ComponentOne

DynamicHelp for WinForms

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

DynamicHelp for WinForms Overview

DynamicHelp for WinForms is a container that allows you to view Help files in WinForms applications. Developers or Help authors can also use **DynamicHelp** to assign specific Help topics to controls within an application. The C1DynamicHelp WinForms control displays context-sensitive help corresponding to the control that is being hovered over or the control that has the focus. This type of Help system is an advanced feature available in applications such as Visual Studio, Microsoft Office, Doc-To-Help, and others.

C1DynamicHelp is conceptually similar to using tooltips to provide Help on individual controls and components. The main difference is that tooltips typically contain only a single sentence or paragraph. An actual Help topic, by contrast, typically contains more content, often including hyperlinks, images, tables, and so on. A C1DynamicHelp Help topic also supports selection and can be copied to the Clipboard.

Topic mapping can be done by software developers or Help authors. The following topics provide detailed instructions for both groups.

- For Developers: Mapping at Design Time
- For Help Authors: Mapping in Authoring Mode

Help with WinForms Edition

Getting Started

For information on installing **ComponentOne Studio WinForms Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with WinForms Edition.

Key Features

The main features of **DynamicHelp for WinForms** include:

- C1DynamicHelp integrates with the application interface

  C1DynamicHelp improves the usability of your application by offering immediate and relevant Help as it is needed, right within the interface of the application. A separate Help system doesn't need to be opened and searched.

- Developers or Help Authors can easily perform topic mapping

  Software developers can map topics to controls at design time, or Help Authors can use **authoring mode**, a special run-time mode with a simple interface, to do the mapping themselves. No additional code needs to be added once C1DynamicHelp is integrated with the application.

- Supports Help content selection and copying

  The Help that is displayed in the C1DynamicHelp Help window can be selected and copied to the Clipboard to be pasted into another application.

- Quickly activate/deactivate **authoring mode**

  The software developer can specify a way for Help authors to quickly activate and deactivate **authoring mode** for topic mapping. Any method can be used, such as a keystroke combination, an environment variable, a .config file, and so on.

DynamicHelp for WinForms Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studio.

Please refer to the pre-installed product samples through the following path:

Documents\ComponentOne Samples\WinForms

The following tables provide a short description for each sample.

Visual Basic Samples

Sample	Description
AddingUICommands	This sample shows how to add a toolbar to a **C1DynamicHelp** control to provide more functionality: • Allows users to navigate back/forward through help navigation history. • Allows users to open the Help file in an external window on a specified tab. • Allows users to pin the current topic down in order to temporarily disable changing topics. This sample uses the **C1DynamicHelp** control.
MultipleForms	This sample demonstrates how to use **C1DynamicHelp** controls on multiple forms, how to set them up to use a single help source, and how to refresh the TopicMap to support controls created dynamically. This sample uses the **C1DynamicHelp** control.
UsingUIElementResolver	This sample demonstrates how to create a custom UIElementResolver and use it in the **C1DynamicHelp** control. This sample uses the **C1DynamicHelp** control.
WorkingWithTopicMap	This sample demonstrates how to load topic map from different locations: from a default topic map file, from a custom file, from application resources. It also shows how to save a modified topic map to different locations. This sample uses the **C1DynamicHelp** control.

C# Samples

Sample	Description
AddingUICommands	This sample shows how to add a toolbar to a **C1DynamicHelp** control to provide more functionality: • Allows users to navigate back/forward through help navigation history. • Allows users to open the Help file in an external window on a specified tab. • Allows users to pin the current topic down in order to temporarily disable changing topics. This sample uses the **C1DynamicHelp** control.
MultipleForms	This sample demonstrates how to use **C1DynamicHelp** controls on multiple forms, how to set them up to use a single help source, and how to refresh the TopicMap to support controls created dynamically. This sample uses the **C1DynamicHelp** control.
UsingUIElementResolver	This sample demonstrates how to create a custom UIElementResolver and use it in the

	C1DynamicHelp control. This sample uses the **C1DynamicHelp** control.
WorkingWithTopicMap	This sample demonstrates how to load topic map from different locations: from a default topic map file, from a custom file, from application resources. It also shows how to save a modified topic map to different locations. This sample uses the **C1DynamicHelp** control.

