace name prefixed
 with "N:", and place an <exclude /> tag in it. Any other tags
 are optional. The example above would exclude the unnamed
 global namespace and the CustomControls.Internal
 namespace from the help file if your build tool supports it.
- To specify namespace summary comments, add a <member>
 node, set it's name attribute to the namespace name prefixed
 with "N:", and add the comments to the <summary> tag in the
 node.
- To specify namespace group summary comments, add a
 <member> node, set it's name attribute to the namespace group

name prefixed with "G:", and add the comments to the <summary> tag in the node.

Once you have created the file, you can add it to the build tool that you use to include the comments in the help file when it is built. Refer to your build tool's documentation for further information

▲ See Also

Other Resources

Namespace Comments
Using NamespaceDoc and NamespaceGroupDoc Classes

[v2014.5.31.0] Sandcastle XML Comments Guide



Using NamespaceDoc and NamespaceGroupDoc Classes

If your build tool supports it, namespace comments can also be specified and maintained in your source code by adding an empty NamespaceDoc class to each namespace. When comments for this class are found in the XML comments files they will be used as the namespace comments.

To keep the NamespaceDoc class from appearing in the help file, leave off the public keyword and mark it with a CompilerGenerated attribute. This will cause the class to be automatically ignored when reflection information is generated for the assembly. The following is an example:

```
namespace Company.Product.Widgets
{
    /// <summary>
    /// These are the namespace comments for <c>(
    /// </summary>
    [System.Runtime.CompilerServices.CompilerGeneclass NamespaceDoc
    {
    }
}

*
```

If the project has namespace grouping enabled, you can also maintain the namespace group comments using a NamespaceGroupDoc class in a similar fashion. The following is an example:

C# VB

```
namespace Company.Product
{
    /// <summary>
    /// These are the group comments for namespac
    /// </summary>
    [System.Runtime.CompilerServices.CompilerGene class NamespaceGroupDoc
    {
    }
}
```

▲ See Also

Reference

XMLCommentsExamples.DocumentationInheritance

Other Resources

Namespace Comments

External XML Comments Files

[v2014.5.31.0] Sandcastle XML Comments Guide



Section Elements

Section elements are used to define the content for the different sections of the documentation of a type or member and are always top-level elements that appear at the root of the XML comments block. Select a topic below for more information.

- event
- example
- exception
- param
- permission
- preliminary
- remarks
- returns
- revisionHistory
- seealso
- summary
- threadsafety
- typeparam
- value

▲ See Also

Other Resources

Usage Notes
Namespace Comments
Block Elements
Inline Elements
Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



event

This element is used to list events that can be raised by a type's member.

■ Syntax

This top-level element is valid on any member in which an event can be raised. *eventType* is the name of the event type that can be raised.

```
<event cref="eventType">description</event>
```

▲ Remarks

There should be one event element for each event type that can be raised.

Note

This is a custom XML comments element implemented by Sandcastle. It will not appear in the list of valid elements for XML comments IntelliSense.


```
/// <summary>
/// This performs an action
/// </summary>
/// <event cref="SomethingHappened">This event is
/// something happened.</event>
/// <conceptualLink target="81bf7ad3-45dc-452f-96"
```

```
public void PerformAnAction()
{
    this.OnSomethingHappened();
}
```

▲ See Also

Reference

PerformAnAction

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



example

This element is used to define an example for a type or one of its members to show how it is used.

■ Syntax

This top-level element is valid on all types and members. A description is optional. One or more code elements are typically included to show the example code. More descriptive text can be included between the code elements as needed.

```
<example>
Optional code description

<code language="cs">
/// Example code
</code>
</example>
```

■ Example

```
/// <summary>
/// This returns a random number
/// </summary>
/// <returns>A random number using <see cref="San
/// <example>
/// The following example demonstrates the use of
///
/// <code language="cs">
/// // Get a new random number
```

```
/// SampleClass sc = new SampleClass(10);
///
/// int random = sc.GetRandomNumber();
/// Console.WriteLine("Random value: {0}", random
/// </code>
///
/// <code language="vb">
/// ' Get a new random number
/// Dim sc As SampleClass = New SampleClass(10)
/// Dim random As Integer = sc.GetRandomNumber()
///
/// Console.WriteLine("Random value: {0}", random
/// </code>
/// </example>
/// <conceptualLink target="1abd1992-e3d0-45b4-b4
/// <conceptualLink target="1bef716a-235b-4d96-a2
public int GetRandomNumber()
{
    return new Random(this.SampleNumber).Next();
}
                                                 \blacksquare
```

▲ See Also

Reference

GetRandomNumber

Other Resources

Section Elements code

[v2014.5.31.0] Sandcastle XML Comments Guide



exception

This element is used to list exceptions that can be thrown by a type's member.

■ Syntax

This top-level element is valid on property, method, event, operator, and type conversion members. *exceptionType* is the name of the exception type that can be thrown.

```
<exception cref="exceptionType">description</exception</pre>
```

▲ Remarks

There should be one exception element for each exception type that can be thrown. The description will be used as the display text for the exception in IntelliSense and the Object Browser.


```
/// <summary>
/// This method processes text
/// </summary>
/// <param name="text">The text to process</paran
/// <exception cref="ArgumentNullException">This
/// parameter is null.</exception>
/// <exception cref="ArgumentException">This is t
/// parameter is an empty string.</exception>
/// <exception cref="InvalidOperationException">1
/// is not currently implemented.</exception>
```

```
/// <conceptualLink target="bbd1e65d-c87c-4b46-9@
public void ProcessText(string text)
{
   if(text == null)
        throw new ArgumentNullException("text car

   if(text.Trim().Length == 0)
        throw new ArgumentException("text cannot

   throw new InvalidOperationException();
}</pre>
```

▲ See Also

Reference

ProcessText

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



param

This element is used to describe method parameters.

■ Syntax

This top-level element is valid on methods and operator overloads to describe each parameter. *paramName* is the name of the parameter being referenced.

```
<param name="paramName">Parameter description</parameter)</pre>
```

▲ Remarks

There should be one param element for each method or operator overload parameter. The description will be used as the display text for the parameter in IntelliSense and the Object Browser.


```
/// <summary>
/// Executes a <see cref="SqlCommand" /> with the
/// <paramref name="storedProcName" /> as a store
/// for updating the values of the specified <par
/// </summary>
/// <param name="storedProcName">The stored proce
/// <param name="row">The row to update</param>
/// <conceptualLink target="e54dcff7-f8f3-4a11-9c
/// <conceptualLink target="fa7d6ea0-93ce-41f6-94
public void CallStoredProcedure(string storedProcedure)
```

```
{
       // ...
   }
   Operator Overload
   /// <summary>
   /// Addition operator overload
   /// </summary>
   /// <param name="left">The left value</param>
   /// <param name="right">The right value</param>
   /// <returns>A new instance containing the sum of
   /// sample numbers</returns>
   /// <conceptualLink target="e54dcff7-f8f3-4a11-9c
   public static SampleClass operator +(SampleClass
   {
       return new SampleClass(left.SampleNumber + ri
   }
                                                     lacksquare

▲ See Also
```

Reference

CallStoredProcedure Addition

Other Resources

Section Elements paramref

[v2014.5.31.0] Sandcastle XML Comments Guide



permission

This element is used to define the permissions required to access a type or member.

■ Syntax

This top-level element is valid on all types and type members.

```
<permission cref="permissionTypeOrMember">descrip
```

The cref attribute is used to specify a permission type required to access the type or member. It is typically one of the .NET permission set types such as PermissionSet or SecurityPermission. Use the inner text to describe how the permission applies to the type or member.

▲ Examples

```
/// <summary>
/// The <c>permission</c> element is used to doct
/// a caller must be granted to access this metho
/// </summary>
/// <permission cref="SecurityPermission">
/// <see cref="SecurityPermissionFlag.Execution">
/// </permission>
/// <permission cref="ReflectionPermission">
/// <see cref="ReflectionPermissionFlag.MemberAco"
/// for reflection.</permission>
/// <conceptualLink target="4af64f3f-a9a3-42d7-aso"
[SecurityPermission(SecurityAction.Demand, Execut
```

```
[ReflectionPermission(SecurityAction.Demand, Memk
public void MethodRequiringSpecificPermissions()
{
}
```

▲ See Also

Reference

MethodRequiringSpecificPermissions

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



preliminary

This element is used to indicate that a particular type or member is preliminary and is subject to change.

■ Syntax

This top-level element is valid on all types and type members.

<preliminary />
or
<preliminary>description</preliminary>

The self-closing version will insert a default message into the topic stating "This API is preliminary and subject to change". You can supply inner text on the element to provide a message of your choosing.

▲ Remarks

This element is used to mark individual types or members as preliminary. Build tools such as the Sandcastle Help File Builder provide a project property that can be used to mark the entire help file as preliminary. This saves you from having to annotate every single class and member with this element.

Note

This is a custom XML comments element implemented by Sandcastle. It will not appear in the list of valid elements for XML comments IntelliSense. If applied to a type, the preliminary message will be propagated to all members of the type and will appear in their topics as well.

▲ Examples

```
/// <summary>
/// This method may go away or its signature may
/// </summary>
/// <preliminary />
/// <conceptualLink target="c16bece7-694e-48ca-86
public void PreliminaryExample()
{
}
/// <summary>
/// A temporary method
/// </summary>
/// /// coing away i
/// <conceptualLink target="c16bece7-694e-48ca-86"
public void TemporaryMethod()
}
                                              lacksquare
```

▲ See Also

Reference

PreliminaryExample TemporaryMethod

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



remarks

This element is used to provide a more detailed description of the type or member.

■ Syntax

This top-level element is valid on all types and type members.

```
<remarks>description</remarks>
```

▲ Remarks

Use it when necessary to provide more detailed information about the type or member such as how it is used, its processing, etc. Remarks will only appear in the help file and can be as long as necessary.

▲ Examples

```
/// <summary>
/// The summary is a brief description of the typp
/// be displayed in IntelliSense and the Object E
/// </summary>
/// <remarks>The remarks element should be used t
/// about the type or member such as how it is us
///
/// <para>Remarks will only appear in the help fi
/// necessary.</para></remarks>
/// <conceptualLink target="c3743eaf-9ef2-4d35-8f]
/// <conceptualLink target="3671b61e-0695-4f0f-bc"
```

```
public void SummaryRemarksExample()
{
}
```

▲ See Also

Reference

SummaryRemarksExample

Other Resources

Section Elements

summary

value

[v2014.5.31.0] Sandcastle XML Comments Guide



returns

This element is used to describe the return value of a method.

■ Syntax

This top-level element is valid on all methods that return a value. The description will appear in the Object Browser.

```
<returns>description</returns>
```

▲ Examples

```
/// <summary>
/// The <c>Increment</c> method increments the st
/// </summary>
/// <returns>The new sample number after being ir
/// <conceptualLink target="d0db2290-08bb-40cc-97
/// <conceptualLink target="fa97c10b-a683-4d10-a@public int Increment()
{
    return ++this.SampleNumber;
}</pre>
```

▲ See Also

Reference

Increment

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



revisionHistory

This element is used to display revision history for a type or its members. It originated in the old NDoc and VBCommenter tools but is still useful today to communicate revision history in the documentation.

■ Syntax

This top-level element is valid on all types and type members.

```
<revisionHistory [visible="true|false"]>
    <revision date="MM/dd/yyyy" version="#.#.#" |
        Comments about the revision.
        </revision>
        </revisionHistory>
```

Note

This is a custom XML comments element implemented by the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense.

An optional visible attribute on the revisionHistory parent element controls visibility of the section within the generated topic. If omitted or set to true, the revision history is included. If set to false, it is not included in the topic.

The revision elements describe one or more revisions to the type or member. The following attributes are supported:

date

This attribute specifies the date of the revision. The value is treated as a string field and is displayed as-is in the generated help topic. No locale date formatting is applied.

version

This attribute specifies the version in which the revision was made. As with the date, the value is treated as a string field and is displayed as-is in the generated help topic.

author

This attribute is optional and specifies the name of the person that made the revision. This attribute's value will not appear in the generated help topic.

visible

This optional attribute can be used to control visibility of the revision in the generated topic. If omitted or set to true, the revision entry is included. If set to false, it is not included in the topic.

The content of the revision element can contain other XML comments elements similar to a remarks section to add additional details.

▲ Example

```
Copy
/// <summary>
/// This class demonstrates the revision history
/// </summary>
/// <remarks>The revision history element can be
/// members.</remarks>
/// <revisionHistory>
        <revision date="04/11/2014" version="9.0.</pre>
/// revision will be displayed since the "visible
/// (i.e. default).</revision>
///
        <revision date="04/12/2014" version="9.0.</pre>
/// visible="true">This revision will be displaye
/// "visible=true" attribute setting.</revision>
        <revision date="04/15/2014" version="9.0.</pre>
/// visible="false">This revision will NOT be dis
/// "visible=false" attribute setting.
        </revision>
        <revision date="04/20/2014" version="9.0.</pre>
///
            <para>This revision will be displayed
```

```
/// attribute is not present (i.e. default).</par
///
            <para>Other XML comments elements are
///
/// used as in a remarks section to add additional
            <list type="bullet">
///
                <item>Point #1</item>
                <item>Point #2</item>
                <item>Point #3</item>
///
            </list>
///
        </revision>
/// </revisionHistory>
/// <conceptualLink target="2a973959-9c9a-4b3b-ak
public class RevisionHistoryClass
{
    /// <summary>
    /// This property has revision history
    /// </summary>
    /// <revisionHistory>
            <revision date="05/25/2014" version='
    /// author="EWOODRUFF">Added an example prope
    /// </revisionHistory>
    public string Revision { get; set; }
    /// <summary>
    /// This method has revision history
    /// </summary>
    /// <revisionHistory>
            <revision date="05/25/2014" version='
    /// author="EWOODRUFF">Added an example method
    /// </revisionHistory>
    public void ExampleMethod()
    {
    }
}

ightharpoons
```

▲ See Also

Reference

RevisionHistoryClass Revision ExampleMethod

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



seealso

This element is used to create a link to another API topic or an external website within the **See Also** section of a topic.

- Syntax
 - Code Reference
 - External Reference
- Remarks
- Examples
- See Also

■ Syntax

This top-level element can be used on any type or its members. Any of the following formats may be used based on the need.

Code Reference

```
<seealso cref="member" [qualifyHint="true"] [auto
or

<seealso cref="member" [autoUpgrade="true"]>inner
```

The code reference form uses a cref attribute to specify the member name to which the link should be connected. The compiler will check that the target member name exists and will generate a warning if it is not found. The member name does not have to be fully qualified as long as the reference is within the containing class or if the appropriate using (Imports in VB) statements are present.

If the self-closing form is used, the inner text of the link will be set to the member name without a namespace or type qualifier. You can specify inner text on the element to use an alternate value for the link text or to add a qualifying namespace or type to the member name. By adding the qualifyHint attribute and setting it to true, you can indicate that the inner text should be qualified. Type names are fully qualified with their namespace. Class members are qualified with their class name.

When specifying type, field, property, event, and non-overloaded method names, the name alone will suffice as the cref attribute value. However, when referencing overloaded methods, you must supply parameter type information to help the compiler determine to which overload you are referring. The parameter type names are enclosed in parentheses and can be specified using the types for the language being used (i.e. int or string for C# or Integer or String for VB). They can also be specified using the .NET Framework type names such as Int32, System. Int32, etc. If an overload takes no parameters, just specify an empty set of parentheses. If you want to link to the auto-generated member overloads help topic, you can specify the autoUpgrade attribute. If set to false or omitted, the link will take you directly to the referenced member. If set to true and the method has overloads, the link will take you to the method overloads page instead. If the method has no overloads, it works the same as if set to false and it takes you to the method page. This is equivalent to the same attribute on the MAML codeEntityReference element. Another alternative is to use the fully qualified member name with an "0:" prefix. See the Method Overload Examples below for details.

External Reference

The external reference form is a customization implemented by Sandcastle. It uses an href attribute to specify the URL of an external website or resource to which the link should be connected. If the self-closing form is used, the inner text of the link will be set to the URL. You can specify inner text on the element to use an alternate value for the link text.

The optional target attribute can be used to specify where the content will be opened. If not specified, _blank is used to open the content in a new browser window. The _self option can be used to replace the current topic with the content of the target URL. The _parent and _top options can be used to force the content to be displayed in the parent frame or full window if the current topic is hosted in one of those manners.

The optional alt attribute can be used to specify alternate text to display for the link's tool tip when the mouse hovers over it. If not specified, no alternate text is added for the tool tip.

▲ Remarks

Use this element to insert a link in the **See Also** section of the topic. Use the **see** element to insert an inline link.

Note

Although listed as a top-level element, the seealso element can be used as an inline element like the see element. The difference is that this element will not render anything inline. As with its top-level usage, it only adds an entry in the **See Also** section.

▲ Examples

```
/// <summary>
/// This event is raised when something interest;
/// </summary>
/// <remarks><para>The <see cref="OnSomethingHapk"
/// raise this event. This event uses the genera
/// delegate that passes <see cref="EventArgs.Emk"
/// to the handlers.</para>
///
/// <para>Alternate <c>see</c> syntax: <see cref=
/// </remarks>
/// <seealso cref="EventArgs"/>
/// <seealso cref="EventArgs.Empty"/>
```

```
/// <seealso cref="EventArgs.Empty" qualifyHint="
/// <seealso cref="EventHandler"/>
/// <seealso cref="OnSomethingHappened">XMLCommer
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public event EventHandler SomethingHappened;
/// <summary>
/// This is a protected virtual method used to ra
/// <see cref="SomethingHappened"/> event.
/// </summary>
/// <seealso cref="SomethingHappened"/>
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
protected virtual void OnSomethingHappened()
    var handler = SomethingHappened;
    if(handler != null)
        handler(this, EventArgs.Empty);
}

ightharpoons
```

```
/// </item>
///
      <item>
        <description><see cref="DoSomething(int,</pre>
/// </item>
/// </list>
/// <para>Alternate syntax to link to the overloa
/// <see cref="0:XMLCommentsExamples.SampleClass.</pre>
/// Overloads</see></para>
/// </remarks>
/// <overloads>This method has four overloads</ov
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething()
{
}
/// <summary>
/// This version of the method takes an integer p
/// </summary>
/// <param name="number">A number to use</param>
/// <seealso cref="DoSomething()" autoUpgrade="tr
/// <seealso cref="DoSomething()" />
/// <seealso cref="DoSomething(string)" />
/// <seealso cref="DoSomething(int, string)" />
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething(int number)
{
}
/// <summary>
/// This version of the method takes a string par
/// </summary>
/// <param name="text">A text value to use</paran
/// <seealso cref="DoSomething()"/>
/// <seealso cref="DoSomething(int)"/>
/// <seealso cref="DoSomething(int, string)"/>
/// <conceptualLink target="983fed56-321c-4daf-af
```

```
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething(string text)
{
}
/// <summary>
/// This version of the method takes both an inte
/// </summary>
/// <param name="number">A number to use</param>
/// <param name="text">A text value to use</paran
/// <seealso cref="DoSomething()"/>
/// <seealso cref="DoSomething(int)"/>
/// <seealso cref="DoSomething(string)"/>
/// <conceptualLink target="983fed56-321c-4daf-a1
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething(int number, string text)
{
}
```

```
External Reference Examples
/// <summary>
/// This is used to demonstrate the external refe
/// and <c>seealso</c> elements.
/// </summary>
/// <remarks>
/// <list type="bullet">
/// <item>
        <description>Basic link: <see href="https"</pre>
///
      </item>
///
     <item>
        <description>Link with target: <see href=</pre>
///
          target="_self" /></description>
///
///
      </item>
///
     <item>
///
        <description>Link with alternate text: <<</pre>
           alt="Discuss SHFB on GitHub" /></descri
///
///
      </item>
```

```
/// <item>
        <description>Link with inner text: <see }</pre>
///
          SHFB on GitHub</see></description>
      </item>
/// </list>
/// <para>Equivalent <c>seealso</c> links are sho
/// </remarks>
/// <seealso href="https://GitHub.com/EWSoftware/
/// <seealso href="https://GitHub.com/EWSoftware/
/// <seealso href="https://GitHub.com/EWSoftware/
/// <seealso href="https://GitHub.com/EWSoftware/
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void SeeElementExternalExample()
{
}

ightharpoons
```

▲ See Also

Reference

SomethingHappened
OnSomethingHappened
SeeElementExternalExample
DoSomething

Other Resources

Section Elements see conceptualLink

[v2014.5.31.0] Sandcastle XML Comments Guide



summary

This element is used to provide a brief description of the type or member.