Using DynamicHelp for WinForms

The process of mapping Help topics to controls within a software application has often been thought of as an error-prone process of passing a topic map back and forth between Help authors and software developers. **DynamicHelp for WinForms** eliminates that painful process by providing two options:

- Software developers can use C1DynamicHelp at design time to map the topics to controls.

  or

- Help authors can map topics to controls themselves using the C1DynamicHelp authoring mode, without having to go through a software developer.

C1DynamicHelp gives you the ability to map a topic to any standard .NET controls and even parts of those controls. You can also map to custom controls. See the Advanced Features topic of this documentation for more information on mapping to control parts or custom controls.

DynamicHelp for WinForms supports several Help formats including HTML Help and NetHelp. No matter who is going to perform the topic mapping, the software developer must specify the Help file that is going to be used in the application before mapping can occur.

The following topics explain both ways to perform topic mapping.

> **Note:** If you are the software developer of the application and would like the Help author to map topics to controls, please provide them with this documentation. The For Help Authors: Mapping in Authoring Mode topic provides all the information needed for topic mapping at run time in authoring mode.

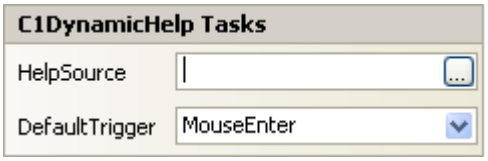
For Developers: Mapping at Design Time

C1DynamicHelp provides visual editing at design time to make it easier to map specific Help topics to controls within an application. You can use the **C1DynamicHelp smart tag** and **Select Help Topic** dialog box to quickly perform topic mapping.

C1DynamicHelp Smart Tag

In Visual Studio, the C1DynamicHelp component includes a smart tag. A smart tag (▶) represents a short-cut tasks menu that provides the most commonly used properties for a control.

To access the **C1DynamicHelp Tasks** menu, click the smart tag in the upper-right corner of the C1DynamicHelp control.



- **HelpSource Property**

  Click the **ellipsis** button to locate and select the .chm or NetHelp .htm file that will be used for topic mapping.

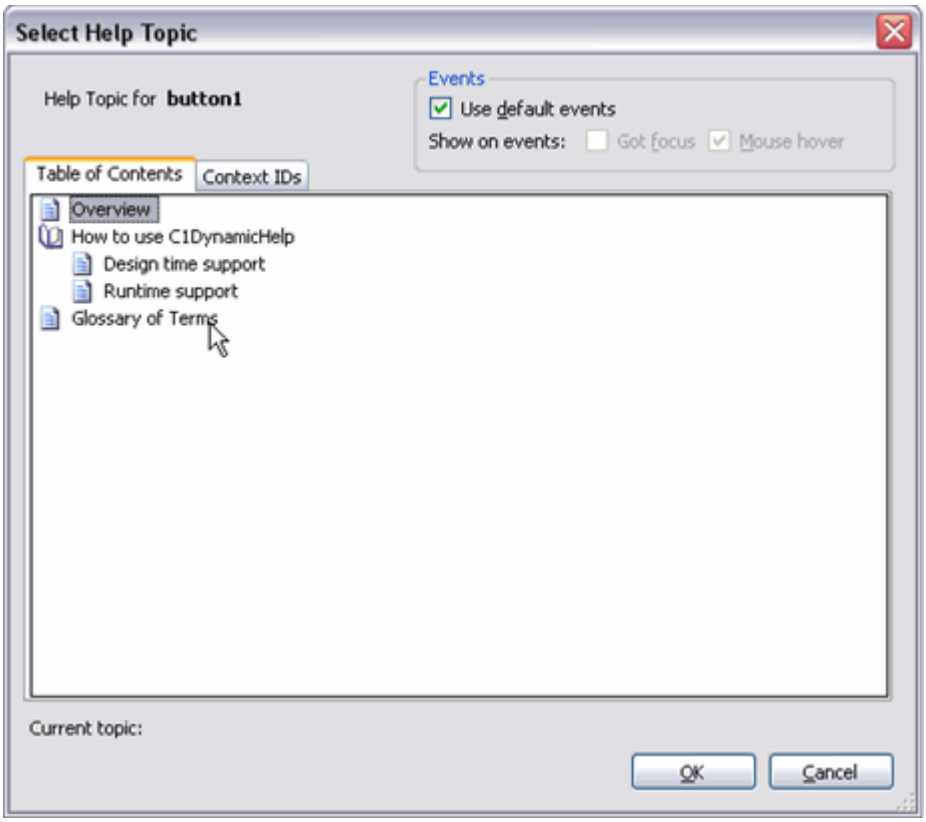
  > **Note:**If using NetHelp, use the default.htm or the file name that was used as the base URL for the NetHelp target.