■ Syntax

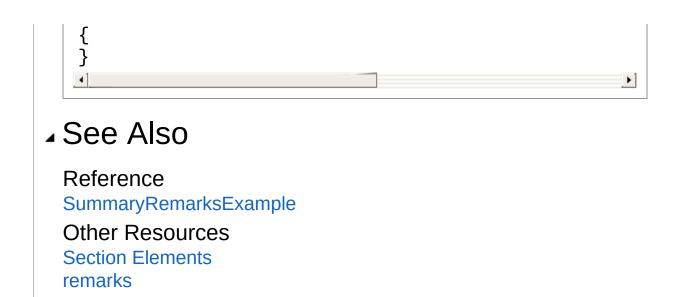
This top-level element is valid on all types and type members.

```
<summary>description</summary>
```

▲ Remarks

The description will appear in IntelliSense and the Object Browser. As such, it should be kept brief. Use the remarks element to provide more detailed information about the type or member.


```
/// <summary>
/// The summary is a brief description of the typ
/// be displayed in IntelliSense and the Object E
/// </summary>
/// <remarks>The remarks element should be used t
/// about the type or member such as how it is us
///
/// <para>Remarks will only appear in the help fi
/// necessary.</para></remarks>
/// <conceptualLink target="c3743eaf-9ef2-4d35-8f]
/// <conceptualLink target="3671b61e-0695-4f0f-bc]
public void SummaryRemarksExample()
```



[v2014.5.31.0] Sandcastle XML Comments Guide



threadsafety

This element is used to indicate whether or not a class or structure's static and instance members are safe for use in multi-threaded scenarios.

■ Syntax

This top-level element is valid on all classes and structures.

```
<threadsafety static="true | false" instance="tru"
</pre>
```

The static attribute specifies whether static members of the class or structure are safe for use in multi-threaded operations. The instance attribute specifies whether instance members of the class or structure are safe for use in multi-threaded operations. If neither attribute is specified, it is the same as setting static to true and instance to false.

Note

This is a custom XML comments element implemented by Sandcastle. It will not appear in the list of valid elements for XML comments IntelliSense.

■ Examples

```
/// <summary>
/// This class demonstrates the <c>threadsafety<///>
/// </summary>
/// <threadsafety static="true" instance="false"
/// <conceptualLink target="fb4625cb-52d0-428e-9c
```

```
public class ThreadSafetyClass
{
    /// <summary>
    /// Per the <c>threadsafety</c> XML comments
    /// indicated that static methods like this (
    /// </summary>
    /// <conceptualLink target="fb4625cb-52d0-428"
    public static void StaticMethod()
    {
        // Thread-safe code goes here
    }
    /// <summary>
    /// Per the <c>threadsafety</c> XML comments
    /// indicated that instance method like this
    /// </summary>
    /// <conceptualLink target="fb4625cb-52d0-428"
    public void InstanceMethod()
    {
        // Non-thread-safe code goes here
    }
}
                                                 F
```

▲ See Also

Reference

ThreadSafetyClass

Other Resources

Section Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



typeparam

This element is used to describe generic parameters on generic types and methods.

■ Syntax

This top-level element is valid on generic types and generic methods to describe each generic type parameter. *typeParamName* is the name of the generic type parameter being referenced.

```
<typeparam name="typeParamName">Type parameter de
```

▲ Remarks

There should be one typeparam element for each generic type parameter in the class or method declaration. The description will be used as the display text for the parameter in IntelliSense and the Object Browser.

▲ Examples

```
/// <summary>
/// This class is used to demonstrate the various
/// related to generics. It serves no useful pur
/// </summary>
/// <typeparam name="T1">This is the first generi
/// <typeparam name="T2">This is the second gener
/// be or derive from <see cref="EventArgs"/>.</t
/// <conceptualLink target="163cae15-9020-4095-9k
public class GenericClass<T1, T2> where T2 : Ever
```

```
{
    /// <summary>
   /// This is a property that gets or sets an i
    /// by the generic type argument <typeparamre
    /// </summary>
    /// <conceptualLink target="073a5ae1-828f-4ba
    public T1 Property { get; set; }
    /// <summary>
    /// This is a method with an argument.
    /// </summary>
    /// <param name="argument"> This is an argume
    /// the generic type argument <typeparamref r
    /// <conceptualLink target="073a5ae1-828f-4ba
    public void Method(T1 argument)
    }
    /// <summary>
    /// This is a generic method that takes two (
    /// </summary>
    /// <typeparam name="T3">This is a generic ty
    /// argument.</typeparam>
    /// <typeparam name="T4">This is a generic ty
    /// value.</typeparam>
    /// <param name="argument">This is an argumer
    /// the generic type argument <typeparamref r
    /// <returns>The default value of the type sp
    /// argument <typeparamref name="T4"/>.</retu
    /// <conceptualLink target="163cae15-9020-409"
    /// <conceptualLink target="073a5ae1-828f-4ba
    public T4 GenericMethod<T3, T4>(T3 argument)
    {
        return default(T4);
    /// <summary>
    /// This is an event that takes a generic arc
    /// </summary>
```

```
/// <remarks>The <see cref="Delegate">delegat
    /// <see cref="EventHandler{T}"/> bound to the
    /// generic type argument <typeparamref name=
    /// </remarks>
    /// <conceptualLink target="073a5ae1-828f-4ba
    public event EventHandler<T2> SomethingHapper
    /// <summary>
    /// This is a protected virtual method used t
    /// <see cref="SomethingHappened"/> event.
    /// </summary>
    /// <param name="e">Arguments for the event (
    /// the generic type argument <typeparamref r
    /// <conceptualLink target="073a5ae1-828f-4ba
    protected virtual void OnSomethingHappened(T2
        var handler = SomethingHappened;
        if(handler != null)
            handler(this, e);
   }
}
```

▲ See Also

Reference

GenericClassT1, T2 GenericMethodT3, T4

Other Resources

Section Elements typeparamref

[v2014.5.31.0] Sandcastle XML Comments Guide



value

This element is used to provide a more detailed description of a property value.

■ Syntax

This top-level element is valid on all properties. Use it when necessary to provide more detailed information about a property's value such as how it is used, valid values, etc.

```
<value>description</value>
```

▲ Remarks

The value description will only appear in the help file and can be as long as necessary. It is similar in nature to the remarks element but is specific to property members.

■ Examples

```
/// <summary>
/// This is used to get or set the sample number
/// </summary>
/// <value>The value can be any valid integer</va
/// <conceptualLink target="f512d714-e100-4296-91
public int SampleNumber { get; set; }
```

▲ See Also

Reference SampleNumber Other Resources Section Elements remarks

[v2014.5.31.0] Sandcastle XML Comments Guide



Block Elements

Block elements format text within the top-level section elements. They are typically used to add structure to the text inside those elements. Select a topic below for more information.

- code
- list
- note
- para

▲ See Also

Other Resources

Usage Notes
Namespace Comments
Section Elements
Inline Elements
Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



code

This element is used to indicate that a multi-line section of text should be formatted as a code block.

- Syntax
- Literal XML And Other Special Characters
- Attributes
 - language
 - title
 - source, region, and removeRegionMarkers
 - tabSize
 - numberLines
 - outlining
 - keepSeeTags
- Nested Code Elements
- Examples
- See Also

■ Syntax

This inline block element can be used within any other element.



Use this element to mark a multi-line block of lines as code. Use the c element to mark inline text as code.



The language and title attributes are supported by Sandcastle. All other attributes are implemented by the **Code Block Component** supplied with the Sandcastle Help File Builder. Refer to its help file for more information about the build component.

Literal XML And Other Special Characters

If you want to include XML or other unencoded information you can do one of the following:

- Utilize the source attribute to read it in from an external file instead.
- Use a CDATA section to encapsulate the code so that special characters and formatting are preserved. For example:

Be sure to start the example code on a new line after the

opening "<! [CDATA[" tag as shown so that the leading whitespace can be normalized correctly.

Note

The inclusion of a CDATA tag anywhere in the code comments care Visual Studio 2005 to lock up when you type an angle bracket with comment block. If you are affected by this issue, there is a hot fix https://connect.microsoft.com/VisualStudio/Downloads/Download Download Dow

▲ Attributes

The following attributes can be used to control how the code block is handled.

language

This attribute allows you to define the code language. Although optional, it is recommended that you always specify a language attribute if using Sandcastle alone. Many of the presentation styles rely on the language attribute to classify the code block and handle it in some way such as attaching it to a language filter or grouping it with like code blocks. The Sandcastle Help File Builder's code block component will apply a default language if one is not specified. The possible language values are as follows:

Language ID (case-insensitive)	Language Syntax Used
cs, C#, CSharp	C#
cpp, cpp#, C++, CPlusPlus	C++
С	С
fs, f#, FSharp, fscript	F#
EcmaScript, js, JavaScript	JavaScript
jscript, jscript#, jscriptnet, JScript.NET	JScript.NET
VB, VB#, vbnet, VB.NET	VB/VB.NET

vbs, vbscript	VBScript
htm, html, xml, xsl	XML
XAML	XAML
jsharp, J#	J#
sql, sql server, sqlserver	SQL script
pshell, powershell, ps1	PowerShell script
py, python	Python
Anything else (i.e. "none")	No language, no special handling.

▲ Important

The full name "language" should be used for the attribute name rather than the abbreviated form "lang". The reason is that the XSL transformations used by the presentation styles expect the full name when they search for the language attribute. The abbreviated form is a legacy syntax that should be avoided. The code block component will auto-correct the attribute name so that it is not missed by the transformations. It is recommended that you always specify a language attribute.

title

This attribute allows you to add a title that appears before the code block. An example of its use would be to label the example with a description. If omitted, the language name will appear for the title. If you do not want a title on a particular block, set the title attribute to a single space (" ").

source, region, and removeRegionMarkers

These three attributes are extensions implemented by the code block build component. The source attribute is used to specify that the code block's content should be read from an external source code file. If used alone, the entire file is imported. The optional region attribute can be used to limit the code to a specific section

of the file delimited with the named #region (#pragma region for C++). The #region and #endregion tags are excluded from the extracted section of code.

This is not to be confused with the XML comments include element. This extension is intended to extract code from actual source files. This allows you to manage your code examples in buildable projects to test them for correctness as a project is developed and altered in the future. It also saves you from managing the code in the XML comments and does not require that the code be HTML encoded as it is when written in the comments. The code will be HTML encoded when it is read in for processing. When used, it is assumed that there is no code within the code element itself and thus it will always be self-closing. Here are some examples:

Retrieve all code from an external file and use the VB.NET syntax to color it. The path is relative to the code block component's basePath configuration element. Typically, the base path is the path of the documentation project file. So, if your documentation project is in a subfolder and the code is in the parent folder in an *Examples* subfolder, the path would look like the following example.

```
<code source="..\Examples\WholeDemo.vb" language=
</pre>
```

Retrieve a specific #region from an external file.

```
<code source="..\Examples\SeveralExamples.vb"
  region="Example 1" language="vbnet" title="Exan
</pre>
```

Note that VB.NET does not allow #Region and #End Region within a method body. Other file types such as XML, XAML, and SQL script do not understand #region statements. However, if you want to extract a region from a VB.NET method body or a section of another file type, you can add the region statements in comments to workaround the limitation. The component will still find it and extract the region.

Commented Region Examples	Copy _
VB.NET Example	

```
Public Sub SomeMethod()
    ' #Region "VB.NET Snippet"
    Dim x As Integer
    For x = 1 To 10
        Console.WriteLine(x)
    Next x
    ' #End Region
End Sub
XAML Example
<Style x:Key="SpecialButton" TargetType="{x:Type</pre>
  <Style.Triggers>
    <!-- #region XAML Snippet -->
    <Trigger Property="Button.IsMouseOver" Value=</pre>
      <Setter Property = "Background" Value="Red'</pre>
    </Trigger>
    <!-- #endregion -->
  </Style.Triggers>
</Style>
SQL Example
-- #region SQL snippet
Select *
From
      tblName
Where LastName = 'SMITH'
-- #endregion
/* #region Another snippet */
Select *
From tblName
Where FirstName = 'JOHN'
/* #endregion */
```

The removeRegionMarkers attribute can be used to specify whether or not region markers within the imported code file or region are removed. If not specified or set to false, any nested region markers are left in the imported code block. If specified and set to true, nested region markers are removed from the imported code. This is useful for removing nested region markers from larger code samples that contain smaller snippets of code used in other examples. The default setting (false) can be overridden via the component configuration.

tabSize

When the code blocks are formatted by the build component, tab characters are replaced with a set number of spaces to preserve formatting. This attribute can be used to override the default setting for a language which is specified in the syntax file. For example, if the default tab size for a language is four, adding tabSize="8" will force it to use eight spaces instead. If set to zero, the syntax file setting is used. This attribute sets the default tab size for unknown languages when used in the component's configuration.

numberLines

This attribute allows you to override the default setting in the component's configuration. For example, if the default setting is false to turn off line numbering, you can add numberLines="true" to enable numbering on a specific code example.

outlining

This attribute allows you to override the default setting in the component's configuration. For example, if the default setting is false to not add collapsible regions, you can add outlining="true" to enable collapsible regions on a specific code example. Note that if a code block contains no #region or #if blocks, outlining is automatically disabled and it will not reserve space in the margin for the markers.

keepSeeTags

When set to true, this attribute allows you to tell the build component's code colorizer to preserve see tags within the code so that they can be rendered as clickable links to the related topic. If set to false, the default, any see tags within the code will be colorized and passed through as literal text. When using this option, you may find that you need to specify inner text for the see tag so that the link text appears as you want it. If the self-closing version of the tag is used, Sandcastle will generally set the link text to the name of the item plus any parameters if it is a generic type or takes parameters

which may not be appropriate within a code sample.

▲ Nested Code Elements

You can import multiple code snippets into one common colorized code block by nesting code elements within a parent code element. On nested code elements, only the source and region attributes will be utilized. All other attributes that control colorization of the merged code block will be retrieved from the parent code element.

```
Nested Code Elements Example

<code title="Nested code elements example" langua
<code source="..\Class1.vb" region="Snippet #1" /
<code source="..\Class1.vb" region="Snippet #2" /
</code>
```

Literal code can also be mixed in between the nested code elements. However, there are some limitations. All elements and literal code should be flush left within the parent code element or you will end up with additional spaces before the first line of each nested code element. There may still be additional spaces before the literal code. This is caused by the indentation included when the XML comments file is created by the compiler. If this is a problem, move the example to an include file and remove the excess leading whitespace. Use an include element to bring it into the XML comments. When done this way, the compiler preserves the spacing when writing the example out to the XML comments file.

▲ Examples

```
/// <summary>
/// This returns a random number
/// </summary>
/// <returns>A random number using <see cref="San
/// <example>
/// The following example demonstrates the use of
/// <code language="cs">
/// // Get a new random number
/// SampleClass sc = new SampleClass(10);
///
/// int random = sc.GetRandomNumber();
///
/// Console.WriteLine("Random value: {0}", random
/// </code>
///
/// <code language="vb">
/// ' Get a new random number
/// Dim sc As SampleClass = New SampleClass(10)
/// Dim random As Integer = sc.GetRandomNumber()
///
/// Console.WriteLine("Random value: {0}", random
/// </code>
/// </example>
/// <conceptualLink target="1abd1992-e3d0-45b4-b4
/// <conceptualLink target="1bef716a-235b-4d96-a2
public int GetRandomNumber()
{
    return new Random(this.SampleNumber).Next();
}
                                                Þ
```

▲ See Also

Reference GetRandomNumber Other Resources Block Elements example

[v2014.5.31.0] Sandcastle XML Comments Guide



list

This element describes content that should be displayed as a list or a table.

■ Syntax

This inline block element can be used within any other element.

```
<list type="bullet">
  <item>
    <term>Optional term</term>
    <description>Required description. If term i
listed as the item element's inner text.</descrip
  </item>
</list>
or
<list type="number" [start="###"]>
  <item>
    <term>Optional term
    <description>Required description. If term i
listed as the item element's inner text.</descrip
  </item>
</list>
or
<list type="definition">
  <item>
```

```
<term>Required term</term>
    <description>Required description</description</pre>
  </item>
</list>
or
<list type="table">
  <listHeader>
    <term>Multi-column table header</term>
    <term>Add additional term or description elen
  </listHeader>
  <item>
    <description>Multi-column table</description>
    <description>Add additional term or descripti
  </item>
</list>

ightharpoons
```

The type attribute defines the list type. The list type value should be one of the values shown below. The list item terms and descriptions can be any valid inline or block XML comments similar to a remarks element.

A Important

The type attribute value is case-sensitive. Unrecognized values will cause the entire list element to be ignored with the item content rendered inline without any special list formatting.

The type attribute can be any of the following values. The requirements of each format are noted in the description.

List Type	Description
bullet	This renders a bulleted list. Each item element will

render a new bulleted item. If term and description elements are omitted, the inner text is rendered after the bullet point. The same thing happens if a description element is included by itself. If both a term and a description element are present, the term is highlighted and separated from the description by a dash.

number

This renders a numbered list. Each item element will render a new numbered item. If term and description elements are omitted, the inner text is rendered after the number. The same thing happens if a description element is included by itself. If both a term and a description element are present, the term is highlighted and separated from the description by a dash.

The number style supports an optional start attribute that can be used to indicate the starting number for the numbered list. This lets you create a list at a later point in the topic that continues numbering from the end point of a prior list. You are responsible for providing the appropriate starting number for the new list.