- **DefaultTrigger Property**

  The DefaultTrigger property allows you to specify whether the Help topic will appear when the control has the focus (**Enter**), the mouse is hovering over the control (**MouseEnter**), or not at all (**None**).

Select Help Topic Dialog Box

Within this dialog box, you can specify the topic to map to the control through the **Table of Contents** tab by simply selecting it, or you can choose the topic's context ID from the **Context IDs** tab. Additionally, you can specify here whether a topic should be shown when: the control gets focus; the mouse hovers over it; both; or neither, in which case, you can show the topic programmatically.



Specifying the Source Help File

No matter who is going to perform the topic mapping, the software developer must specify the Help file that is going to be used in the application before mapping can occur. Either NetHelp or a .chm (compiled HTML Help) can be used by C1DynamicHelp. If using NetHelp, all NetHelp project files must be deployed with your application. If using HTML Help, only the .chm must be included.

Any other types of Help files can be used, but this requires some additional programming; you can use the Provider property and IHelpProvider interface.

> **Note:** In Doc-To-Help, NetHelp project files are stored by default in the NetHelp folder; the compiled HTML Help file is stored in the HTMLHelp folder.

To specify the source Help file to be used:

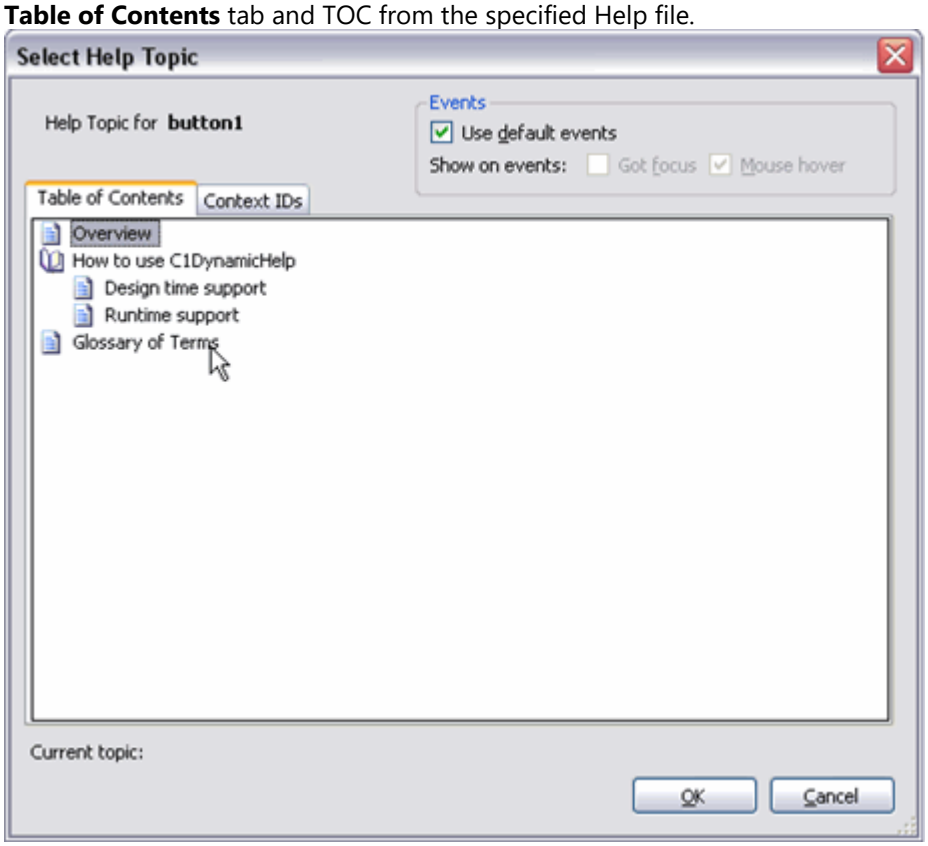
1. From the Toolbox, double-click the C1DynamicHelp control to add it to your form. If you have not already