Note

start is a custom attribute implemented by Sandcastle.

definition A definition list. A term and description element are required within each item element. The term is highlighted and rendered on a separate line with the description following in a new paragraph.

table

A multi-column table. The item elements create new rows in the table. With this type, the optional listHeader element can be used to define the column headers. Use a term or a description element to define the columns in the listHeader and each item element.


```
/// <summary>
/// This method shows various examples of the <c>
/// </summary>
/// <remarks>
/// <para>A simple bulleted list. The <c>term</c
/// elements are optional for simple string descr
/// <list type="bullet">
/// <item>First item</item>
/// <item>Second item</item>
/// <item>Third item</item>
/// </list>
///
/// <para>Bullet list with terms and definitions.
/// separated from the definition with a dash.
/// <list type="bullet">
/// <item>
///
       <term>maxLen</term>
       <description>field must contain no more t
/// of characters</description>
/// </item>
/// <item>
///
       <term>minLen</term>
       <description>field must contain at least
/// of characters</description>
/// </item>
/// <item>
///
       <term>maxVal</term>
       <description>field must contain a number
/// specified value</description>
/// </item>
/// <item>
       <term>minVal</term>
       <description>field must contain a number
/// specified value</description>
/// </item>
///
     <item>
```

```
/// <term>pattern</term>
/// <description>field must match the specifi
/// </description>
/// </item>
/// </list>
///
/// <para>A simple numbered list. The <c>term</c
/// elements are optional for simple string descr
///
/// <list type="number">
/// <item>First item</item>
/// <item>Second item</item>
/// <item>Third item</item>
/// </list>
///
/// <para>This next numbered list uses the option
/// continue numbering where the last one left of
///
/// <list type="number" start="4">
/// <item>Fourth item</item>
/// <item>Fifth item</item>
/// <item>Sixth item</item>
/// </list>
///
/// <para>Numbered list with terms and definition
///
/// <list type="number">
/// <item>
///
       <term>maxLen</term>
       <description>field must contain no more t
/// of characters</description>
/// </item>
/// <item>
///
       <term>minLen</term>
       <description>field must contain at least
/// of characters</description>
/// </item>
/// <item>
       <term>maxVal</term>
```

```
<description>field must contain a number
/// specified value</description>
/// </item>
///
///
     <item>
       <term>minVal</term>
       <description>field must contain a number
/// specified value</description>
/// </item>
/// <item>
///
       <term>pattern</term>
       <description>field must match the specifi
/// </description>
/// </item>
/// </list>
///
/// <para>Definition list.</para>
///
/// <list type="definition">
/// <item>
///
       <term>maxLen</term>
       <description>field must contain no more t
/// of characters</description>
/// </item>
/// <item>
///
///
       <term>minLen</term>
       <description>field must contain at least
/// of characters</description>
/// </item>
/// <item>
///
        <term>maxVal</term>
///
       <description>field must contain a number
/// specified value</description>
///
     </item>
///
      <item>
       <term>minVal</term>
///
///
       <description>field must contain a number
/// specified value</description>
///
      </item>
///
      <item>
```

```
/// <term>pattern</term>
       <description>field must match the specifi
///
/// </description>
/// </item>
/// </list>
///
/// <para>Two-column table list with terms and de
///
/// <list type="table">
      <listheader>
///
///
///
        <term>Item</term>
       <description>Description</description>
/// </listheader>
/// <item>
///
       <term>maxLen</term>
///
        <description>field must contain no more t
/// of characters</description>
/// </item>
/// <item>
///
       <term>minLen</term>
       <description>field must contain at least
/// of characters</description>
/// </item>
/// <item>
///
///
       <term>maxVal</term>
       <description>field must contain a number
/// specified value</description>
///
    </item>
/// <item>
///
        <term>minVal</term>
///
       <description>field must contain a number
/// specified value</description>
/// </item>
///
      <item>
///
       <term>pattern</term>
       <description>field must match the specifi
/// </description>
      </item>
/// </list>
```

```
///
/// <para>A table with multiple columns. <c>tern
/// can be used to create the columns in each row
///
/// <list type="table">
      <listheader>
        <term>Column 1</term>
///
///
        <term>Column 2</term>
        <term>Column 3</term>
///
        <term>Column 4</term>
///
/// </listheader>
///
    <item>
///
        <term>R1, C1</term>
        <term>R1, C2</term>
///
        <term>R1, C3</term>
///
        <term>R1, C4</term>
///
/// </item>
/// <item>
        <description>R2, C1</description>
///
///
        <description>R2, C2</description>
        <description>R2, C3</description>
///
        <description>R2, C4</description>
      </item>
/// </list>
/// </remarks>
/// <conceptualLink target="e433d846-db15-4ac8-as
public void VariousListExamples()
{
}

ightharpoons
```

Reference

VariousListExamples

Other Resources

Block Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



note

This element is used to create a note-like section within a topic to draw attention to some important information.

■ Syntax

This inline block element can be used within any other element.

<note type="noteType">note content</note>

The type attribute defines the note type. The *noteType* value should be one of the values shown below. If omitted, a default note type of "note" is assumed. The note content can be any valid inline or block XML comments similar to a remarks element.

▲ Important

The type attribute value is case-sensitive. Unrecognized values will default to the "note" style.

The type attribute can be any of the following values which fall into four different categories that use different icons to the left of the note title. The title will generally be the note type name or an expanded form of it.

Category	Type Value		
General note	note, tip, implement, caller, inherit		
Cautionary	caution, warning, important		
Security	security, security note		

Language cs, CSharp, c#, C#, visual c# note, cpp, CPP, c++, note C++, visual c++ note, vb, VB, VisualBasic, visual basic note, JSharp, j#, J#, visual j# note

Note

This is a custom XML comments element implemented by Sandcastle. It will not appear in the list of valid elements for XML comments IntelliSense.

▲ Examples

```
/// <summary>
/// This shows the result of the various <c>note<
/// </summary>
/// <remarks>
/// <para>These are various examples of the diffe
///
/// <note>
/// This example demonstrates the handling of a <
/// defined type. It defaults to the "note" styl
/// </note>
///
/// <note type="tip">
/// Always document your code to help others unde
/// </note>
///
/// <note type="implement">
/// Override this method in a derived class to do
/// </note>
///
/// <note type="caller">
/// Calling this implementation will have no effe
/// </note>
///
/// <note type="inherit">
/// Types inheriting this base method will have r
/// </note>
```

```
///
/// <note type="caution">
/// Use of this method is not recommended.
/// </note>
///
/// <note type="warning">
/// XML is case-sensitive so the note type must t
/// it to be interpreted correctly.
/// </note>
///
/// <note type="important">
/// Calling this method excessively will only slo
/// </note>
///
/// <note type="security">
/// It is always safe to call this method.
/// </note>
///
/// <note type="security note">
/// This method requires no special privileges
/// </note>
///
/// <note type="C#">
/// Use parenthesis when calling this method in (
/// </note>
///
/// <note type="VB">
/// Parenthesis are not required when calling thi
/// </note>
///
/// <note type="C++">
/// Use parenthesis when calling this method in (
/// </note>
///
/// <note type="J#">
/// Use parenthesis when calling this method in .
/// </note>
///
/// <para>See the <conceptualLink target="4302a60"
```

```
/// topic for a full list of all possible note ty
/// </remarks>
public virtual void VariousNoteExamples()
{
}
```

Reference

VariousNoteExamples

Other Resources

Block Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



para

This element is used to start a new paragraph within other elements.

■ Syntax

This inline block element can be used within any other element.

```
<para>paragraph content</para>
```

▲ Remarks

As an alternative, you can use the HTML or tags to create paragraphs. Making the element self-closing or empty will cause the element to be ignored unless it creates an initial paragraph break.

▲ Example

```
/// <summary>
/// A simple demonstration of the <c>para</c> ele
/// </summary>
/// <remarks>The <c>para</c> element on the first
/// optional.
///
/// <para>Subsequent blocks of text must be wrap;
/// paragraphs.</para>
///
/// <para>Paragraph 1.</para>
/// <para>Paragraph 2.</para>
/// <para />
/// <para />
```

```
/// <para></para>
/// <para>Self-closing and empty paragraphs are i
/// an initial paragraph break.</para>
/// <para>Paragraph 4.</para>
///
/// HTML paragraph elements can also be used
/// They work the same way.</remarks>
/// <conceptualLink target="c7973ac7-5a4f-4e4d-97
public void ParagraphExample()
{
}</pre>
```

Reference

ParagraphExample

Other Resources

Block Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



Inline Elements

Inline elements are typically used inside the other section and block elements to provide formatting or links to other API topics. Select a topic below for more information.

- C
- paramref
- see
- typeparamref

▲ See Also

Other Resources

Usage Notes
Namespace Comments
Section Elements
Block Elements
Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



C

This element is used to indicate that text within a description should be marked as code.

■ Syntax

This inline element can be used within any other element.

```
<c>text</c>
```

▲ Remarks

Use this element to mark inline text as code. Use the code element to mark a multi-line block of lines as code.

▲ Example

```
/// <summary>
/// The <c>Increment</c> method increments the st
/// </summary>
/// <returns>The new sample number after being ir
/// <conceptualLink target="d0db2290-08bb-40cc-97
/// <conceptualLink target="fa97c10b-a683-4d10-a@public int Increment()
{
    return ++this.SampleNumber;
}</pre>
```

Reference Increment Other Resources Inline Elements code

[v2014.5.31.0] Sandcastle XML Comments Guide



paramref

This element is used to indicate that a word in the comments refers to a parameter, typically a method parameter.

■ Syntax

This inline element can be used within any other element. *paramName* is the name of the parameter being referenced.

```
copy _
copy _
copy _
```

▲ Example

```
/// <summary>
/// Executes a <see cref="SqlCommand" /> with the
/// <paramref name="storedProcName" /> as a store
/// for updating the values of the specified <par
/// </param name="storedProcName">The stored proce
/// <param name="row">The row to update</param>
/// <conceptualLink target="e54dcff7-f8f3-4a11-9c
/// <conceptualLink target="fa7d6ea0-93ce-41f6-94
public void CallStoredProcedure(string storedProcedure)
{
    // ...
}</pre>
```

▲ See Also

Reference
CallStoredProcedure
Addition
Other Resources
Inline Elements
param

[v2014.5.31.0] Sandcastle XML Comments Guide



see

This element is used to create an inline link to another API topic or an external website within the text in which it occurs. It can also be used to apply language-specific formatting to a limited set of keywords.

- Syntax
 - Code Reference
 - External Reference
 - Language Keyword Formatting
- Remarks
- Examples
- See Also

■ Syntax

This inline element can be used within any other element. Any of the following formats may be used based on the need.

Code Reference

```
<see cref="member" [qualifyHint="true"] [autoUpgr
or

<see cref="member" [autoUpgrade="true"]>inner te>
```

The code reference form uses a cref attribute to specify the member name to which the link should be connected. The compiler will check that the target member name exists and will generate a warning if it is not found. The member name does not have to be fully qualified as long as the reference is within the containing class or if the appropriate using (Imports in VB) statements are present.

If the self-closing form is used, the inner text of the link will be set to the member name without a namespace or type qualifier. You can specify inner text on the element to use an alternate value for the link text or to add a qualifying namespace or type to the member name. By adding the qualifyHint attribute and setting it to true, you can indicate that the inner text should be qualified. Type names are fully qualified with their namespace. Class members are qualified with their class name.

When specifying type, field, property, event, and non-overloaded method names, the name alone will suffice as the cref attribute value. However, when referencing overloaded methods, you must supply parameter type information to help the compiler determine to which overload you are referring. The parameter type names are enclosed in parentheses and can be specified using the types for the language being used (i.e. int or string for C# or Integer or String for VB). They can also be specified using the .NET Framework type names such as Int32, System. Int32, etc. If an overload takes no parameters, just specify an empty set of parentheses. If you want to link to the auto-generated member overloads help topic, you can specify the autoUpgrade attribute. If set to false or omitted, the link will take you directly to the referenced member. If set to true and the method has overloads, the link will take you to the method overloads page instead. If the method has no overloads, it works the same as if set to false and it takes you to the method page. This is equivalent to the same attribute on the MAML codeEntityReference element. Another alternative is to use the fully qualified member name with an "0:" prefix. See the Method Overload Examples below for details.

External Reference

The external reference form is a customization implemented by Sandcastle. It uses an href attribute to specify the URL of an external website or resource to which the link should be connected. If the self-closing form is used, the inner text of the link will be set to

the URL. You can specify inner text on the element to use an alternate value for the link text.

The optional target attribute can be used to specify where the content will be opened. If not specified, _blank is used to open the content in a new browser window. The _self option can be used to replace the current topic with the content of the target URL. The _parent and _top options can be used to force the content to be displayed in the parent frame or full window if the current topic is hosted in one of those manners.

The optional alt attribute can be used to specify alternate text to display for the link's tooltip when the mouse hovers over it. If not specified, no alternate text is added for the tooltip.

Language Keyword Formatting

```
<see langword="null | Nothing | nullptr | static
   Overridable | true | True | false | False | &
    sealed | NotInheritable |
    async | await | async/await | Async | Await |
    async/let!" />
```

The language keyword formatting form uses a langword attribute to specify a language keyword that will receive special handling when the topic is viewed. When one of the noted keywords is used, the keyword text displayed is based on the language selected in the language filter if the presentation style has one. If a keyword other than one of those listed is specified, it is passed through with no special formatting or handling.

▲ Remarks

Use this element to insert an inline link. Use the seealso element to insert a link in the **See Also** section of the topic.

▲ Examples



```
/// This event is raised when something interesti
/// </summary>
/// <remarks><para>The <see cref="OnSomethingHapg
/// raise this event. This event uses the general
/// delegate that passes <see cref="EventArgs.Emg
/// to the handlers.</para>
///
/// <para>Alternate <c>see</c> syntax: <see cref=
/// </remarks>
/// <seealso cref="EventArgs"/>
/// <seealso cref="EventArgs.Empty"/>
/// <seealso cref="EventArgs.Empty" qualifyHint='</pre>
/// <seealso cref="EventHandler"/>
/// <seealso cref="OnSomethingHappened">XMLCommer
/// <conceptualLink target="983fed56-321c-4daf-a1
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public event EventHandler SomethingHappened;
/// <summary>
/// This is a protected virtual method used to ra
/// <see cref="SomethingHappened"/> event.
/// </summary>
/// <seealso cref="SomethingHappened"/>
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
protected virtual void OnSomethingHappened()
{
    var handler = SomethingHappened;
    if(handler != null)
        handler(this, EventArgs.Empty);
}
```

```
Method Overload Examples

/// <summary>
/// This version of the method takes no parameter
/// </summary>
```

```
/// <remarks><para>To see all overloads, refer to
/// <see cref="DoSomething()" autoUpgrade="true"</pre>
///
/// <para>Other overloads:</para>
/// <list type="bullet">
/// <item>
        <description><see cref="DoSomething(int)"</pre>
///
///
      </item>
/// <item>
/// <desc
/// </item>
        <description><see cref="DoSomething(strir")</pre>
/// <item>
        <description><see cref="DoSomething(int,</pre>
/// </item>
/// </list>
///
/// <para>Alternate syntax to link to the overloa
/// <see cref="0:XMLCommentsExamples.SampleClass.</pre>
/// Overloads</see></para>
/// </remarks>
/// <overloads>This method has four overloads</ov
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething()
{
}
/// <summary>
/// This version of the method takes an integer p
/// </summary>
/// <param name="number">A number to use</param>
/// <seealso cref="DoSomething()" autoUpgrade="tr
/// <seealso cref="DoSomething()" />
/// <seealso cref="DoSomething(string)" />
/// <seealso cref="DoSomething(int, string)" />
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething(int number)
```

```
{
}
/// <summary>
/// This version of the method takes a string par
/// </summary>
/// <param name="text">A text value to use</paran
/// <seealso cref="DoSomething()"/>
/// <seealso cref="DoSomething(int)"/>
/// <seealso cref="DoSomething(int, string)"/>
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething(string text)
{
}
/// <summary>
/// This version of the method takes both an inte
/// </summary>
/// <param name="number">A number to use</param>
/// <param name="text">A text value to use</paran
/// <seealso cref="DoSomething()"/>
/// <seealso cref="DoSomething(int)"/>
/// <seealso cref="DoSomething(string)"/>
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-bf
public void DoSomething(int number, string text)
{
}
```

```
/// External Reference Examples

/// <summary>
/// This is used to demonstrate the external refe
/// and <c>seealso</c> elements.
/// </summary>
/// <remarks>
/// <list type="bullet">
```

```
///
      <item>
///
        <description>Basic link: <see href="https"</pre>
///
      </item>
///
      <item>
///
        <description>Link with target: <see href=</pre>
          target="_self" /></description>
///
///
      </item>
///
      <item>
        <description>Link with alternate text: <<</pre>
///
          alt="Discuss SHFB on GitHub" /></descri
///
///
      </item>
///
      <item>
///
        <description>Link with inner text: <see }</pre>
          SHFB on GitHub</see></description>
///
      </item>
/// </list>
/// <para>Equivalent <c>seealso</c> links are sho
/// </remarks>
/// <seealso href="https://GitHub.com/EWSoftware/
/// <seealso href="https://GitHub.com/EWSoftware/
/// <seealso href="https://GitHub.com/EWSoftware/
/// <seealso href="https://GitHub.com/EWSoftware/
/// <conceptualLink target="983fed56-321c-4daf-af
/// <conceptualLink target="16cdb957-a35b-4c17-b1
public void SeeElementExternalExample()
{
}
```

```
/// <summary>
/// This demonstrates the <c>see</c> element's <c
/// </summary>
/// <returns>This method is <see langword="static"
/// <see langword="null"/>.</returns>
/// <conceptualLink target="983fed56-321c-4daf-afpublic static string SeeLangWordExamples()
{
```

```
return null;
}
```

Reference

SomethingHappened
OnSomethingHappened
SeeElementExternalExample
SeeLangWordExamples
DoSomething

Other Resources

Inline Elements seealso conceptualLink

[v2014.5.31.0] Sandcastle XML Comments Guide



typeparamref

This element is used to indicate that a word in the comments refers to a type parameter on a generic type.

■ Syntax

This inline element can be used within any other element on a generic type or its members. *typeParamName* is the name of the parameter being referenced.

```
<typeparamref name="typeParamName" />
```

▲ Examples

```
/// <summary>
/// This class is used to demonstrate the various
/// related to generics. It serves no useful pur
/// </summary>
/// <typeparam name="T1">This is the first generi
/// <typeparam name="T2">This is the second gener
/// be or derive from <see cref="EventArgs"/>.</t
/// <conceptualLink target="163cae15-9020-4095-9k
public class GenericClass<T1, T2> where T2 : Ever
{
    /// <summary>
    /// This is a property that gets or sets an i
    /// by the generic type argument <typeparamre
    /// </summary>
    /// <conceptualLink target="073a5ae1-828f-4ba
    public T1 Property { get; set; }
```

```
/// <summary>
/// This is a method with an argument.
/// </summary>
/// <param name="argument"> This is an argume
/// the generic type argument <typeparamref r
/// <conceptualLink target="073a5ae1-828f-4ba
public void Method(T1 argument)
{
}
/// <summary>
/// This is a generic method that takes two (
/// </summary>
/// <typeparam name="T3">This is a generic ty
/// argument.</typeparam>
/// <typeparam name="T4">This is a generic ty
/// value.</typeparam>
/// <param name="argument">This is an argumer
/// the generic type argument <typeparamref r
/// <returns>The default value of the type sp
/// argument <typeparamref name="T4"/>.</retu
/// <conceptualLink target="163cae15-9020-409"
/// <conceptualLink target="073a5ae1-828f-4ba
public T4 GenericMethod<T3, T4>(T3 argument)
    return default(T4);
}
/// <summary>
/// This is an event that takes a generic arç
/// </summary>
/// <remarks>The <see cref="Delegate">delegat
/// <see cref="EventHandler{T}"/> bound to the
/// generic type argument <typeparamref name=
/// </remarks>
/// <conceptualLink target="073a5ae1-828f-4ba
public event EventHandler<T2> SomethingHapper
/// <summary>
```

```
/// This is a protected virtual method used t
/// <see cref="SomethingHappened"/> event.
/// </summary>
/// <param name="e">Arguments for the event c
/// the generic type argument <typeparamref r
/// <conceptualLink target="073a5ae1-828f-4baprotected virtual void OnSomethingHappened(T2{
    var handler = SomethingHappened;

    if(handler != null)
        handler(this, e);
}
</pre>
```

Reference

GenericMethodT3, T4 GenericClassT1, T2

Other Resources

Inline Elements typeparam

[v2014.5.31.0] Sandcastle XML Comments Guide



Miscellaneous Elements

Miscellaneous elements are typically top-level or mixed use elements that do not fit one of the other categories. Select a topic below for more information.

- AttachedEventComments
- AttachedPropertyComments
- Code Contract Elements
- conceptualLink
- exclude
- filterpriority
- include
- inheritdoc
- overloads
- token

▲ See Also

Other Resources

Usage Notes
Namespace Comments
Section Elements
Block Elements
Inline Elements



AttachedEventComments

This element is used to define the content that should appear on the auto-generated attached event member topic for a given WPF routed event member.

■ Syntax

This top-level element is valid on any routed event member. The member on which the element appears should have its own set of member-specific XML comments as well.

Include any top-level XML comments elements as you would on a standard member. These elements will be formatted in an identical fashion and will appear in the auto-generated attached event member topic.



This is a custom XML comments element implemented by the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense.

▲ Remarks

Because the attached property and attached event members of WPF classes are compiler-generated, there is no way to associate XML comments with them directly without managing a standalone XML comments file. While it is possible to do this, it is less convenient than keeping the comments in the code. The help file builder provides a solution to this through its **GenerateInheritedDocs** tool. As part of the process of generating inherited documentation, the tool will look for attached property and attached event fields. If it finds them, it will automatically inherit their comments for the related compiler-generated members as default comments to prevent a "missing comments" warning.

In addition, if it finds comments for those field members, it will check for an AttachedPropertyComments element (for attached properties) or an AttachedEventComments element (for attached events) and, if found, will use the XML comments nested within those elements for the related compiler-generated members. This allows you to provide comments for the field member and the related compiler-generated member that are entirely different but are managed from within the code.

Note

Because the attached property and event members are compiler-generated, you must fully qualify their names if you want to create a link to them with a see element as shown in the example below.

▲ Example

```
/// <summary>
/// This defines the
/// <see cref="E:XMLCommentsExamples.Documentatic
/// attached event.
/// </summary>
/// <AttachedEventComments>
/// <summary>
/// This attached event is raised when an item is
/// </summary>
/// cremarks>There's a bit more too it to get the
/// documentation example.</remarks>
/// <conceptualLink target="3563f000-5677-4cd9-a1
/// </AttachedEventComments>
```

```
/// <conceptualLink target="3563f000-5677-4cd9-af
public static readonly RoutedEvent ItemActivateEv
    "ItemActivate", RoutingStrategy.Bubble, type(
    typeof(AttachedEventsAndPropertiesTest));
/// <summary>
/// Add an event handler to an object
/// </summary>
/// <param name="o">The dependency object</param>
/// <param name="handler">The event handler</para
public static void AddItemActivateHandler(Depende
{
    ((UIElement)o).AddHandler(AttachedEventsAndPr
}
/// <summary>
/// Remove an event handler from an object
/// </summary>
/// <param name="o">The dependency object</param>
/// <param name="handler">The event handler</para
public static void RemoveItemActivateHandler(Depe
{
    ((UIElement)o).RemoveHandler(AttachedEventsAr
}
```

Reference

ItemActivate
ItemActivateEvent

Other Resources

Miscellaneous Elements AttachedPropertyComments

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedPropertyComments

This element is used to define the content that should appear on the auto-generated attached property member topic for a given WPF dependency property member.

■ Syntax

This top-level element is valid on any dependency property member. The member on which the element appears should have its own set of member-specific XML comments as well.

Include any top-level XML comments elements as you would on a standard member. These elements will be formatted in an identical fashion and will appear in the auto-generated attached property member topic.



This is a custom XML comments element implemented by the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense.

▲ Remarks

Because the attached property and attached event members of WPF classes are compiler-generated, there is no way to associate XML comments with them directly without managing a standalone XML comments file. While it is possible to do this, it is less convenient than keeping the comments in the code. The help file builder provides a solution to this through its **GenerateInheritedDocs** tool. As part of the process of generating inherited documentation, the tool will look for attached property and attached event fields. If it finds them, it will automatically inherit their comments for the related compiler-generated members as default comments to prevent a "missing comments" warning.

In addition, if it finds comments for those field members, it will check for an AttachedPropertyComments element (for attached properties) or an AttachedEventComments element (for attached events) and, if found, will use the XML comments nested within those elements for the related compiler-generated members. This allows you to provide comments for the field member and the related compiler-generated member that are entirely different but are managed from within the code.

Note

Because the attached property and event members are compiler-generated, you must fully qualify their names if you want to create a link to them with a see element as shown in the example below.

▲ Example

```
/// <summary>
/// This defines the
/// <see cref="P:XMLCommentsExamples.Documentatic"
/// attached property.
/// </summary>
/// <AttachedPropertyComments>
/// <summary>This attached property indicates whe
/// view when selected.
/// </summary>
/// <value>The default value is false</value>
/// <conceptualLink target="c0346d23-f376-4948-81"
/// <conceptualLink target="c0346d23-f376-4948-81"
```

```
public static readonly DependencyProperty IsBroug
    DependencyProperty.RegisterAttached(
    "IsBroughtIntoViewWhenSelected",
    typeof(bool),
    typeof(AttachedEventsAndPropertiesTest),
    new UIPropertyMetadata(false, OnIsBroughtInto
/// <summary>
/// Get the property value
/// </summary>
/// <param name="treeViewItem">The tree view item
/// <returns>The property value</returns>
public static bool GetIsBroughtIntoViewWhenSelect
{
    return (bool)treeViewItem.GetValue(IsBrought]
}
/// <summary>
/// Sets the property value
/// </summary>
/// <param name="treeViewItem">The tree view item
/// <param name="value">The property value</param
public static void SetIsBroughtIntoViewWhenSelect
  TreeViewItem treeViewItem, bool value)
{
    treeViewItem.SetValue(IsBroughtIntoViewWhenSe
}
/// <summary>
/// Connect or disconnect the event handler when
/// </summary>
/// <param name="dep0bj">The dependency object</p
/// <param name="e">The event arguments</param>
static void OnIsBroughtIntoViewWhenSelectedChange
  DependencyObject depObj, DependencyPropertyChar
{
    TreeViewItem item = depObj as TreeViewItem;
    if(item == null)
```

```
return;
    if(e.NewValue is bool == false)
        return;
    if((bool)e.NewValue)
        item.Selected += OnTreeViewItemSelected;
    else
        item.Selected -= OnTreeViewItemSelected;
}
/// <summary>
/// Bring the item into view
/// </summary>
/// <param name="sender">The sender of the event<
/// <param name="e">The event arguments</param>
static void OnTreeViewItemSelected(object sender,
{
    // Only react to the Selected event raised by
    // whose IsSelected property was modified. Iq
    // who are merely reporting that a descendant
    if(!Object.ReferenceEquals(sender, e.Origina)
        return;
    TreeViewItem item = e.OriginalSource as Tree\
    if(item != null)
        item.BringIntoView();
}
```

Reference

IsBroughtIntoViewWhenSelected
IsBroughtIntoViewWhenSelectedProperty

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



Code Contract Elements

The Code Contracts Library created by Microsoft contains a tool (**ccdocgen**) that can be ran after a build to insert contract XML documentation elements into the XML comments file for an assembly.

A Important

The Code Contracts Library contains an outdated patch to several Sandcastle presentation style files. Since code contracts element support is built into the latest release of the presentation styles, do not apply this patch.

Below is a list of the elements that the tool may insert into each member. Refer to the Code Contracts user manual for more information (section 8 at the time this topic was written).

Element	Description
ensures	May appear under method elements, property getters, and property setters. The element body is the string of the original postcondition.
ensuresOnThrow	May appear under method elements, property getters, and property setters. The element body is the string of the original exceptional postcondition.
invariant	May appear under classes. The element body is the string of the original invariant.
pure	May appear under methods marking them as pure. No additional information is present.
requires	May appear under method elements, property getters, and property setters. The element body is the string of the original

precondition.

▲ See Also

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



conceptualLink

This element is used to create a link to a MAML topic within the **See Also** section of a topic or an inline link to a MAML topic within one of the other XML comments elements.

■ Syntax

This element is valid on all types and type members as a top-level or an inline element.

```
<conceptualLink target="topicGUID" />
or
<conceptualLink target="topicGUID">inner text</col>
```

The target attribute specifies the GUID of the MAML topic to which the link should be connected. When used as a top-level element, a link will be created to the MAML topic in the **See Also** section. When used as an inline element within another XML comments element, it creates an inline link to the MAML topic. The self-closing form will use the topic title as the link text. Specifying inner text on the element will use that text instead for the link text.

Note

This is a custom XML comments element implemented by the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense.

▲ Example

```
/// <summary>
/// See the <conceptualLink target="db2703b4-12bc"
/// for more information.
/// </summary>
/// <remarks><c>conceptualLink</c> is classed as
/// <conceptualLink target="9341fdc8-1571-405c-86"
/// elements</conceptualLink>.</remarks>
/// <conceptualLink target="db2703b4-12bc-4cf5-86"
/// <conceptualLink target="16cdb957-a35b-4c17-bff
public void ConceptualLinkExample()
{
}</pre>
```

Reference

ConceptualLinkExample

Other Resources

Miscellaneous Elements

see

seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



exclude

This element is used to indicate that a particular type or member should be excluded from the documentation.

■ Syntax

This top-level element is valid on all types and type members.

<exclude />

▲ Remarks

This element takes precedence over all other visibility options and the type or member will not appear in the resulting help file at all.

Note

This is a custom XML comments element implemented by the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense.

This element has been deprecated in favor of using the project's **API Filter** project property to exclude members. The help file builder will translate members with this element into an API filter entry. Be aware that if used on an overloaded member, all versions of the overloaded member will be removed from the documentation due to the way the API filter works in Sandcastle.

▲ Example

/// <summary>
/// This method will not appear in the help file

```
/// </summary>
/// <exclude />
public void UndocumentedMethod()
{
}
```

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



filterpriority

This element is used by the Visual Basic editor to control the IntelliSense visibility for methods and properties. It has no effect on the appearance of the topic in the compiled help file.

■ Syntax

This top-level element can be used on methods and properties.

<filterpriority>1 | 2 | 3</filterpriority>

▲ Remarks

This element determines how a member appears in Visual Basic IntelliSense. A value of 1 means that it should appear in the **Common** tab, 2 means it should appear in the **All** tab, and 3 means it should be hidden from IntelliSense completely.

Note

This is effectively equivalent to using EditorBrowsableAttribute. However, unlike the attribute, the XML comments element only has effect in Visual Basic and is ignored by all other languages. As such, it is rarely used. The most common place to see it is in the XML comments for the base .NET Framework classes themselves.

/// <summary>
/// Filter priority example
/// </summary>

```
/// <remarks>This element is rarely used as <see
/// does the same thing and is effective across a
/// <filterpriority>1</filterpriority>
/// <conceptualLink target="0522f3bf-0a57-4d70-a2
public void FilterPriorityExample()
{
}</pre>
```

Reference

FilterPriorityExample

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



include

This element lets you import comments from another XML file that describe the types and members in your source code. The comments are imported into the XML comments file generated at build time by the compiler.

■ Syntax

This element is valid on all types and type members as a top-level or an inline element.

```
<include file="xmlFilename" path="xpath-filter-e>
```

The file attribute specifies the path to an external XML file. The filename can be a fully qualified or a relative path. If no path is specified on the filename or it is prefixed with a folder name alone, the file is assumed to be in the same folder as the Visual Studio project or a subfolder beneath it. If a relative path is given that starts with ..., the file is assumed to be in a location relative to the current source code file.

The path element specifies an XPath query used to import one or more elements into the member's XML comments. The structure of the XML file is entirely up to you. A common convention is to use elements with an ID attribute and to look them up with an XPath query such as

query such as "rootElement/subElement[@id='idValue']/*". The "/*" at the end of the query pulls in the content of the matching element.

▲ Remarks

Using this element allows you to keep common sets of XML comments or examples in a shared file as an alternative to repeatedly cutting and pasting the same elements into multiple

locations in your source code. This allows you to maintain the comment elements in a single location. When the project is built, the updated comments will automatically be imported into each of the locations where they are referenced. It also allows one person to work on the code while another writes the comments.

Note

The XPath query typically uses quote marks to surround an ID value. If you use double quotes to surround the XPath expression, use single quotes to surround the ID value. You can also use single quotes to surround the XPath expression and use double quotes to surround the ID value. Examples of both ways are shown below.

A missing XML file or an ill-formed XPath query will result in a compiler warning being generated. A valid XPath query that fails to locate anything will not generate any warning but will cause missing comments in the resulting XML comments file.

▲ Examples

```
Comments Examples
/// <include file="IncludeComments.xml" path="Con</pre>
public void IncludeAllExample()
{
    // This pulls in all comments for the method.
}
/// <summary>
/// In this example, the <c>include</c> element i
/// the comments.
/// </summary>
/// <remarks>
/// <para>As shown in these examples (see source
/// can be used on the XPath guery when nested gu
/// be consistent.</para>
/// <para>This pulls in a single paragraph elemen
///
/// <include file="IncludeComments.xml" path="Con
```

```
/// <para>This pulls in a all the content of the
///
/// <include file="IncludeComments.xml" path='Con
/// </remarks>
/// <conceptualLink target="3de64a85-dafb-4a01-85
/// <seealso cref="IncludeAllExample"/>
public void IncludeSectionsExample()
{
}
```

```
Example XML File
<?xml version="1.0" encoding="utf-8" ?>
<Comments>
  <!-- This element's content is imported by elem
  <IncludeAllExample>
    <summary>
      This summary came from an external XML file
    </summary>
    <remarks>
      These remarks came from an external XML fil
    </remarks>
    <conceptualLink target="db2703b4-12bc-4cf5-86"</pre>
    <seealso cref="IncludeSectionsExample" />
  </IncludeAllExample>
  <!-- These examples use an ID attribute to ider
  <Example id="paraExample">
    <para>This paragraph was imported from an ext
    <remarks>This element is not imported as it i
  </Example>
  <Example id="AnotherExample">
    <para>Paragraph 1</para>
    <para>Paragraph 2</para>
```



Reference

IncludeAllExample IncludeSectionsExample

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



inheritdoc

This element can help minimize the effort required to document complex APIs by allowing common documentation to be inherited from base types/members.

- Syntax
- Top-Level Inheritance Rules
- Inline Inheritance Rules
- Additional Comment File Sources and IntelliSense
- Examples
- See Also

Note

This is a custom XML comments element implemented by Sandcastle and the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense.

■ Syntax

Although typically used as a top-level element, it can be used as an inline element as well. The use of the element by itself on a member is enough to satisfy the compiler so that it will not issue warnings about missing comments on public members. Using it in conjunction with other elements allows you to inherit common information such as value and parameter descriptions while overriding the inherited documentation for other elements such as summary and remarks. Documentation can be inherited from any member from classes within your own assemblies or from members of other assemblies as well as the base .NET Framework class library. The syntax of the element is as follows:

```
<inheritdoc [cref="member"] [select="xpath-filter
</pre>
```

The optional cref attribute overrides the standard search method to allow documentation inheritance from an alternate user-specified member indicated by the *member* value.

The optional select attribute applies the specified XPath filter expression to the inherited comments. This is useful if you want to limit the inherited documentation to a specific subset of elements or just select a particular instance or set of comments. The expression can be any valid XPath query that will result in a node set.

By making use of the cref and select attributes either by themselves or together, you can fine tune the inheritance of documentation. You can also nest the element within other elements to further refine the level of inheritance.

When using the Sandcastle tools alone, the InheritDocumentationComponent must be added to the BuildAssembler configuration file to handle the inheritdoc elements. When using the Sandcastle Help File Builder, its GenerateInheritedDocumentation tool handles the task of generating the inherited documentation. The following documentation is based on the Sandcastle Help File Builder's implementation. The build component works in a similar fashion but does not support all of the scenarios related to merging and overriding existing elements.

▲ Top-Level Inheritance Rules

The inheritdoc element is valid at the root level (i.e. the same level as summary elements) on types, interfaces, virtual members, interface member implementations, and constructors. Its use on any other member type will result in no comments being inherited unless a cref attribute is specified. Note that the element is also valid in project summary and namespace summary comments as long as a cref attribute is specified to indicate from where to inherit the comments. When specified at the root level in a set of XML comments, the documentation search is performed as follows:

- If an explicit cref attribute is specified, the documentation from the specified namespace/type/member is inherited. If a cref attribute is not specified, the following rules apply.
- For types and interfaces:
 - Inherit documentation from all base classes working backwards up the inheritance chain.

 Inherit documentation from all interface implementations (if any) working through them in the order listed in the reflection information file (usually alphabetically).

For constructors:

- Search backwards up the type inheritance chain for a constructor with a matching signature.
- If a match is found, its documentation is inherited.
- For virtual members and interface implementations:
 - If the member is an override, documentation is inherited from the member it overrides.
 - If the member is part of an interface, documentation is inherited from the interface member being implemented.
- Explicit interface implementations will automatically inherit documentation from the interface member that they implement if no documentation is supplied by the user. This is done automatically because these members are by definition private and the compiler will not issue a warning if the user does not supply documentation. As such, you can omit the inheritdoc element from them unless you want to customize the comments.
- With or without an explicit cref attribute, if the inherited documentation itself contains inheritdoc elements, they will be expanded recursively working backwards up the inheritance chain.
- In all cases, if a select attribute is present, it is used to filter the inherited comments based on the specified XPath query.

When inheriting documentation at the root level, if the following elements already exist in the member's comments, the inherited versions are ignored:

- example
- exclude
- filterpriority

- preliminary
- summary
- remarks
- returns
- threadsafety
- value

The overloads element will never be inherited. This prevents the doubling of comments on the overloads page. However, you can inherit the contents of the overloads element using a select attribute with a value of "overloads/*". See below for an example.

All other elements will be inherited unless they match an element by the same name that contains a cref, href, name, vref, or xref attribute with an identical value in the member's comments. To merge comments in one of the above elements from one or more sources, use one or more nested inheritdoc elements within the given element. See below for examples.

Be aware that when param elements are inherited, the parameter's name in your class's member must match the base member's parameter name. If they do not match, you will not see any inherited documentation for the parameter. Also, if you supply comments for one parameter but omit comments for other parameters in order to inherit their documentation from a base implementation, the compiler will issue a warning. In this case, you can use a #pragma warning directive to disable the warning temporarily or add it to the project settings to disable the warning globally. See below for an example.

▲ Inline Inheritance Rules

The inheritdoc element can also be nested within other XML comments elements such as summary, remarks, example etc. in order to inherit specific parts of the documentation within those elements. When nested, the same root level inheritance rules apply and will be used to locate the first member with comments from which to inherit documentation. In addition, a filter will be automatically included based on the parent element or elements within which the inheritdoc element is nested. The cref and select attributes can also be applied to further qualify how the

documentation is inherited. If you do not want to have the parent elements automatically included in the filter, you must supply a select attribute with a rooted XPath query that specifies from where to obtain the comments (i.e. select="/summary/node()").

▲ Additional Comment File Sources and IntelliSense

In the Sandcastle Help File Builder, additional sources of inherited documentation (i.e. comments from third party class libraries) can be added to the **Documentation Sources** project node. This allows you to inherit documentation from base class libraries without having to add them as documented assemblies in your project.

Since the XML comments produced by the compiler are incomplete when using inheritdoc, it is highly recommended that you make use of the IntelliSense Build Component to produce an IntelliSense XML comments file. It will include the fully expanded set of inherited documentation so that Visual Studio can provide useful and accurate API help in the code editor and object browser.