added the C1DynamicHelp control to the Toolbox, see Adding the C1DynamicHelp Component to a Project.
2. Click the C1DynamicHelp smart tag (▶) to open the **Tasks** menu.
3. Click the **ellipsis** button next to the HelpSource property.
4. Locate and select the desired Help file. You can use any *.chm file or NetHelp *.htm file (use the default.htm or the file name that was used as the base URL for the NetHelp target).
5. Click **Open**.

Mapping a Topic to a Control at Design Time

Once a source Help file is specified, you can easily map topics to controls within your application through C1DynamicHelp.

1. From the Toolbox, double-click the C1DynamicHelp control to add it to your form. An extender property, **Help Topic on C1DynamicHelp**, is added to all controls when C1DynamicHelp is added to the form.
2. Click the C1DynamicHelp smart tag (▶) to open the **Tasks** menu.
3. Click the **ellipsis** button next to the HelpSource property.
4. Locate and select the desired Help file, and click **Open**.
5. Select one of the controls on your form and click the **ellipsis** button next to the **HelpTopic on C1DynamicHelp** property in the Properties window. The **Select Help Topic** dialog box appears and shows the **Table of Contents** tab and TOC from the specified Help file.



By default, the **Use default events** check box is selected (the selected options will vary by the control you've chosen to map to). **Got focus** sets the Help topic to display when the control has been selected; **Mouse hover** sets the Help topic to display when the control has been hovered over. To change the default options, clear the **Use default events** check box and select either the **Got focus** or **Mouse hover** check box.

- **If mapping to the TOC:** From the **Select Help Topic** dialog box, **Table of Contents** tab, choose a topic.
- **If mapping to the context ID list:** From the **Select Help Topic** dialog box, **Context ID** tab, choose the context ID/Topic pairing.

> **Note:** In order to map to a topic, the topic must be in the Help TOC or have a context ID assigned to it; otherwise, it will not appear in the **Select Help Topic** dialog box. Also, if the Help author renames one of the TOC items, the mapping will break. You must get the updated Help and re-create the mapping with the updated TOC.

6. Click **OK** to close the **Select Help Topic** dialog box. The topic is mapped to the control and appears in the C1DynamicHelp Help window at run time when a user clicks or mouses over the control.

Preparing C1DynamicHelp for Authoring Mode

To allow Help authors to activate and deactivate C1DynamicHelp **authoring mode** so that they can map topics in your application, you have a wide variety of options. You can use anything from a special keystroke combination to a registry node that, if present, activates authoring mode.

This example shows how to specify a keystroke combination.