▲ Examples

The following show various examples of using the inheritdoc element. See the comments within each for details about what the examples are showing.

```
Constructor Documentation Inheritance

/// <summary>
/// This exception class is thrown by the applic
/// unrecoverable error.
/// </summary>
/// <conceptualLink target="86453FFB-B978-4A2A-9
[Serializable]
public class CustomException : Exception
{
/// <summary>
/// Default constructor.
/// </summary>
```

```
/// <overloads>There are four overloads for
public CustomException()
}
/// <inheritdoc />
public CustomException(string message) : bas
{
    // Inherit documentation from the base E
    // this constructor's signature.
}
/// <inheritdoc />
public CustomException(string message, Excep
  base(message, innerException)
{
    // Inherit documentation from the base E
    // this constructor's signature.
}
/// <inheritdoc />
protected CustomException(SerializationInfo
  StreamingContext context) : base(info, con
{
    // Inherit documentation from the base E
    // this constructor's signature.
```

/// <summary> /// A class with an explicit interface implemen /// </summary> /// <remarks>Note that you must enable the D /// SHFB project options in order to see the ex /// <conceptualLink target="86453FFB-B978-4A2A-public class ExplicitImplementation : ICollecti

```
#region ICollection Members
void ICollection.CopyTo(Array array, int ir
    // Comments are automatically inherite
    // interface members with no comments.
}
int ICollection.Count
    get
        // Comments are automatically inher
        // interface members with no commer
        return 0;
    }
}
bool ICollection.IsSynchronized
    get
    {
        // Comments are automatically inher
        // interface members with no commer
        return true;
    }
}
/// <inheritdoc />
/// <remarks>This is a dummy class and alwa
object ICollection.SyncRoot
    get
        // In this case, we inherit the <su
        // comments and add a <remarks> con
        // comments, we need to specify the
```

```
return null;
    }
}
#endregion
#region IEnumerable Members
/// <inheritdoc />
/// <returns>This is a dummy class so it the
IEnumerator IEnumerable.GetEnumerator()
{
    // In this case, we automatically inher
    // <summary> but override the <returns>
    // because we specified comments, we ha
    // tag too.
    throw new Exception("The method or open
}
#endregion
#region ICloneable Members
/// <inheritdoc />
public object Clone()
{
    // Not explicitly implemented so we hav
    // documentation on this one.
    return null;
#endregion
```



```
/// <summary>
/// A base class from which to inherit document
/// </summary>
/// <remarks>
/// <para>These remarks are for the base class.
///
/// <para>This information applies to all class
/// <see cref="BaseInheritDoc"/>:
/// <list type="bullet">
/// <item><description>Point #1.</description><
/// <item><description>Point #2.</description><
/// <item><description>Point #3.</description><
/// </list>
/// </para>
/// </remarks>
/// <conceptualLink target="86453FFB-B978-4A2A-
public class BaseInheritDoc
{
    /// <summary>
    /// Constructor
    /// </summary>
    public BaseInheritDoc()
    {
    }
    /// <summary>
    /// The ToString implementation for BaseInl
    /// </summary>
    /// <returns>A string representing the obje
    public override string ToString()
    {
        return base.ToString();
    }
    /// <summary>
    /// Summary for the method with an example
    /// </summary>
    /// <returns>True all the time</returns>
```

```
/// <example>
/// This example is from the base class
/// <code>
/// // 'x' is always true
/// bool x = instance.MethodWithExample();
/// </code>
/// </example>
public virtual bool MethodWithExample()
{
    return true;
}
/// <summary>
/// The method in the base class has lots (
/// </summary>
/// <remarks>Remarks for the base class</re
/// <param name="x">The parameter</param>
/// <exception cref="ArgumentException">Thi
/// <exception cref="ArgumentOutOfRangeExce
/// less than zero.</exception>
/// <example>
/// <code>
/// /// Example goes here
/// </code>
/// </example>
/// <seealso cref="ToString" />
/// <seealso cref="MethodWithExample"/>
public virtual void MethodWithLotsOfComment
/// <summary>
/// A method with two examples
/// </summary>
/// <example>
/// <span id="Example 1">
/// This is example #1:
/// <code>
/// // Example #1
```

```
/// </code>
     /// </span>
     /// <span id="Example 2">
     /// This is example #2:
     /// <code>
     /// // Example #2
     /// </code>
     /// </span>
     /// </example>
     protected virtual void MethodWithTwoExample
         // By using a <span> with an ID, we car
         // selection by an override in a derive
 #endregion
- #region Derived class
 /// <summary>
 /// This is a derived class with inherited docu
 /// </summary>
 /// <remarks>This will inherit just the last &I
 /// the base class's <remarks&gt; tag:
 /// <inheritdoc select="para[last()]" />
 /// </remarks>
 /// <conceptualLink target="86453FFB-B978-4A2A-
 public class DerivedClassWithInheritedDocs : Ba
 {
     // Note in the <remarks> tag above that we
     // parts of a comment tag's text by using a
     // can allow you to merge comments from var
     // set of comments in a given tag. An impl
     // the selection to the <remarks> tag is ac
     // the select attribute were omitted, the {
     // from the base class would be inherited.
     /// <inheritdoc cref="Object.ToString" />
```

```
public override string ToString()
    // This override ignores the base class
    // cref attribute to obtain the comment
    // System.Object.ToString instead.
    return base.ToString();
}
/// <summary>
/// This overloaded method does something
/// </summary>
/// <param name="p1">The string parameter
/// <overloads>
/// <summary>There are three overloads for
/// <remarks>These remarks are from the ove
/// first version.</remarks>
/// </overloads>
public void OverloadedMethod(string p1)
{
}
#pragma warning disable 1573
/// <inheritdoc cref="OverloadedMethod(str:</pre>
/// <param name="p2">The second string para
public void OverloadedMethod(string p1, str
{
    // Inherit documentation from the first
    // comments for the second parameter.
    // Note that because we supplied commer
    // but not the other, the compiler will
    // we can shut it up by using a "#pragr
    // shown.
}
/// <inheritdoc cref="OverloadedMethod(str:</pre>
        select="param|overloads/*" />
/// <param name="x">An integer parameter</pr
public void OverloadedMethod(string p1, int
```

```
{
    // This example inherits the comments 1
    // the first version, the content of the
    // first version, and adds comments for
#pragma warning restore 1573
/// <summary>
/// An override of the method with an examp
/// </summary>
/// <returns>Always returns false</returns>
/// <example>
/// <inheritdoc />
/// This example applies to the derived
/// <code>
/// if(derivedInstance.MethodWithExample())
        Console.WriteLine("This is never re
/// </code>
/// </example>
public override bool MethodWithExample()
    // The <example> tag inherits the example
    // and adds a new example of its own.
    // limits the nested tag to inheriting
    // <example> tag in the base class's co
    return false;
}
/// <inheritdoc select="summary|remarks|par
public override void MethodWithLotsOfCommer
{
    // For this override, we don't want all
    // from the <summary>, <remarks>, and <
}
/// <summary>
/// This only includes one of the examples
/// </summary>
```

```
/// <example>
    /// <inheritdoc select="span[@id='Example 2
    /// </example>
    protected override void MethodWithTwoExamp.
    {
        // Here, we use a filter to select a gr
        // a <span> tag from the base member's
    }
    /// <summary>
    /// This uses a shared example from a base
    /// public and this doesn't override.
    /// </summary>
    /// <example>
    /// <inheritdoc cref="MethodWithTwoExamples</pre>
            select="span[@id='Example 2']" />
    /// </example>
    public void MethodUsingSharedExample()
    {
        // This method uses a cref attribute ar
        // a specific example from a member to
    }
#endregion
```

Reference

BaseInheritDoc

CustomException

DerivedClassWithInheritedDocs

ExplicitImplementation

SetDocumentation

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



overloads

This element is used to define the content that should appear on the auto-generated overloads topic for a given set of member overloads.

■ Syntax

This top-level element is valid on any overloaded member. However, it should only appear on one of the members in each overload set. The member on which the element appears should have its own set of member-specific XML comments as well.

In its simplest form, the element contains a description that will appear as the summary in the auto-generated overloads member topic. The expanded form allows you to include any top-level XML comments elements as you would on a standard member. These elements will be formatted in an identical fashion and will appear in the auto-generated overloads member topic.

▲ Example

```
// Simple overloads form
/// <summary>
/// This is used to sum an enumerable list of val
/// </summary>
/// <param name="values">The values to sum</param
/// <returns>The sum of the values</returns>
/// <overloads>There are two overloads for this n
/// <conceptualLink target="5b11b235-2b6c-4dfc-86"
public int SumValues(IEnumerable<int> values)
{
    return values.Sum(v => v);
}
/// <summary>
/// This is used to sum two enumerable list of va
/// </summary>
/// <param name="firstValues">The first set of values">The first set of values
/// <param name="secondValues">The second set of
/// <returns>The sum of the values from both enum
/// <conceptualLink target="5b11b235-2b6c-4dfc-86"
public int SumValues(IEnumerable<int> firstValues
{
    return firstValues.Sum(v => v) + secondValues
}
// Expanded overloads form
/// <summary>
/// This is used to average an enumerable list of
/// </summary>
/// <param name="values">The values to average</pr
/// <returns>The average of the values</returns>
/// <overloads>
/// <summary>
```

```
/// These methods are used to compute the average
/// </summary>
/// <remarks>
/// These methods serve no other purpose than to
/// <c>overloads</c> XML comments element.
/// </remarks>
/// <example>
/// <code language="cs">
/// SampleClass sc = new SampleClass(0);
///
/// Console.WriteLine("Average: {0}", sc.Average(
/// Console.WriteLine("Average: {0}", sc.Average(
        new[] { 10, 20, 30, 40}));
/// </code>
/// </example>
/// </overloads>
/// <conceptualLink target="5b11b235-2b6c-4dfc-86"
public double AverageValues(IEnumerable<double> \)
{
    return values.Average(v => v);
}
/// <summary>
/// This is used to get the average of two enumer
/// </summary>
/// <param name="firstValues">The first set of values">The first set of values
/// <param name="secondValues">The second set of
/// <returns>The average of the values from both
/// <conceptualLink target="5b11b235-2b6c-4dfc-86"
public double AverageValues(IEnumerable<double> 1
  IEnumerable<double> secondValues)
{
    return firstValues.Concat(secondValues).Avera
}
```

Reference

SumValues AverageValues

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



token

This element represents a replaceable tag within a topic. 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.

■ Syntax

This element is valid on all types and type members as a top-level or an inline element.

<token>tokenName</token>

The tokenName inner text specifies the ID of the token to insert into the topic.

Note

This is a custom XML comments element implemented by the Sandcastle Help File Builder. It will not appear in the list of valid elements for XML comments IntelliSense. The API Token Resolution build component must be added to the project's Component Configurations property in order for tokens in XML comments to be replaced. Only general token values can be used when referenced from XML comments. MAML elements will not be resolved and rendered.

Those using the Sandcastle tools by themselves can achieve the same results by adding the SharedContentComponent to their reference build configuration file for **BuildAssembler**. The configuration will match the settings from the copy used for conceptual builds.

See the Sandcastle Help File Builder's help file for more information on the component and on token files in general.

▲ Example

```
/// <summary>
/// The help file was built on <token>BuildDate
/// <token>SandcastleVersion</token>.
/// </summary>
/// <remarks>For tokens to be resolved, the <b>AF
/// must be added to the SHFB project.</remarks>
/// <conceptualLink target="8c9273f3-0000-43cd-bk
public void TokenExample()
{
}</pre>
```

▲ See Also

Reference

TokenExample

Other Resources

Miscellaneous Elements

[v2014.5.31.0] Sandcastle XML Comments Guide



Frequently Asked Questions (FAQ)

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

None to date.

▲ See Also

Other Resources

Welcome
Namespace Comments
Section Elements
Block Elements
Inline Elements
Miscellaneous Elements
Links to Resources

[v2014.5.31.0] Sandcastle XML Comments Guide



Links to Resources

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

- Information and Guides
- Discussion Groups, Blogs, Etc.
- Build Tools

▲ Information and Guides

The most recent version of this help file and other guides on MAML and XML comments can be download from the Sandcastle Help File Builder project on GitHub (currently the v2015.10.10.0 release as of October 10, 2015).

→ Discussion Groups, Blogs, Etc.

- To discuss the help file builder and tools, suggest new features, report bugs in them, and request help using them, visit the Sandcastle Help File Builder project site.
- For discussion and questions about help systems in general, visit the Visual Studio General Questions forum.
- The retired Microsoft Developer Documentation and Help System forum may also provide information on older help systems.

■ Build Tools

- Microsoft .NET Frameworks and SDKs.
- The HTML Help Workshop for building HTML Help 1.x (.CHM) help files.

▲ See Also

Other Resources Welcome

[v2014.5.31.0] Sandcastle XML Comments Guide



Making a Donation

If you would like to support this project, 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!

[v2014.5.31.0] Sandcastle XML Comments Guide



Version History

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

■ Version History

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

- Version 2015.10.10.0
- Version 2014.11.22.0
- Version 2014.5.31.0
- Version 2014.1.26.0
- Version 1.0.0.0

[v2014.5.31.0] Sandcastle XML Comments Guide



Version 2015.10.10.0

Release notes for version 2015.10.9.0.

△ Changes in This Release

- Merged changes from Rob Siklos about support for the virtual and abstract language keywords on the see XML comments element.
- Updated the see and seealso topics to note support for the autoUpgrade and qualifyHint attributes. Based on a suggestion from Stoffelche.

▲ See Also

Other Resources

Version History

[v2014.5.31.0] Sandcastle XML Comments Guide



Version 2014.11.22.0

Release notes for version 2014.11.22.0

△ Changes in This Release

 Added information on creating and using a NamespaceGroupDoc class to add comments for name space group topics when namespace grouping is enabled in the Sandcastle Help File Builder project.

▲ See Also

Other Resources Version History

[v2014.5.31.0] Sandcastle XML Comments Guide



Version 2014.5.31.0

Release notes for version 2014.5.31.0

△ Changes in This Release

 Added a new topic for the section element revisionHistory contributed by THLucas.

▲ See Also

Other Resources Version History

[v2014.5.31.0] Sandcastle XML Comments Guide



Version 2014.1.26.0

Release notes for version 2014.1.26.0

△ Changes in This Release

- Switched to the new date-based version numbering scheme.
- Added information on the member ID prefix codes from the .NET Documentation Guidelines document.
- Added brief information on the code contracts elements from the Code Contracts User Manual.
- Added async, Async, await, Await, let!, async/await, Async/Await, and async/let! as possible langword attribute values on the see element.

▲ See Also

Other Resources

Version History

[v2014.5.31.0] Sandcastle XML Comments Guide



Version 1.0.0.0

Version 1.0.0.0 was released on December 16th, 2012.

△ Changes in This Release

• Initial release.

▲ See Also

Other Resources Version History

[v2014.5.31.0] Sandcastle XML Comments Guide



Glossary

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

В

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

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: MAML

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.

M

MAML

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

See Also: conceptual content

R

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 originally 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 Sandcastle Help File Builder 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 2013 or later.

See Also: Sandcastle

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.

[v2014.5.31.0] Sandcastle XML Comments Guide



XMLCommentsExamples Namespace

This namespace contains classes used to demonstrate the various XML comments elements.

	Class	Description
4 3	GenericClassT1, T2	This class is used to demonstrate the various XML comments elements related to generics. It serves no useful purpose.
4 ;	RevisionHistoryClass	This class demonstrates the revision history
4 \$	SampleClass	This class is used to demonstrate the various XML comments elements. It serves no useful purpose.
9 \$	ThreadSafetyClass	This class demonstrates the threadsafety XML comments element.

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClassT1, T2 Class

This class is used to demonstrate the various XML comments elements related to generics. It serves no useful purpose.

■ Inheritance Hierarchy

SystemObject XMLCommentsExamplesGenericClassT1, T2

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public class GenericClass<T1, T2>
where T2 : EventArgs
```

Type Parameters

T1

This is the first generic argument.

T2

This is the second generic argument constrained to be or derive from EventArgs.

The GenericClassT1, T2 type exposes the following members.

→ Constructors

Name Description

Ge	nericClassT1,	Initializes a new instance of the
T2		GenericClassT1, T2 class

Top

▲ Properties

Name	Description
Property	This is a property that gets or sets an instance of the type specified by the generic type argument $T1$.

Top

Methods

	Name	Description
≅ 🚱	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
Ģ 	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≘	GenericMethodT3, T4	This is a generic method that takes two other generic types
≡	GetHashCode	Serves as the default

		hash function. (Inherited from Object.)
≅©	GetType	Gets the Type of the current instance. (Inherited from Object.)
ġ ®	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
= Q	Method	This is a method with an argument.
ēj ©	OnSomethingHappened	This is a protected virtual method used to raise the SomethingHappened event.
⊒	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

■ Events

	Name	Description
4	SomethingHappened	This is an event that takes a generic argument.

Top

▲ See Also

Reference XMLCommentsExamples Namespace Other Resources typeparam

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2* Constructor

Initializes a new instance of the Generic Class T1, T2 class

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public GenericClass()
```

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2* Properties

The GenericClassT1, T2 generic type exposes the following members.

▲ Properties

	Name	Description
É	Property	This is a property that gets or sets an instance of the type specified by the generic type argument <i>T1</i> .

Top

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2*Property Property

This is a property that gets or sets an instance of the type specified by the generic type argument T1.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public T1 Property { get; set; }
```

Property Value

Type: *T1*

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

Other Resources

typeparamref

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2* Methods

The GenericClassT1, T2 generic type exposes the following members.

Methods

specified object is ed to the current object. (Inherited from Object (Inherited from Object to to free resources and perform other cleans operations before it is reclaimed by garbagy collection. (Inherited from Object (Inherited	Name	Na	ne	Description
Thatize to free resources and perform other cleanul operations before it is reclaimed by garbagy collection. (Inherited from Object to the collection operations before it is reclaimed by garbagy collection. (Inherited from Object to the collection operations before it is reclaimed by garbagy collection. This is a generic method that takes two other generic types GetHashCode Serves as the default hash function.	Equals	Eq	als	Determines whether the specified object is equal to the current object. (Inherited from Object.)
Generic Method 13, 14 This is a generic method that takes two ther generic types GetHashCode Serves as the defaul hash function.	Finalize	Fin	alize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
hash function.	GenericMe	Ge	nericMethodT3, T4	method that takes two
	GetHashC	Ge	HashCode	Serves as the default hash function. (Inherited from Object.)
current instance.	GetType	Ge	Туре	Gets the Type of the current instance. (Inherited from Object.)

Ģ	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
≓	Method	This is a method with an argument.
Ģ	OnSomethingHappened	This is a protected virtual method used to raise the SomethingHappened event.
≡	ToString	Returns a string that represents the current object. (Inherited from Object.)

Top

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2*GenericMethod*T3*, *T4* Method

This is a generic method that takes two other generic types

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

argument

Type: *T3*

This is an argument of the type specified by the generic type argument T3.

Type Parameters

T3

This is a generic type argument for the method argument.

T4

This is a generic type argument for the return value.

Return Value

Type: *T4*

The default value of the type specified by the generic type argument *T4*.

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

Other Resources

typeparam typeparamref

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2*Method Method

This is a method with an argument.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void Method(
    T1 argument
)
```

Parameters

argument

Type: *T1*

This is an argument of the type specified by the generic type argument T1.

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

Other Resources

typeparamref

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2*OnSomethingHappened Method

This is a protected virtual method used to raise the SomethingHappened event.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

е

Type: *T2*

Arguments for the event of the type specified by the generic type argument T2.

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

Other Resources

typeparamref

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2* Events

The GenericClassT1, T2 generic type exposes the following members.

▲ Events

	Name	Description
4	SomethingHappened	This is an event that takes a generic argument.

Top

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



GenericClass*T1*, *T2*SomethingHappened Event

This is an event that takes a generic argument.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

C# VB C++ F#

public event EventHandler<T2> SomethingHappened

Value

Type: SystemEventHandlerT2

▲ Remarks

The delegate for this event is EventHandlerTEventArgs bound to the type specified by the generic type argument *T2*.

▲ See Also

Reference

GenericClassT1, T2 Class XMLCommentsExamples Namespace

Other Resources

typeparamref



RevisionHistoryClass Class

This class demonstrates the revision history

▲ Inheritance Hierarchy

SystemObject XMLCommentsExamplesRevisionHistoryClass

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

C#	VB	C++	F#	Сору
pub]	lic	class	RevisionHistoryClass	

The RevisionHistoryClass type exposes the following members.

→ Constructors

	Name	Description
=	RevisionHistoryClass	Initializes a new instance of the RevisionHistoryClass class

•

Top

→ Properties

Name	Description	
Haine	Description	



Revision

This property has revision history

Top

Methods

	Name	Description
≟	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
₫	ExampleMethod	This method has revision history
ġ Ŷ	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
₫	GetHashCode	Serves as the default hash function. (Inherited from Object.)
∃	GetType	Gets the Type of the current instance. (Inherited from Object.)
ÿ ©	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
≟	ToString	Returns a string that represents the current object.

(Inherited from Object.)

Top

▲ Remarks

The revision history element can be used on the class and its members.

▲ Revision History

Date	Version	Description
04/11/2014	9.0.1.0	This revision will be displayed since the "visible" attribute is not present (i.e. default).
04/12/2014	9.0.1.1	This revision will be displayed due to it's "visible=true" attribute setting.
04/20/2014	9.0.1.3	This revision will be displayed since the "visible" attribute is not present (i.e. default). Other XML comments elements are supported here and can be used as in a remarks section to add additional details. • Point #1 • Point #2 • Point #3

▲ See Also

Reference

XMLCommentsExamples Namespace

Other Resources

revisionHistory

[v2014.5.31.0] Sandcastle XML Comments Guide



RevisionHistoryClass Constructor

Initializes a new instance of the RevisionHistoryClass class

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public RevisionHistoryClass()
```

▲ See Also

Reference

RevisionHistoryClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



RevisionHistoryClass Properties

The RevisionHistoryClass type exposes the following members.

▲ Properties

Name	Description
Revision	This property has revision history

Top

▲ See Also

Reference

RevisionHistoryClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



RevisionHistoryClassRevision Property

This property has revision history

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public string Revision { get; set; }
```

Property Value

Type: String

▲ Revision History

Date	Version	Description
05/25/2014	9.0.1.4	Added an example property

▲ See Also

Reference

RevisionHistoryClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



RevisionHistoryClass Methods

The RevisionHistoryClass type exposes the following members.

Methods

	Name	Description
≘ ∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
⊒	ExampleMethod	This method has revision history
ĕ	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≡	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡ ∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
ÿ ©	MemberwiseClone	Creates a shallow copy of the

		current Object. (Inherited from Object.)
∃	ToString	Returns a string that represents the current object. (Inherited from Object.)

Top

▲ See Also

Reference

RevisionHistoryClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



RevisionHistoryClassExample Method

This method has revision history

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#
public void ExampleMethod()
```

▲ Revision History

Date	Version	Description
05/25/2014	9.0.1.4	Added an example method

▲ See Also

Reference

RevisionHistoryClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClass Class

This class is used to demonstrate the various XML comments elements. It serves no useful purpose.

▲ Inheritance Hierarchy

SystemObject XMLCommentsExamplesSampleClass

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

C#	VB	C++	F#	Copy _
pub	lic	class	SampleClass	

The SampleClass type exposes the following members.

→ Properties

Name	Description
SampleNumber	This is used to get or set the sample number

Top

Methods

Name	Description

∍ ù F	AverageValues(IEnumerableDouble)	This is used to average an enumerable list of values
≡	AverageValues(IEnumerableDouble, IEnumerableDouble)	This is used to get the average of two enumerable list of values
⊒ ♠	CallStoredProcedure	Executes a SqlCommand with the specified storedProcName as a stored procedure initialized for updating the values of the specified row.
∃	ConceptualLinkExample	See the conceptualLink topic for more information
=	DoSomething	This version of the method takes no parameters.
≅	DoSomething(Int32)	This version of the method takes an integer parameter.
≅©	DoSomething(String)	This version of the method takes a string parameter.
=	DoSomething(Int32, String)	This version of the method takes both

		an integer and a string parameter.
≓	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≅ ©	FilterPriorityExample	Filter priority example
ĕ û	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≡	GetHashCode	Serves as the defau hash function. (Inherited from Object.)
· F	GetRandomNumber	This returns a random number
ΞΦ	GetType	Gets the Type of the current instance. (Inherited from Object.)
≅ ©	IncludeAllExample	This summary came from an external

		XML file.
⊒₩	IncludeSectionsExample	In this example, the include element is used to pull in sections of the comments.
≟	Increment	The Increment method increments the stored number by one
ē	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
≟	MethodRequiringSpecificPermissions	The permission element is used to document the types of permissions that a caller must be granted to access this method.
ē	OnSomethingHappened	This is a protected virtual method used to raise the SomethingHappene event.
≅ ©	ParagraphExample	A simple demonstration of the para element.
=0	PerformAnAction	This performs an

		action
ΞΦ	PreliminaryExample	This method may go away or its signature may change in a later release.
≡	ProcessText	This method processes text
ΞΦ	SeeElementExternalExample	This is used to demonstrate the external reference type of see and seealso elements.
≡ ¢ S	SeeLangWordExamples	This demonstrates the see element's langword form
ΞΦ	SummaryRemarksExample	The summary is a brief description of the type or type member and will be displayed in IntelliSense and the Object Browser.
≘ ∳	SumValues(IEnumerableInt32)	This is used to sum an enumerable list c values
ΞΦ	SumValues(IEnumerableInt32, IEnumerableInt32)	This is used to sum two enumerable list of values
≟	TemporaryMethod	A temporary methoc

	TokenExample	The help file was built on October 10, 2015 using Sandcastle v2015.10.10.0.
⊒	ToString	Returns a string that represents the current object. (Inherited from Object.)
≡ ₩	VariousListExamples	This method shows various examples of the list XML comments element.
≡	VariousNoteExamples	This shows the resu of the various note types.

Тор

■ Events

	Name	Description
9	SomethingHappened	This event is raised when something interesting happens.

Top

▲ Operators

	Name	Description
<u>(/-</u> =+) S	Addition	Addition operator overload

Top

▲ Thread Safety

Any public **static** (**Shared** in Visual Basic) members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

▲ See Also

Reference

XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClass Properties

The SampleClass type exposes the following members.

▲ Properties

	Name	Description
E	SampleNumber	This is used to get or set the sample number

Top

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSampleNumber Property

This is used to get or set the sample number

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public int SampleNumber { get; set; }
```

Property Value

Type: Int32

The value can be any valid integer

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

value

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClass Methods

The SampleClass type exposes the following members.

Methods

	Name	Description
:∳ ₹	AverageValues(IEnumerableDouble)	This is used to average an enumerable list of values
≓	AverageValues(IEnumerableDouble, IEnumerableDouble)	This is used to get the average of two enumerable list of values
≡ ♠	CallStoredProcedure	Executes a SqlCommand with the specified storedProcName as a stored procedure initialized for updating the values of the specified row.
≓ ₩	ConceptualLinkExample	See the conceptualLink topic for more information
=0	DoSomething	This version of the

		method takes no parameters.
≡©	DoSomething(Int32)	This version of the method takes an integer parameter.
≡©	DoSomething(String)	This version of the method takes a string parameter.
∃	DoSomething(Int32, String)	This version of the method takes both an integer and a string parameter.
≡ ♠	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≡	FilterPriorityExample	Filter priority example
·	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≡	GetHashCode	Serves as the defau hash function.

		(Inherited from Object.)
e ù F	GetRandomNumber	This returns a random number
∃	GetType	Gets the Type of the current instance. (Inherited from Object.)
ΞΦ	IncludeAllExample	This summary came from an external XML file.
≓	IncludeSectionsExample	In this example, the include element is used to pull in sections of the comments.
Ξ ῷ	Increment	The Increment method increments the stored number by one
<u>~</u>	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
∄	MethodRequiringSpecificPermissions	The permission element is used to document the types of permissions that a caller must be granted to access

		this method.
ġ ©	OnSomethingHappened	This is a protected virtual method used to raise the SomethingHappene event.
≡	ParagraphExample	A simple demonstration of the para element.
=0	PerformAnAction	This performs an action
∃	PreliminaryExample	This method may go away or its signature may change in a later release.
έ∳	ProcessText	This method processes text
≟	SeeElementExternalExample	This is used to demonstrate the external reference type of see and seealso elements.
∉∳ S	SeeLangWordExamples	This demonstrates the see element's langword form
≅ ∳	SummaryRemarksExample	The summary is a brief description of the type or type member and will be displayed in

		IntelliSense and the Object Browser.
∃	SumValues(IEnumerableInt32)	This is used to sum an enumerable list c values
≡	SumValues(IEnumerableInt32, IEnumerableInt32)	This is used to sum two enumerable list of values
≓©	TemporaryMethod	A temporary methoc
∃ 🗳	TokenExample	The help file was built on October 10, 2015 using Sandcastle v2015.10.10.0.
∃	ToString	Returns a string that represents the current object. (Inherited from Object.)
=	VariousListExamples	This method shows various examples of the list XML comments element.
≅©	VariousNoteExamples	This shows the resu of the various note types.

Top

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassAverageValues Method

These methods are used to compute the average of enumerable lists of integers

■ Overload List

	Name	Description
F	AverageValues(IEnumerableDouble)	This is used to average an enumerable list of values
≓ ♠	AverageValues(IEnumerableDouble, IEnumerableDouble)	This is used to get the average of two enumerable list of values

Top

▲ Remarks

These methods serve no other purpose than to demonstrate the use of the overloads XML comments element.

▲ Examples

C#

```
SampleClass sc = new SampleClass(0);

Console.WriteLine("Average: {0}", sc.Average(new|
Console.WriteLine("Average: {0}", sc.Average(new|
new[] { 10, 20, 30, 40}));
```

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassAverageValues Method (IEnumerableDouble)

This is used to average an enumerable list of values

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

values

Type: System.Collections.GenericlEnumerableDouble

The values to average

Return Value

Type: Double

The average of the values

▲ See Also

Reference

SampleClass Class AverageValues Overload

XMLCommentsExamples Namespace Other Resources overloads

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassAverageValues Method (IEnumerableDouble, IEnumerableDouble)

This is used to get the average of two enumerable list of values

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

firstValues

Type: System.Collections.GenericIEnumerableDouble

The first set of values to average

secondValues

Type: System.Collections.GenericlEnumerableDouble

The second set of values to average

Return Value

Type: Double

The average of the values from both enumerable lists

▲ See Also

Reference

SampleClass Class AverageValues Overload XMLCommentsExamples Namespace

Other Resources overloads

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassCallStoredProce Method

Executes a SqlCommand with the specified *storedProcName* as a stored procedure initialized for updating the values of the specified *row*.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

storedProcName

Type: SystemString

The stored procedure name to execute

row

Type: SystemInt32
The row to update

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources param paramref

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassConceptualLinkE Method

See the conceptualLink topic for more information.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public void ConceptualLinkExample()
```

▲ Remarks

conceptualLink is classed as one of the miscellaneous elements.

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources

conceptualLink

See Also information

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassDoSomething Method

This method has four overloads

■ Overload List

	Name	Description
≅∳	DoSomething	This version of the method takes no parameters.
≅∳	DoSomething(Int32)	This version of the method takes an integer parameter.
≟∳	DoSomething(String)	This version of the method takes a string parameter.
≟∳	DoSomething(Int32, String)	This version of the method takes both an integer and a string parameter.

Top

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassDoSomething Method

This version of the method takes no parameters.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void DoSomething()
```

▲ Remarks

To see all overloads, refer to the DoSomething topic.

Other overloads:

- DoSomething(Int32)
- DoSomething(String)
- DoSomething(Int32, String)

Alternate syntax to link to the overloads topic: DoSomething Overloads

▲ See Also

Reference

SampleClass Class DoSomething Overload XMLCommentsExamples Namespace

Other Resources

see seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassDoSomething Method (Int32)

This version of the method takes an integer parameter.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void DoSomething(
    int number
)
```

Parameters

number

Type: SystemInt32 A number to use

▲ See Also

Reference

SampleClass Class

DoSomething Overload

XMLCommentsExamples Namespace

SampleClassDoSomething

SampleClassDoSomething

SampleClassDoSomething(String)

SampleClassDoSomething(Int32, String)

Other Resources

see seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassDoSomething Method (String)

This version of the method takes a string parameter.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void DoSomething(
          string text
)
```

Parameters

text

Type: SystemString A text value to use

▲ See Also

Reference

SampleClass Class

DoSomething Overload

XMLCommentsExamples Namespace

SampleClassDoSomething

SampleClassDoSomething(Int32)

SampleClassDoSomething(Int32, String)

Other Resources

see seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassDoSomething Method (Int32, String)

This version of the method takes both an integer and a string parameter.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void DoSomething(
    int number,
    string text
)
```

Parameters

```
number
Type: SystemInt32
A number to use
text
Type: SystemString
```

A text value to use

▲ See Also

Reference

SampleClass Class
DoSomething Overload
XMLCommentsExamples Namespace

SampleClassDoSomething
SampleClassDoSomething(Int32)
SampleClassDoSomething(String)

Other Resources

see seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassFilterPriorityExar Method

Filter priority example

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#
public void FilterPriorityExample()
```

▲ Remarks

This element is rarely used as EditorBrowsableAttribute does the same thing and is effective across all languages.

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

filterpriority

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassGetRandomNum Method

This returns a random number

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public int GetRandomNumber()
```

Return Value

Type: Int32

A random number using SampleNumber as the seed.

▲ Examples

The following example demonstrates the use of this method.

```
C# VB

// Get a new random number
SampleClass sc = new SampleClass(10);

int random = sc.GetRandomNumber();

Console.WriteLine("Random value: {0}", random);
```

▲ See Also

Reference

example

SampleClass Class XMLCommentsExamples Namespace

Other Resources code

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassIncludeAllExamp Method

This summary came from an external XML file.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void IncludeAllExample()
```

▲ Remarks

These remarks came from an external XML file.

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

SampleClassIncludeSectionsExample

Other Resources

conceptualLink

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassIncludeSectionsE Method

In this example, the include element is used to pull in sections of the comments.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#
public void IncludeSectionsExample()
```

▲ Remarks

As shown in these examples (see source code). Single and double quotes can be used on the XPath query when nested quotes are required. The key is to be consistent.

This pulls in a single paragraph element.

This paragraph was imported from an external XML file.

This pulls in a all the content of the element.

Paragraph 1

Paragraph 2

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

SampleClassIncludeAllExample Other Resources include

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassIncrement Method

The Increment method increments the stored number by one

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public int Increment()
```

Return Value

Type: Int32

The new sample number after being incremented

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

C

returns

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassMethodRequirinç Method

The permission element is used to document the types of permissions that a caller must be granted to access this method.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

▲ Syntax

```
C# VB C++ F#

public void MethodRequiringSpecificPermissions()
```

▲ .NET Framework Security

Permission	Description
System.Security.PermissionsSecurityPermission	Execution privilege.
System.Security.PermissionsReflectionPermission	Member access privilege for reflection.

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources permission

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassOnSomethingHa Method

This is a protected virtual method used to raise the SomethingHappened event.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

protected virtual void OnSomethingHappened()
```

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace SampleClassSomethingHappened

Other Resources

see seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassParagraphExam_| Method

A simple demonstration of the para element.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void ParagraphExample()
```

▲ Remarks

The para element on the first block of text is optional. Subsequent blocks of text must be wrapped in it to create new paragraphs.

Paragraph 1.

Paragraph 2.

Self-closing and empty paragraphs are ignored unless they create an initial paragraph break.

Paragraph 4.

HTML paragraph elements can also be used.

They work the same way.

▲ See Also

Reference SampleClass Class XMLCommentsExamples Namespace Other Resources para

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassPerformAnAction Method

This performs an action

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void PerformAnAction()
```

▲ Events

Event Type	Reason
SampleClassSomethingHappened	This event is raised to let the user know something happened.

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

event

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassPreliminaryExam Method

This API is preliminary and subject to change.

This method may go away or its signature may change in a later release.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void PreliminaryExample()
```

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources preliminary

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassProcessText Method

This method processes text

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void ProcessText(
         string text
)
```

Parameters

text

Type: SystemString
The text to process

▲ Exceptions

Exception	Condition
ArgumentNullException	This is thrown if the <i>text</i> parameter is null.
ArgumentException	This is thrown if the <i>text</i> parameter is an empty string.
InvalidOperationException	This is thrown because the method

is not currently implemented.

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources exception

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSeeElementExte Method

This is used to demonstrate the external reference type of see and seealso elements.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void SeeElementExternalExample()
```

▲ Remarks

- Basic link: https://GitHub.com/EWSoftware/SHFB
- Link with target: https://GitHub.com/EWSoftware/SHFB/wiki
- Link with alternate text: https://GitHub.com/EWSoftware/issues
- Link with inner text: SHFB on GitHub

Equivalent seealso links are shown below.

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources

https://GitHub.com/EWSoftware/SHFB https://GitHub.com/EWSoftware/SHFB/wiki https://GitHub.com/EWSoftware/SHFB/issues SHFB on GitHub see seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSeeLangWordE> Method

This demonstrates the see element's langword form

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#
public static string SeeLangWordExamples()
```

Return Value

Type: String

This method is and always returns .

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

see

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSummaryRemar Method

The summary is a brief description of the type or type member and will be displayed in IntelliSense and the Object Browser.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void SummaryRemarksExample()
```

▲ Remarks

The remarks element should be used to provide more detailed information about the type or member such as how it is used, its processing, etc.

Remarks will only appear in the help file and can be as long as necessary.

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources

summary remarks [v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSumValues Method

There are two overloads for this method

■ Overload List

	Name	Description
≅ ©	SumValues(IEnumerableInt32)	This is used to sum an enumerable list of values
≓	SumValues(IEnumerableInt32, IEnumerableInt32)	This is used to sum two enumerable list of values

Top

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSumValues Method (IEnumerableInt32)

This is used to sum an enumerable list of values

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

values

Type: System.Collections.GenericIEnumerableInt32

The values to sum

Return Value

Type: Int32

The sum of the values

▲ See Also

Reference

SampleClass Class SumValues Overload

XMLCommentsExamples Namespace Other Resources overloads

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSumValues Method (IEnumerableInt32, IEnumerableInt32)

This is used to sum two enumerable list of values

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

firstValues

Type: System.Collections.GenericlEnumerableInt32

The first set of values to sum

secondValues

Type: System.Collections.GenericlEnumerableInt32

The second set of values to sum

Return Value

Type: Int32

The sum of the values from both enumerable lists

▲ See Also

Reference

SampleClass Class SumValues Overload XMLCommentsExamples Namespace

Other Resources overloads

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassTemporaryMethc Method

This method will be going away in the production release.

A temporary method

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

▲ Syntax

```
C# VB C++ F#

public void TemporaryMethod()
```

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources

preliminary

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassTokenExample Method

The help file was built on October 10, 2015 using Sandcastle v2015.10.10.0.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void TokenExample()
```

▲ Remarks

For tokens to be resolved, the **API Token Resolution** build component must be added to the SHFB project.

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

token

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassVariousListExam Method

This method shows various examples of the list XML comments element.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

Syntax

```
C# VB C++ F#

public void VariousListExamples()
```


A simple bulleted list. The term and description elements are optional for simple string descriptions.

- First item
- Second item
- Third item

Bullet list with terms and definitions. The term is highlighted and separated from the definition with a dash.

- maxLen field must contain no more than the specified number of characters
- minLen field must contain at least the specified number of characters
- maxVal field must contain a number that is no larger than the specified value
- minVal field must contain a number that is no smaller than the specified value

pattern - field must match the specified regular expression

A simple numbered list. The term and description elements are optional for simple string descriptions.

- 1. First item
- 2. Second item
- 3. Third item

This next numbered list uses the optional start attribute to continue numbering where the last one left off.

- 4. Fourth item
- 5. Fifth item
- 6. Sixth item

Numbered list with terms and definitions.

- maxLen field must contain no more than the specified number of characters
- 2. **minLen** field must contain at least the specified number of characters
- 3. **maxVal** field must contain a number that is no larger than the specified value
- 4. **minVal** field must contain a number that is no smaller than the specified value
- 5. **pattern** field must match the specified regular expression Definition list.

maxLen

field must contain no more than the specified number of characters

minLen

field must contain at least the specified number of characters

maxVal

field must contain a number that is no larger than the specified value

minVal

field must contain a number that is no smaller than the specified value

pattern

field must match the specified regular expression

Two-column table list with terms and definitions.

Item	Description
maxLen	field must contain no more than the specified number of characters
minLen	field must contain at least the specified number of characters
maxVal	field must contain a number that is no larger than the specified value
minVal	field must contain a number that is no smaller than the specified value
pattern	field must match the specified regular expression

A table with multiple columns. term or description can be used to create the columns in each row.

Column 1	Column 2	Column 3	Column 4
R1, C1	R1, C2	R1, C3	R1, C4
R2, C1	R2, C2	R2, C3	R2, C4

▲ See Also

Reference

SampleClass Class

XMLCommentsExamples Namespace

Other Resources

list



SampleClassVariousNoteExar Method

This shows the result of the various note types.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public virtual void VariousNoteExamples()
```

▲ Remarks

These are various examples of the different note types.

Note

This example demonstrates the handling of a note element with no defined type. It defaults to the "note" style.

■ Tip

Always document your code to help others understand how it is used.

■ Notes to Implementers

Override this method in a derived class to do something useful

Notes to Callers

Calling this implementation will have no effect at all

Notes to Inheritors

Types inheriting this base method will have no use for it as it does nothing

Caution

Use of this method is not recommended.

A Caution

XML is case-sensitive so the note type must be entered as shown in order for it to be interpreted correctly.

▲ Important

Calling this method excessively will only slow down your application.

Security Note

It is always safe to call this method.

Security Note

This method requires no special privileges

C# Note

Use parenthesis when calling this method in C#.

■ Visual Basic Note

Parenthesis are not required when calling this method in Visual Basic.

C++ Note

Use parenthesis when calling this method in C++.

J# Note

Use parenthesis when calling this method in J#.

See the note topic for a full list of all possible note types.

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide

Send comments on this topic to Eric@EWoodruff.us



SampleClass Events

The SampleClass type exposes the following members.

▲ Events

	Name	Description
4	SomethingHappened	This event is raised when something interesting happens.