1. Specify a source Help file for C1DynamicHelp.
2. Select your **Form** and set the **KeyPreview** property to **True** in the Properties window.
3. Select **View** | **Code** so you can add code to override the **OnKeyDown** method and specify a keystroke combination to activate and deactivate **authoring mode**.
4. Add the following code:

To write code in Visual Basic

Visual Basic

```
' toggle authoring mode when the user hits Ctrl+Shift+A
Protected Overrides Sub OnKeyDown(ByVal e As System.Windows.Forms.KeyEventArgs)
    If (e.KeyCode = Keys.A And e.Control And e.Shift) Then
        C1DynamicHelp1.AuthoringMode = Not C1DynamicHelp1.AuthoringMode
    End If
    MyBase.OnKeyDown(e)
End Sub
```

To write code in C#

C#

```
// toggle authoring mode when the user hits Ctrl+Shift+A
override protected void OnKeyDown(KeyEventArgs e)
{
    if (e.KeyCode == Keys.A && e.Control && e.Shift)
    {
        c1DynamicHelp1.AuthoringMode = !c1DynamicHelp1.AuthoringMode;
    }
    base.OnKeyDown(e);
}
```

C1DynamicHelp **authoring mode** will be activated or deactivated when the user presses the **Ctrl**+**Shift**+**A** keys.

For Help Authors: Mapping in Authoring Mode

Topic mapping is generally performed by the application Help author or Information Developer; for that reason, the roles of the Information Developer and Software Developer in the process are clearly defined here, although one person may perform both roles.

The files used for mapping include:

- NetHelp project files or .chm (compiled HTML Help)

  The Help file format is specified by the software developer. If using NetHelp, all of the files in the output folder of your Help project must be deployed with the application. If using HTML Help, only the .chm must be included.

  > **Note:** In Doc-To-Help, NetHelp project files are stored by default in the NetHelp folder; the compiled HTML Help file is stored in the HTMLHelp folder.

- Default.htm.xml or HelpFileName.chm.xml

  When you create mappings, they are saved to the **default.htm.xml** or **HelpFileName.chm.xml** located in the same directory as your source Help file, by default. This XML is used to show the topics when the application runs.

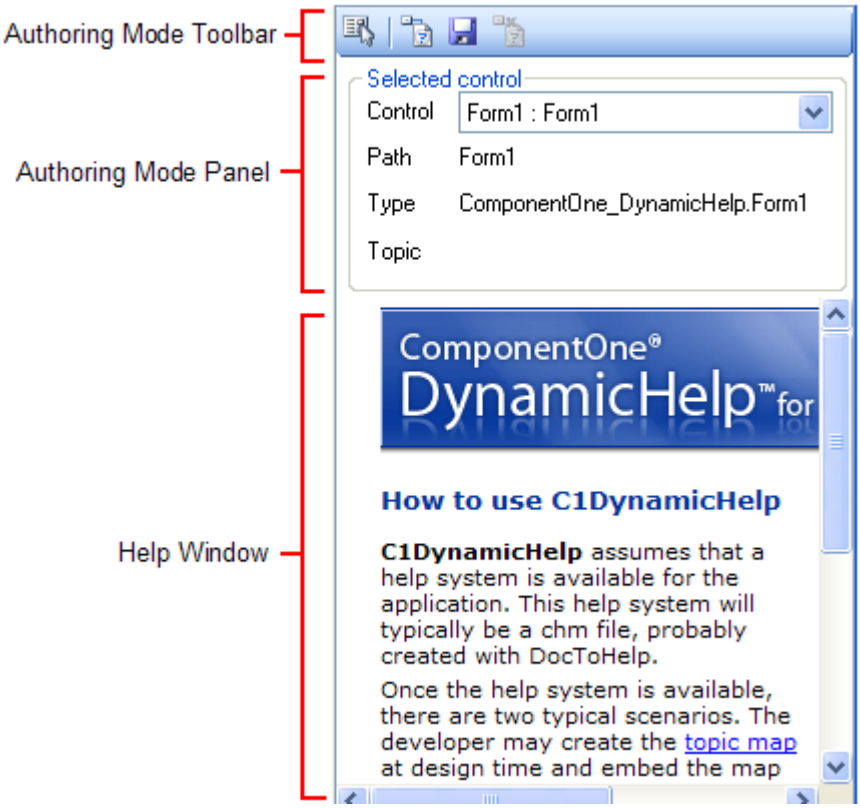
You cannot begin mapping until you place your Help into the correct folder. NetHelp or .chm files should be copied to the install folder specified by the software developer for your application (for example: \\program files\componentone\DocToHelp\Help).

When you have finished mapping topics, the .chm or NetHelp project files and the **.xml** containing the topic mappings must be given to the software developer.

> **Note:** The **.xml** mapping file should never be edited manually. If mappings need to be deleted, use the **C1DynamicHelp authoring mode** (see below).

C1DynamicHelp Authoring Mode

DynamicHelp for WinForms provides a special run-time mode, **authoring mode**, for Help authors to use for assigning topics to controls within an application. When activated, the authoring mode panel and toolbar appear.



Information about the control being mapped to is provided in the **Selected control** area of the panel, which is made up of the following fields:

Control	Control name : type of control.
Path	The path of the control relative to its placement on the form. For example, a Node within a TreeView control may have a path that looks like this: Form1\TreeView1\Node0.
Type	Type of control.
Topic	Topic to associate with the control.

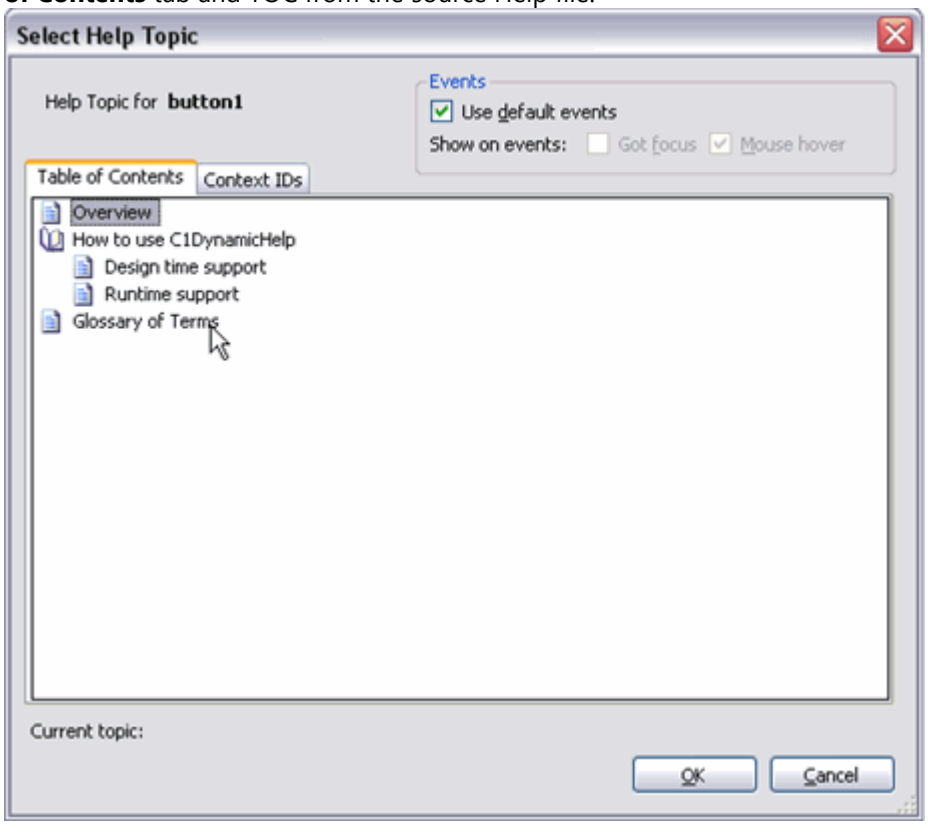
The software developer controls how you to activate and deactivate **authoring mode**.

Mapping a Topic to a Control in Authoring Mode

To map a control to a topic using the **C1DynamicHelp authoring mode**, use the following steps:

1. Display the area of the application's interface that you would like to map. For example, if you are going to map ribbons, make sure they are all available. If you are going to map windows, make sure they are all open.
2. Activate **authoring mode** using the method specified by the software developer. The authoring mode panel appears.
3. Click the **Select control** button  .
4. Click the item you would like to map a Help topic to. As you move the mouse over the controls, you will see information about them in the authoring mode panel: **Control** name, **Path**, control **Type**, and associated **Topic**, if applicable.
5. Click the **Attach topic to control** button  . The **Select Help Topic** dialog box appears and shows the **Table**

of Contents tab and TOC from the source Help file.