Top

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassSomethingHappe Event

This event is raised when something interesting happens.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F# Copy _

public event EventHandler SomethingHappened
```

Value

Type: SystemEventHandler

▲ Remarks

The OnSomethingHappened method is used to raise this event. This event uses the general EventHandler delegate that passes EventArgs.Empty to the handlers.

Alternate see syntax: EventArgsEmpty

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace SystemEventArgs EventArgsEmpty EventArgsEmpty SystemEventHandler
XMLCommentsExamples.SampleClass.OnSomethingHappened
Other Resources
see
seealso

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClass Operators

The SampleClass type exposes the following members.

■ Operators

	Name	Description
<u>(/-</u> =+) S	Addition	Addition operator overload

Top

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SampleClassAddition Operator

Addition operator overload

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

left

Type: XMLCommentsExamplesSampleClass

The left value

right

Type: XMLCommentsExamplesSampleClass

The right value

Return Value

Type: SampleClass

A new instance containing the sum of the left and right sample

numbers

▲ See Also

Reference

SampleClass Class XMLCommentsExamples Namespace

Other Resources param

[v2014.5.31.0] Sandcastle XML Comments Guide



ThreadSafetyClass Class

This class demonstrates the threadsafety XML comments element.

▲ Inheritance Hierarchy

SystemObject XMLCommentsExamplesThreadSafetyClass

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

C# VE	C++	F#	Copy
public	class	ThreadSafetyClass	

The ThreadSafetyClass type exposes the following members.

→ Constructors

	Name	Description
≅©	ThreadSafetyClass	Initializes a new instance of the ThreadSafetyClass class

Top

Methods

Name	Description
------	-------------

≓♦	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
ĕ o	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≟ û	GetHashCode	Serves as the default hash function. (Inherited from Object.)
∃	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓◊	InstanceMethod	Per the threadsafety XML comments element on the class, the developer has indicated that instance method like this one are not safe for multi-threaded use.
Ģ	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
₫♦ S	StaticMethod	Per the threadsafety XML comments element on the class, the developer has indicated that static methods like this one are safe for multi-threaded use.

Returns a string that represents the current object. (Inherited from Object.)

Top

▲ Thread Safety

Any public **static** (**Shared** in Visual Basic) members of this type are thread safe. Any instance members are not guaranteed to be thread safe.

▲ See Also

Reference XMLCommentsExamples Namespace Other Resources threadsafety

[v2014.5.31.0] Sandcastle XML Comments Guide



ThreadSafetyClass Constructor

Initializes a new instance of the ThreadSafetyClass class

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public ThreadSafetyClass()
```

▲ See Also

Reference

ThreadSafetyClass Class
XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ThreadSafetyClass Methods

The ThreadSafetyClass type exposes the following members.

Methods

	Name	Description
∃ ©	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
ē	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
∃	GetHashCode	Serves as the default hash function. (Inherited from Object.)
₫◊	GetType	Gets the Type of the current instance. (Inherited from Object.)
∄	InstanceMethod	Per the threadsafety XML comments element on the class, the developer has indicated that instance

		method like this one are not safe for multi-threaded use.
ÿ ©	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
=\$	StaticMethod	Per the threadsafety XML comments element on the class, the developer has indicated that static methods like this one are safe for multi-threaded use.
∃ ₩	ToString	Returns a string that represents the current object. (Inherited from Object.)

Top

▲ See Also

Reference

ThreadSafetyClass Class XMLCommentsExamples Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ThreadSafetyClassInstanceMonths Method

Per the threadsafety XML comments element on the class, the developer has indicated that instance method like this one are not safe for multi-threaded use.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void InstanceMethod()
```

▲ See Also

Reference

ThreadSafetyClass Class
XMLCommentsExamples Namespace

Other Resources

threadsafety

[v2014.5.31.0] Sandcastle XML Comments Guide



ThreadSafetyClassStaticMeth Method

Per the threadsafety XML comments element on the class, the developer has indicated that static methods like this one are safe for multi-threaded use.

Namespace: XMLCommentsExamples

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public static void StaticMethod()
```

▲ See Also

Reference

ThreadSafetyClass Class
XMLCommentsExamples Namespace

Other Resources

threadsafety

[v2014.5.31.0] Sandcastle XML Comments Guide



XMLCommentsExamples.Doc Namespace

These are comments from the DocumentationInheritance namespace's NamespaceDoc class

	Class	Description
* \$	AttachedEventsAndPropertiesTest	This class is used to test inherited documentation on attached properties and events
9 \$	BaseInheritDoc	A base class from which to inherit documentation
4 3	CustomException	This exception class is thrown by the application if it encounters an unrecoverable error.
4 \$	DerivedClassWithInheritedDocs	This is a derived class with inherited

		documentation.
₹	ExplicitImplementation	A class with an explicit interface implementation

→ Structures

	Structure	Description
\$	SetDocumentation	A structure with a generic base class that has inherited documentation on its members

▲ See Also

Other Resources

Using NamespaceDoc and NamespaceGroupDoc Classes

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Class

This class is used to test inherited documentation on attached properties and events

▲ Inheritance Hierarchy

SystemObject XMLCommentsExamples.DocumentationInheritanceAt

Namespace: XMLCommentsExamples.DocumentationInheritance Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax



The AttachedEventsAndPropertiesTest type exposes the following members.

Methods

	Name	Description
=♦ S	AddItemActivateHandler	Add an event handler to an object
₫0 S	GetIsBroughtIntoViewWhenSelected	Get the

		property value
∉ \$ S	RemoveItemActivateHandler	Remove an event handler from an object
=\$ S	SetIsBroughtIntoViewWhenSelected	Sets the property value

Тор

Fields

	Name	Description
₽ S	IsBroughtIntoViewWhenSelectedProperty	This defines the IsBroughtIntoVicattached proper
₽ S	ItemActivateEvent	This defines the attached event.

Top

▲ Attached Properties

Name	Description
IsBroughtIntoViewWhenSelected	This attached property indicates whether or not a tree view item is brought

into view when selected.

Top

▲ Attached Events

	Name	Description
4	ItemActivate	This attached event is raised when an item is activated

Top

▲ See Also

Reference

XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

AttachedEventComments AttachedPropertyComments

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Methods

The AttachedEventsAndPropertiesTest type exposes the following members.

Methods

	Name	Description
≅ ù S	AddItemActivateHandler	Add an event handler to an object
≓ ∲ S	GetIsBroughtIntoViewWhenSelected	Get the property value
∃ ∲ S	RemoveItemActivateHandler	Remove an event handler from an object
≘ © S	SetIsBroughtIntoViewWhenSelected	Sets the property value

Top

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Method

Add an event handler to an object

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

Parameters

0

Type: System.WindowsDependencyObject

The dependency object

handler

Type: System.WindowsRoutedEventHandler

The event handler

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Method

Get the property value

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

treeViewItem

Type: System.Windows.ControlsTreeViewItem

The tree view item

Return Value

Type: Boolean

The property value

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace [v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Method

Remove an event handler from an object

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

0

Type: System.WindowsDependencyObject

The dependency object

handler

Type: System.WindowsRoutedEventHandler

The event handler

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Method

Sets the property value

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

treeViewItem

Type: System.Windows.ControlsTreeViewItem

The tree view item

value

Type: SystemBoolean
The property value

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace [v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Fields

The AttachedEventsAndPropertiesTest type exposes the following members.

	Name	Description
∲ S	IsBroughtIntoViewWhenSelectedProperty	This defines the IsBroughtIntoVic attached proper
∲ S	ItemActivateEvent	This defines the attached event.

Top

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Field

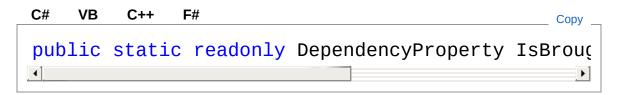
This defines the IsBroughtIntoViewWhenSelected attached property.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax



Field Value

Type: DependencyProperty

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class
XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

AttachedPropertyComments

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Field

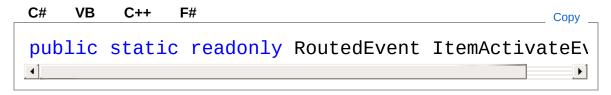
This defines the ItemActivate attached event.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax



Field Value

Type: RoutedEvent

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class

XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

AttachedEventComments

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Attached Properties

The AttachedEventsAndPropertiesTest type exposes the following members.

▲ Attached Properties

Name	Description
IsBroughtIntoViewWhenSelected	This attached property indicates whether or not a tree view item is brought into view when selected.

Top

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Attached Property

This attached property indicates whether or not a tree view item is brought into view when selected.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax



Property Value

Type: Boolean

The default value is false

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class
XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

AttachedPropertyComments

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Attached Events

The AttachedEventsAndPropertiesTest type exposes the following members.

▲ Attached Events

	Name	Description
4	ItemActivate	This attached event is raised when an item is activated

Top

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



AttachedEventsAndProperties Attached Event

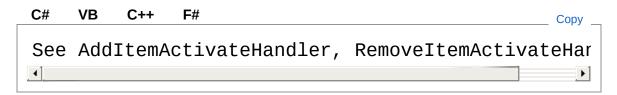
This attached event is raised when an item is activated

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax



Value

Type: System.WindowsRoutedEventHandler

▲ Remarks

There's a bit more too it to get the event raised but this is just a documentation example.

▲ See Also

Reference

AttachedEventsAndPropertiesTest Class

XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

AttachedEventComments

[v2014.5.31.0] Sandcastle XML Comments Guide



BaseInheritDoc Class

A base class from which to inherit documentation

▲ Inheritance Hierarchy

SystemObject XMLCommentsExamples.DocumentationInheritanceBaxMLCommentsExamples.DocumentationInheritanceDerivedClassW

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

C#	VB	C++	F#	Copy _
pub	Lic	class	BaseInheritDoc	

The BaseInheritDoc type exposes the following members.

→ Constructors

	Name	Description
≡	BaseInheritDoc	Constructor

Top

Methods

Name Description

Ξ₩	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
ē ♥	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
Ξ₩	GetHashCode	Serves as the default hash function. (Inherited from Object.)
Ξ₩	GetType	Gets the Type of the current instance. (Inherited from Object.)
Ģ	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)

e ù ¥	MethodWithExample	Summary for the method with an example
= ↓ F	MethodWithLotsOfComments	The method in the base class has lots of comments.
ş © ¥	MethodWithTwoExamples	A method with two examples
≡	ToString	The ToString implementation for BaseInheritDoc (Overrides ObjectToString.)

Top

▲ Remarks

These remarks are for the base class.

This information applies to all classes that derive from BaseInheritDoc:

- Point #1.
- Point #2.
- Point #3.

▲ See Also

Reference

XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

inheritdoc

[v2014.5.31.0] Sandcastle XML Comments Guide



BaseInheritDoc Constructor

Constructor

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F# Copy

public BaseInheritDoc()
```

▲ See Also

Reference

BaseInheritDoc Class

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



BaseInheritDoc Methods

The BaseInheritDoc type exposes the following members.

Methods

	Name	Description
≘⊷	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
Ģ	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
=	GetHashCode	Serves as the default hash function. (Inherited from

		Object.)
≅ 🗳	GetType	Gets the Type of the current instance. (Inherited from Object.)
Ģ ̃ ♥	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
=♦ !	MethodWithExample	Summary for the method with an example
=↓ !	MethodWithLotsOfComments	The method in the base class has lots of comments.
∮ ¥	MethodWithTwoExamples	A method with two examples
=	ToString	The ToString implementation for BaseInheritDoc (Overrides ObjectToString.)

Top

▲ See Also

Reference

BaseInheritDoc Class

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



BaseInheritDocMethodWithEx Method

Summary for the method with an example

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public virtual bool MethodWithExample()
```

Return Value

Type: Boolean
True all the time

This example is from the base class

```
// 'x' is always true
bool x = instance.MethodWithExample();
```

▲ See Also

Reference

BaseInheritDoc Class

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



BaseInheritDocMethodWithLo Method

The method in the base class has lots of comments.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public virtual void MethodWithLotsOfComments(
          int x
)
```

Parameters

X

Type: SystemInt32
The parameter

▲ Exceptions

Exception	Condition	
ArgumentException	Thrown if x is zero	
ArgumentOutOfRangeException	Thrown if x is less than zero.	

▲ Remarks

Remarks for the base class

▲ Examples

/// Example goes here

▲ See Also

Reference

BaseInheritDoc Class

XMLCommentsExamples.DocumentationInheritance Namespace

BaseInheritDocToString

BaseInherit Doc Method With Example

[v2014.5.31.0] Sandcastle XML Comments Guide

Send comments on this topic to Eric@EWoodruff.us

Copy



BaseInheritDocMethodWithTw Method

A method with two examples

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#
protected virtual void MethodWithTwoExamples()
```

▲ Examples

This is example #1:

```
// Example #1
```

This is example #2:

```
// Example #2
```

▲ See Also

Reference

BaseInheritDoc Class

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



BaseInheritDocToString Method

The ToString implementation for BaseInheritDoc

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public override string ToString()
```

Return Value

Type: String

A string representing the object

▲ See Also

Reference

BaseInheritDoc Class

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Class

This exception class is thrown by the application if it encounters an unrecoverable error.

▲ Inheritance Hierarchy

SystemObject SystemException

XMLCommentsExamples.DocumentationInheritanceCustomExcepti

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

[SerializableAttribute]
public class CustomException : Exception
```

The CustomException type exposes the following members.

→ Constructors

	Name	Description
≅©	CustomException	Default constructor.
≡	CustomException(String)	Initializes a new instance of the

		Exception class with a specified error message.
Ģ̃ ©	CustomException(SerializationInfo, StreamingContext)	Initializes a new instance of the Exception class with serialized data.
≟	CustomException(String, Exception)	Initializes a new instance of the Exception class with a specified error message and a reference to the inner exception that is the cause of this exception.

Тор

▲ Properties

Name	Description
Data	Gets a collection of key/value

	pairs that provide additional user- defined information about the exception. (Inherited from Exception.)
HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
Message	Gets a message that describes the current exception. (Inherited from Exception.)
Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the immediate frames on the call stack. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Methods

	Name	Description
a∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
Ģ̄ Û	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≡	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
≡	GetHashCode	Serves as the default hash function. (Inherited from Object.)
□	GetObjectData	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
∃	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)

ĕ	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
ΞΦ	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Top

■ Events

Name	Description
SerializeObjectState	Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)

Top

▲ See Also

Reference

XMLCommentsExamples.DocumentationInheritance Namespace Other Resources

inheritdoc

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Constructor

There are four overloads for the constructor

▲ Overload List

	Name	Description
≟	CustomException	Default constructor.
≅Ŵ	CustomException(String)	Initializes a new instance of the Exception class with a specified error message.
Ģ ¯	CustomException(SerializationInfo, StreamingContext)	Initializes a new instance of the Exception class with serialized data.
⊕	CustomException(String, Exception)	Initializes a new instance of the Exception class with a

specified error message and a reference to the inner exception that is the cause of this exception.

Top

▲ See Also

Reference

CustomException Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Constructor

Default constructor.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public CustomException()
```

▲ See Also

Reference

CustomException Class CustomException Overload

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Constructor (String)

Initializes a new instance of the Exception class with a specified error message.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

message

Type: SystemString

The message that describes the error.

▲ See Also

Reference

CustomException Class
CustomException Overload

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of the Exception class with serialized data.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

info

Type: System.Runtime.SerializationSerializationInfo
The SerializationInfo that holds the serialized object data about the exception being thrown.

context

Type: System.Runtime.SerializationStreamingContext
The StreamingContext that contains contextual information
about the source or destination.

▲ Exceptions

Exception	Condition
ArgumentNullException	The <i>info</i> parameter is null.
SerializationException	The class name is null or HResult is zero (0).

▲ See Also

Reference

CustomException Class

CustomException Overload

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Constructor (String, Exception)

Initializes a new instance of the Exception class with a specified error message and a reference to the inner exception that is the cause of this exception.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

message

Type: SystemString

The error message that explains the reason for the exception. *innerException*

Type: SystemException

The exception that is the cause of the current exception, or a null reference (Nothing in Visual Basic) if no inner exception is specified.

▲ See Also

Reference

CustomException Class
CustomException Overload
XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Properties

The CustomException type exposes the following members.

▲ Properties

	Name	Description
	Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
É	HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from Exception.)
É	InnerException	Gets the Exception instance that caused the current exception. (Inherited from Exception.)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the

	application or the object that causes the error. (Inherited from Exception.)
StackTrace	Gets a string representation of the immediate frames on the call stack. (Inherited from Exception.)
TargetSite	Gets the method that throws the current exception. (Inherited from Exception.)

Top

▲ See Also

Reference

CustomException Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Methods

The CustomException type exposes the following members.

Methods

	Name	Description
≅ ©	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
∮	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≡	GetBaseException	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
≘∲	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡	GetObjectData	When overridden in a derived

		class, sets the SerializationInfo with information about the exception. (Inherited from Exception.)
≟	GetType	Gets the runtime type of the current instance. (Inherited from Exception.)
- 	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
≅∳	ToString	Creates and returns a string representation of the current exception. (Inherited from Exception.)

Top

▲ See Also

Reference

CustomException Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



CustomException Events

The CustomException type exposes the following members.

▲ Events

Name	Description
SerializeObjectState	Occurs when an exception is serialized to create an exception state object that contains serialized data about the exception. (Inherited from Exception.)

Top

▲ See Also

Reference

CustomException Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

This is a derived class with inherited documentation.

▲ Inheritance Hierarchy

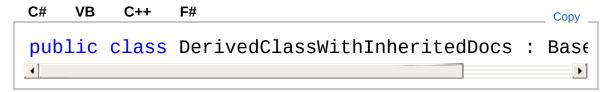
SystemObject XMLCommentsExamples.DocumentationInheritanceBa XMLCommentsExamples.DocumentationInheritanceDerivedClassW

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax



The DerivedClassWithInheritedDocs type exposes the following members.

→ Constructors

	Name	Description
ΞΦ	DerivedClassWithInheritedDocs	Initializes a new instance DerivedClassWithInherite class

Top

▲ Methods

	Name	Description
∉©	Equals	Determines whether the spethe current object. (Inherited from Object.)
Ģ [¯] ©	Finalize	Allows an object to try to fre other cleanup operations be garbage collection. (Inherited from Object.)
€₩	GetHashCode	Serves as the default hash (Inherited from Object.)
€₩	GetType	Gets the Type of the current (Inherited from Object.)
∳	MemberwiseClone	Creates a shallow copy of the (Inherited from Object.)
e ù F	MethodUsingSharedExample	This uses a shared example that is not public and this do
· F	MethodWithExample	An override of the method v (Overrides BaseInheritDock
∃	MethodWithLotsOfComments	The method in the base class (Overrides BaseInheritDocMethodWith
ş® ¥	MethodWithTwoExamples	This only includes one of the (Overrides BaseInheritDocMethodWith)
≡	OverloadedMethod(String)	This overloaded method do

∉∳	OverloadedMethod(String, Int32)	There are three overloads for
≟	OverloadedMethod(String, String)	This overloaded method do
≝	ToString	Returns a string that represe (Overrides BaseInheritDocT

Top

▲ Remarks

This will inherit just the last <para> tag from the base class's <remarks> tag:

This information applies to all classes that derive from BaseInheritDoc:

- Point #1.
- Point #2.
- Point #3.

▲ See Also

Reference

XMLCommentsExamples.DocumentationInheritance Namespace

Other Resources

inheritdoc

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo Constructor

Initializes a new instance of the DerivedClassWithInheritedDocs class

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public DerivedClassWithInheritedDocs()
```

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

The DerivedClassWithInheritedDocs type exposes the following members.