By default, the **Use default events** check box is selected (the selected options will vary by the control you've chosen to map to). **Got focus** sets the Help topic to display when the control has been selected; **Mouse hover** sets the Help topic to display when the control has been hovered over. To change the default options, clear the **Use default events** check box and select either the **Got focus** or **Mouse hover** check box.

- **If mapping to the TOC:** From the **Select Help Topic** dialog box, **Table of Contents** tab, choose a topic. Click **OK**. The topic chosen will display in the **Topic** field of the authoring mode panel.
- **If mapping to the context ID list:** From the **Select Help Topic** dialog box, **Context ID** tab, choose the context ID/Topic pairing. Click **OK**. The topic chosen will display in the **Topic** field of the authoring mode panel.

> **Note:** In order to map to a topic, the topic must be in the Help TOC or have a context ID assigned to it; otherwise, it will not appear in the **Select Help Topic** dialog box. Also, if you rename one of the TOC items, the mapping will break. Simply drop the updated Help into the proper folder and re-create the mapping with the updated TOC.

6. Click the **Save** button .
7. Continue mapping.
8. To close the window, use the method specified by the software developer.
9. Deliver the **.xml** mapping file to the software developer.

> **Note:** After mapping, you should backup the **.xml** mapping file elsewhere on your machine. If you uninstall or reinstall your software product, the **.xml** mapping file will be deleted and replaced.

Removing the Topic Mapping for a Selected Control

To remove the topic mapping for a selected control, use the following steps:

1. Activate **authoring mode** using the method specified by the software developer. The authoring mode panel

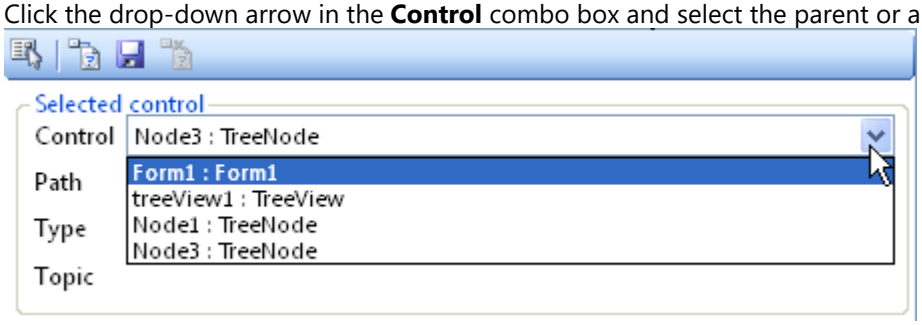
appears.

2. Click the **Select Control** button and select the control that you would like to remove the topic mapping from. The control name appears in the **Control** field of the authoring mode panel, so you can make sure the correct control has been selected.
3. Click the **Detach Topic from Control** button. The topic mapping is removed. Notice there is no longer a topic specified in the **Topic** field.
4. Click the **Save** button.
5. Close the mapping window and deliver the proper Help and **.xml** mapping file to the software developer.

Assign a Topic to a Parent Control

There may be times you need to attach a topic not to a control you can select, but to its parent or ancestor. Since the parent can be completely covered by child controls and thus unselectable, there is a special provision to do it in authoring mode.

1. Activate **authoring mode** using the method specified by the software developer. The authoring mode panel appears.
2. Click the **Select control** button.
3. Click the child control of the parent or ancestor control you would like to map a Help topic to.
4. Click the drop-down arrow in the **Control** combo box and select the parent or ancestor to map to.



5. Click the **Attach topic to control** button. The **Select Help Topic** dialog box appears and shows the **Table of Contents** tab and TOC from the source Help file.
6. Select the topic from the **Table of Contents** tab or select the context ID/Topic pairing on the **Context IDs** tab, and specify any desired events.
7. Click the **Save** button.
8. Close the mapping window and deliver the proper Help and **.xml** mapping file to the software developer.