Methods

	Name	Description
≡ ©	Equals	Determines whether the spetthe current object. (Inherited from Object.)
Ģ ©	Finalize	Allows an object to try to free other cleanup operations be garbage collection. (Inherited from Object.)
=0	GetHashCode	Serves as the default hash (Inherited from Object.)
=	GetType	Gets the Type of the current (Inherited from Object.)
Ģ	MemberwiseClone	Creates a shallow copy of the (Inherited from Object.)
e ù F	MethodUsingSharedExample	This uses a shared example that is not public and this do
e ù F	MethodWithExample	An override of the method v (Overrides BaseInheritDock

Ξ₩	MethodWithLotsOfComments	The method in the base clas (Overrides BaseInheritDocMethodWith
ş ⊕ ¥	MethodWithTwoExamples	This only includes one of the (Overrides BaseInheritDocMethodWith)
≡	OverloadedMethod(String)	This overloaded method do
ΞΦ	OverloadedMethod(String, Int32)	There are three overloads for
≘ ©	OverloadedMethod(String, String)	This overloaded method do
≓	ToString	Returns a string that represe (Overrides BaseInheritDocT

Top

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

This uses a shared example from a base member that is not public and this doesn't override.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void MethodUsingSharedExample()
```

▲ Examples

This is example #2:

```
// Example #2
```

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

An override of the method with an example

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public override bool MethodWithExample()
```

Return Value

Type: Boolean

Always returns false

▲ Examples

This example is from the base class

```
// 'x' is always true
bool x = instance.MethodWithExample();
```

This example applies to the derived class:

```
if(derivedInstance.MethodWithExample())
    Console.WriteLine("This is never reached");
```

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

The method in the base class has lots of comments.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public override void MethodWithLotsOfComments(
    int x
)
```

Parameters

X

Type: SystemInt32
The parameter

▲ Remarks

Remarks for the base class

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

This only includes one of the examples

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#
protected override void MethodWithTwoExamples()
```

▲ Examples

This is example #2:

```
// Example #2
```

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

There are three overloads for this method.

■ Overload List

	Name	Description
≓	OverloadedMethod(String)	This overloaded method does something
≘	OverloadedMethod(String, Int32)	There are three overloads for this method.
≘©	OverloadedMethod(String, String)	This overloaded method does something

Top

▲ Remarks

These remarks are from the overloads tag on the first version.

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDoo Method (String)

This overloaded method does something

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void OverloadedMethod(
          string p1
)
```

Parameters

р1

Type: SystemString
The string parameter

▲ See Also

Reference

DerivedClassWithInheritedDocs Class

OverloadedMethod Overload

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo Method (String, Int32)

There are three overloads for this method.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void OverloadedMethod(
          string p1,
          int x
)
```

Parameters

```
p1Type: SystemStringThe string parameterXType: SystemInt32An integer parameter
```

▲ Remarks

These remarks are from the overloads tag on the first version.

▲ See Also

Reference

DerivedClassWithInheritedDocs Class OverloadedMethod Overload XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDoo Method (String, String)

This overloaded method does something

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public void OverloadedMethod(
          string p1,
          string p2
)
```

Parameters

p1
Type: SystemString
The string parameter
p2

Type: SystemString

The second string parameter

▲ See Also

Reference

DerivedClassWithInheritedDocs Class OverloadedMethod Overload XMLCommentsExamples.DocumentationInheritance Namespace [v2014.5.31.0] Sandcastle XML Comments Guide



DerivedClassWithInheritedDo

Returns a string that represents the current object.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public override string ToString()
```

Return Value

Type: String

A string that represents the current object.

▲ See Also

Reference

DerivedClassWithInheritedDocs Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementation Class

A class with an explicit interface implementation

▲ Inheritance Hierarchy

SystemObject XMLCommentsExamples.DocumentationInheritanceEx

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public class ExplicitImplementation : ICollection ICloneable, IEnumerable
```

The ExplicitImplementation type exposes the following members.

→ Constructors

	Name	Description
=	ExplicitImplementation	Initializes a new instance of the ExplicitImplementation class

Top

	Name	Description
∃	Clone	Creates a new object that is a copy of the current instance.
≅©	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
Ģ	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≟	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≟ ∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
Ģ ♥	MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
≡	ToString	Returns a string that represents the current object. (Inherited from Object.)

Top

▲ Explicit Interface Implementations

	Name	Description
⊶o g̃ ∲	ICollectionCopyTo	Copies the elements of the ICollection to an Array, starting at a particular Array index.
⊶ <u>™</u>	ICollectionCount	Gets the number of elements contained in the ICollection.
⊶o gi∳	IEnumerableGetEnumerator	Returns an enumerator that iterates through a collection.
→ ○ [28]	ICollectionIsSynchronized	Gets a value indicating whether access to the ICollection is synchronized (thread safe).
⊶o maga	ICollectionSyncRoot	Gets an object that can be used to synchronize access to the ICollection.

Top

▲ Remarks

Note that you must enable the

DocumentExplicitInterfaceImplementations SHFB project options in order to see the explicitly implemented members.

▲ See Also

Reference

XMLCommentsExamples.DocumentationInheritance Namespace
Other Resources
inheritdoc

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementation Constructor

Initializes a new instance of the ExplicitImplementation class

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public ExplicitImplementation()
```

▲ See Also

Reference

ExplicitImplementation Class
XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementation Properties

▲ Explicit Interface Implementations

	Name	Description
~ ○ <u>P</u>	ICollectionCount	Gets the number of elements contained in the ICollection.
~ ○ 📸	ICollectionIsSynchronized	Gets a value indicating whether access to the ICollection is synchronized (thread safe).
⊶0 📸	ICollectionSyncRoot	Gets an object that can be used to synchronize access to the ICollection.

Top

▲ See Also

Reference

ExplicitImplementation Class
XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementationICollect Property

Gets the number of elements contained in the ICollection.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

int ICollection.Count { get; }
```

Return Value

Type: Int32

The number of elements contained in the ICollection.

Implements
ICollectionCount

▲ See Also

Reference

ExplicitImplementation Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementationICollect Property

Gets a value indicating whether access to the ICollection is synchronized (thread safe).

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

bool ICollection.IsSynchronized { get; }
```

Return Value

Type: Boolean

true if access to the ICollection is synchronized (thread safe);

otherwise, false.

Implements

ICollectionIsSynchronized

▲ See Also

Reference

ExplicitImplementation Class

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementationICollect Property

Gets an object that can be used to synchronize access to the ICollection.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F# Copy _
Object ICollection.SyncRoot { get; }
```

Return Value

Type: Object

An object that can be used to synchronize access to the ICollection.

Implements

ICollectionSyncRoot

▲ Remarks

This is a dummy class and always returns null.

▲ See Also

Reference

ExplicitImplementation Class
XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementation Methods

The ExplicitImplementation type exposes the following members.

Methods

	Name	Description
∃©	Clone	Creates a new object that is a copy of the current instance.
∃ ♠	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
ē Û	Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
≡	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≟ ∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
ē	MemberwiseClone	Creates a shallow copy of the

		current Object. (Inherited from Object.)
≡	ToString	Returns a string that represents the current object (Inherited from Object.)

Top

▲ Explicit Interface Implementations

	Name	Description
⊶o g̃ ©	ICollectionCopyTo	Copies the elements of the ICollection to an Array, starting at a particular Array index.
⊶o g̃♥	IEnumerableGetEnumerator	Returns an enumerator that iterates through a collection.

Top

▲ See Also

Reference

ExplicitImplementation Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementationICollect Method

Copies the elements of the ICollection to an Array, starting at a particular Array index.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

void ICollection.CopyTo(
         Array array,
         int index
)
```

Parameters

array

Type: SystemArray

The one-dimensional Array that is the destination of the elements copied from ICollection. The Array must have zero-based indexing.

index

Type: SystemInt32

The zero-based index in array at which copying begins.

Implements

ICollectionCopyTo(Array, Int32)

▲ Exceptions

Exception	Condition
ArgumentNullException	array is null.
ArgumentOutOfRangeException	index is less than zero.
ArgumentException	array is multidimensionalor- The number of elements in the source ICollection is greater than the available space from <i>index</i> to the end of the destination <i>array</i> or- The type of the source ICollection cannot be cast automatically to the type of the destination <i>array</i> .

▲ See Also

Reference

ExplicitImplementation Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementationIEnume Method

Returns an enumerator that iterates through a collection.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

IEnumerator IEnumerable.GetEnumerator()
```

Return Value

Type: IEnumerator

This is a dummy class so it throws an exception

Implements

IEnumerableGetEnumerator

▲ See Also

Reference

ExplicitImplementation Class XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



ExplicitImplementationClone Method

Creates a new object that is a copy of the current instance.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public Object Clone()
```

Return Value

Type: Object

A new object that is a copy of this instance.

Implements
ICloneableClone

▲ See Also

Reference

ExplicitImplementation Class
XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentation Structure

A structure with a generic base class that has inherited documentation on its members

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

The SetDocumentation type exposes the following members.

→ Properties

Name	Description
Count	Gets the number of elements contained in the ICollectionT.
IsReadOnly	Gets a value indicating whether the ICollectionT is read-only.

Top

Methods

	Name	Description
≟	Add	Adds an element to the current set and returns a value to indicate if the element was successfully added.
≅ ∳	Clear	Removes all items from the ICollectionT.
∃	Contains	Determines whether the ICollectionT contains a specific value.
≅∳	СоруТо	Copies the elements of the ICollectionT to an Array, starting at a particular Array index.
∃	Equals	Indicates whether this instance and a specified object are equal. (Inherited from ValueType.)
⊕	ExceptWith	Removes all elements in the specified collection from the current set.
⊕	GetEnumerator	Returns an enumerator that iterates through the collection.
≅ ∳	GetHashCode	Returns the hash code for this instance. (Inherited from ValueType.)

⊒	GetType	Gets the Type of the current instance. (Inherited from Object.)
≘₩	IntersectWith	Modifies the current set so that it contains only elements that are also in a specified collection.
≓	IsProperSubsetOf	Determines whether the current set is a proper (strict) subset of a specified collection.
ΞΦ	IsProperSupersetOf	Determines whether the current set is a proper (strict) superset of a specified collection.
≘•	IsSubsetOf	Determines whether a set is a subset of a specified collection.
ΞΦ	IsSupersetOf	Determines whether the current set is a superset of a specified collection.
≘ 	Overlaps	Determines whether the current set overlaps with the specified collection.
≟	Remove	Removes the first occurrence of a specific object from the ICollectionT.
≡	SetEquals	Determines whether the current set and the

		specified collection contain the same elements.
⊒₩	SymmetricExceptWith	Modifies the current set so that it contains only elements that are present either in the current set or in the specified collection, but not both.
⊒	ToString	Returns the fully qualified type name of this instance. (Inherited from ValueType.)
Ξ₩	UnionWith	Modifies the current set so that it contains all elements that are present in the current set, in the specified collection, or in both.

Top

▲ Explicit Interface Implementations

	Name	Description
∞ <u>§</u>	ICollectionInt32Add	Adds an item to the ICollectionT.
⊶o <u>ē</u>	IEnumerableGetEnumerator	Returns an enumerator that iterates through a collection.

Top

▲ See Also

Reference

 $XML Comments \textbf{Examples}. Documentation Inheritance\ Names pace$

Other Resources

inheritdoc

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentation Properties

The SetDocumentation type exposes the following members.

▲ Properties

Name	Description
Count	Gets the number of elements contained in the ICollectionT.
IsReadOnly	Gets a value indicating whether the ICollectionT is read-only.

Top

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationCount Property

Gets the number of elements contained in the |CollectionT.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public int Count { get; }
```

Return Value

Type: Int32

The number of elements contained in the ICollectionT.

Implements

ICollectionTCount

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIsReadOnly Property

Gets a value indicating whether the ICollectionT is read-only.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public bool IsReadOnly { get; }
```

Return Value

Type: Boolean

true if the ICollectionT is read-only; otherwise, false.

Implements

ICollectionTIsReadOnly

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentation Methods

The SetDocumentation type exposes the following members.

Methods

Adds an element to the current set and returns a value to indicate if the element was successfully added. Removes all items from the ICollectionT. Determines whether the ICollectionT contains a
the ICollectionT. Determines whether the ICollectionT contains a
ICollectionT contains a
specific value.
Copies the elements of the ICollectionT to an Array, starting at a particular Array index.
Indicates whether this instance and a specified object are equal. (Inherited from ValueType.)

	ExceptWith	Removes all elements in the specified collection from the current set.
≓	GetEnumerator	Returns an enumerator that iterates through the collection.
≓	GetHashCode	Returns the hash code for this instance. (Inherited from ValueType.)
≡	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡	IntersectWith	Modifies the current set so that it contains only elements that are also in a specified collection.
≓	IsProperSubsetOf	Determines whether the current set is a proper (strict) subset of a specified collection.
≟	IsProperSupersetOf	Determines whether the current set is a proper (strict) superset of a specified collection.
≟	IsSubsetOf	Determines whether a set is a subset of a specified collection.
≡	IsSupersetOf	Determines whether the current set is a superset

		of a specified collection.
Ξ₩	Overlaps	Determines whether the current set overlaps with the specified collection.
Ξ₩	Remove	Removes the first occurrence of a specific object from the ICollectionT.
∃	SetEquals	Determines whether the current set and the specified collection contain the same elements.
± ©	SymmetricExceptWith	Modifies the current set so that it contains only elements that are present either in the current set or in the specified collection, but not both.
⊕	ToString	Returns the fully qualified type name of this instance. (Inherited from ValueType.)
≅₩	UnionWith	Modifies the current set so that it contains all elements that are present in the current set, in the specified collection, or in both.

▲ Explicit Interface Implementations

	Name	Description
⊶o g̃♥	ICollectionInt32Add	Adds an item to the ICollectionT.
⊶o ḡ	IEnumerableGetEnumerator	Returns an enumerator that iterates through a collection.

Top

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationICollectionI Method

Adds an item to the ICollectionT.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

item

Type: SystemInt32

The object to add to the ICollectionT.

Implements

ICollectionTAdd(T)

▲ Exceptions

Exception	Condition
NotSupportedException	The ICollectionT is read-only.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIEnumerable Method

Returns an enumerator that iterates through a collection.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

C# VB C++ F#

IEnumerator IEnumerable.GetEnumerator()

Return Value

Type: IEnumerator

An IEnumerator object that can be used to iterate through the collection.

Implements

IEnumerableGetEnumerator

▲ See Also

Reference

SetDocumentation Structure

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationAdd Method

Adds an element to the current set and returns a value to indicate if the element was successfully added.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public bool Add(
    int item
)
```

Parameters

item

Type: SystemInt32

The element to add to the set.

Return Value

Type: Boolean

true if the element is added to the set; false if the element is already in the set.

Implements

ISetTAdd(T)

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationClear Method

Removes all items from the ICollectionT.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void Clear()
```

Implements

ICollectionTClear

▲ Exceptions

Exception	Condition
NotSupportedException	The ICollectionT is read-only.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationContains Method

Determines whether the ICollectionT contains a specific value.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public bool Contains(
    int item
)
```

Parameters

item

Type: SystemInt32

The object to locate in the ICollectionT.

Return Value

Type: Boolean

true if *item* is found in the ICollectionT; otherwise, false.

Implements

ICollectionTContains(T)

▲ See Also

Reference

SetDocumentation Structure

XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationCopyTo Method

Copies the elements of the ICollectionT to an Array, starting at a particular Array index.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public void CopyTo(
    int[] array,
    int arrayIndex
)
```

Parameters

array

Type: SystemInt32

The one-dimensional Array that is the destination of the elements copied from ICollectionT. The Array must have zero-based indexing.

arrayIndex

Type: SystemInt32

The zero-based index in array at which copying begins.

Implements

ICollectionTCopyTo(T, Int32)

▲ Exceptions

Exception	Condition
ArgumentNullException	array is null.
ArgumentOutOfRangeException	arrayIndex is less than 0.
ArgumentException	The number of elements in the source ICollectionT is greater than the available space from <i>arrayIndex</i> to the end of the destination <i>array</i> .

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationExceptWith Method

Removes all elements in the specified collection from the current set.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericIEnumerableInt32

The collection of items to remove from the set.

Implements

ISetTExceptWith(IEnumerableT)

▲ Exceptions

Exception	Condition
ArgumentNullException	other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationGetEnumer Method

Returns an enumerator that iterates through the collection.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

```
C# VB C++ F#

public IEnumerator<int> GetEnumerator()
```

Return Value

Type: IEnumeratorInt32

An enumerator that can be used to iterate through the collection.

Implements

IEnumerableTGetEnumerator

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIntersectWi Method

Modifies the current set so that it contains only elements that are also in a specified collection.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericlEnumerableInt32

The collection to compare to the current set.

Implements

ISetTIntersectWith(IEnumerableT)

▲ Exceptions

Exception	Condition
ArgumentNullException	other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIsProperSu Method

Determines whether the current set is a proper (strict) subset of a specified collection.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericIEnumerableInt32

The collection to compare to the current set.

Return Value

Type: Boolean

true if the current set is a proper subset of other; otherwise, false.

Implements

ISetTIsProperSubsetOf(IEnumerableT)

▲ Exceptions

Exception Condition

other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIsProperSu Method

Determines whether the current set is a proper (strict) superset of a specified collection.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericIEnumerableInt32

The collection to compare to the current set.

Return Value

Type: Boolean

true if the current set is a proper superset of other; otherwise, false.

Implements

ISetTIsProperSupersetOf(IEnumerableT)

▲ Exceptions

Exception Condition

other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIsSubsetOf Method

Determines whether a set is a subset of a specified collection.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

→ Syntax

Parameters

other

Type: System.Collections.GenericlEnumerableInt32

The collection to compare to the current set.

Return Value

Type: Boolean

true if the current set is a subset of other; otherwise, false.

Implements

ISetTIsSubsetOf(IEnumerableT)

▲ Exceptions

Exception	Condition
ArgumentNullException	other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationIsSuperset@ Method

Determines whether the current set is a superset of a specified collection.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericlEnumerableInt32

The collection to compare to the current set.

Return Value

Type: Boolean

true if the current set is a superset of *other*; otherwise, false.

Implements

ISetTIsSupersetOf(IEnumerableT)

▲ Exceptions

Exception Condition

other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationOverlaps Method

Determines whether the current set overlaps with the specified collection.

Namespace: XMLCommentsExamples.DocumentationInheritance
Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericIEnumerableInt32

The collection to compare to the current set.

Return Value

Type: Boolean

true if the current set and *other* share at least one common element; otherwise, false.

Implements

ISetTOverlaps(IEnumerableT)

Exception	Condition
ArgumentNullException	other is null.

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationRemove Method

Removes the first occurrence of a specific object from the ICollectionT.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

```
C# VB C++ F#

public bool Remove(
    int item
)
```

Parameters

item

Type: SystemInt32

The object to remove from the ICollectionT.

Return Value

Type: Boolean

true if *item* was successfully removed from the ICollectionT; otherwise, false. This method also returns false if *item* is not found in the original ICollectionT.

Implements

ICollectionTRemove(T)

Exception	Condition
NotSupportedException	The ICollectionT is read-only.

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationSetEquals Method

Determines whether the current set and the specified collection contain the same elements.

Namespace: XMLCommentsExamples.DocumentationInheritance Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericIEnumerableInt32

The collection to compare to the current set.

Return Value

Type: Boolean

true if the current set is equal to other; otherwise, false.

Implements

ISetTSetEquals(IEnumerableT)

▲ Exceptions

Exception Condition

other is null.

▲ See Also

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationSymmetricE Method

Modifies the current set so that it contains only elements that are present either in the current set or in the specified collection, but not both.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericIEnumerableInt32

The collection to compare to the current set.

Implements

ISetTSymmetricExceptWith(IEnumerableT)

Exception	Condition
ArgumentNullException	other is null.

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide



SetDocumentationUnionWith Method

Modifies the current set so that it contains all elements that are present in the current set, in the specified collection, or in both.

Namespace: XMLCommentsExamples.DocumentationInheritance

Assembly: XMLCommentsExamples (in XMLCommentsExamples.dll)

Version: 2014.1.26.0

■ Syntax

Parameters

other

Type: System.Collections.GenericlEnumerableInt32

The collection to compare to the current set.

Implements

ISetTUnionWith(IEnumerableT)

Exception	Condition
ArgumentNullException	other is null.

Reference

SetDocumentation Structure XMLCommentsExamples.DocumentationInheritance Namespace

[v2014.5.31.0] Sandcastle XML Comments Guide