Advanced Features

DynamicHelp for WinForms requires almost no programming to be used in most cases, but sometimes you may want to use the best of it. The following advanced features are described in this topic:

- **Displaying any Help format**

  C1DynamicHelp allows you to use different help formats. This topic describes two built-in help providers, ChmProvider and NetHelpProvider, and how to implement your own help provider to display a help file in the format you want.

- **Mapping to part of a control or custom controls**

  C1DynamicHelp gives you the ability to map a topic not only to standard controls but also to parts of controls or to custom controls. C1DynamicHelp provides this feature for standard .Net controls derived from the

System.Windows.Forms.Control class. But if you want to extend it to your needs, you can create your own UI element resolver class.

Displaying any Help Format

You may want to use different help formats in your application:

- HTML help;
- NetHelp;
- MS Help 2.0;
- WinHelp;
- RoboHelp WebHelp;
- and so on.

C1DynamicHelp provides means for displaying these or any other help formats.

HTML Help and NetHelp formats can be displayed by the C1DynamicHelp control without any additional programming. There are two built-in help providers in C1DynamicHelp, ChmProvider and NetHelpProvider, which are used automatically when the HelpSource property points to a .chm or an .htm/html file, correspondingly.

Any other types of help files can be used, but this requires some additional programming. In this case you will have to create your own help provider class that implements the IHelpProvider interface. This interface provides all necessary information about a help format to the C1DynamicHelp control. So any help provider can read data from a help source and provides methods to get help topics, to get context IDs, to open help in an external window, to get a topic URL that can be displayed by the C1DynamicHelp control, and so on. After creating your own help provider, all you need is to set it to the Provider property.

For more details see:

- the IHelpProvider interface, which must be implemented by every help provider;
- the Provider property;
- the NetHelpProvider sample, which is the full code actually used in the C1DynamicHelp control.

Mapping to Control Parts or Custom Controls

Mapping to Control Parts or Custom Controls

C1DynamicHelp allows you to map a help topic to any control derived from the System.Windows.Forms.Control class. In most cases it will be enough, but sometimes you may want to map a topic not to a control itself but to some part of it or map a topic to a custom control which is not derived from the System.Windows.Forms.Control. We will refer to these parts of controls and the custom controls as UI elements. To inform the C1DynamicHelp control how to handle UI elements, you will need to create your own class derived from the UIElementResolver class and set it as the Resolver property.

You will need to create a class derived from the UIElementResolver only if you are using custom controls that cannot be handled by the C1DynamicHelp control automatically and then only if you need to associate help topics with parts (UI elements) of those controls, not with the controls themselves.

You have no need to create objects of UIElementResolver type, it is sufficient to define a class derived from UIElementResolver and override its virtual methods. These overridden methods must provide necessary information about UI elements inside custom controls used in your application: methods to find UI elements inside a control by name, coordinates, etc., and other methods necessary for associating dynamic help to UI elements inside a control.

C1DynamicHelp automatically handles most popular, standard controls. For example, it allows mapping a topic to all standard .Net controls and to their parts. So, for example, you can associate a help topic to a separate node of the standard **TreeView** control or to a separate **ListViewItem** of the **ListView** control; that you can do without creating

your own UIElementResolver class.

For more details see:

- the UIElementResolver class, from which you derive your own UI element resolver class;
- the Resolver property;
- the **UsingUIElementResolver** sample, which shows how to create custom UI element resolver for the **C1Ribbon** control to allow mapping topics to items inside **C1Ribbon** such as tabs, buttons, groups, etc.

DynamicHelp for WinForms Tutorials

The following tutorials provide step-by-step instructions on mapping Help topics to controls within an application. Tutorial 1 explains how to map Help topics at design time within Visual Studio. This is usually done by a software developer. Tutorial 2 explains how software developers can set up C1DynamicHelp **authoring mode** and how Help authors can then use it to assign topics to controls on a form, without having to manually edit a topic map.

Tutorial 1: Mapping Help Topics at Design Time

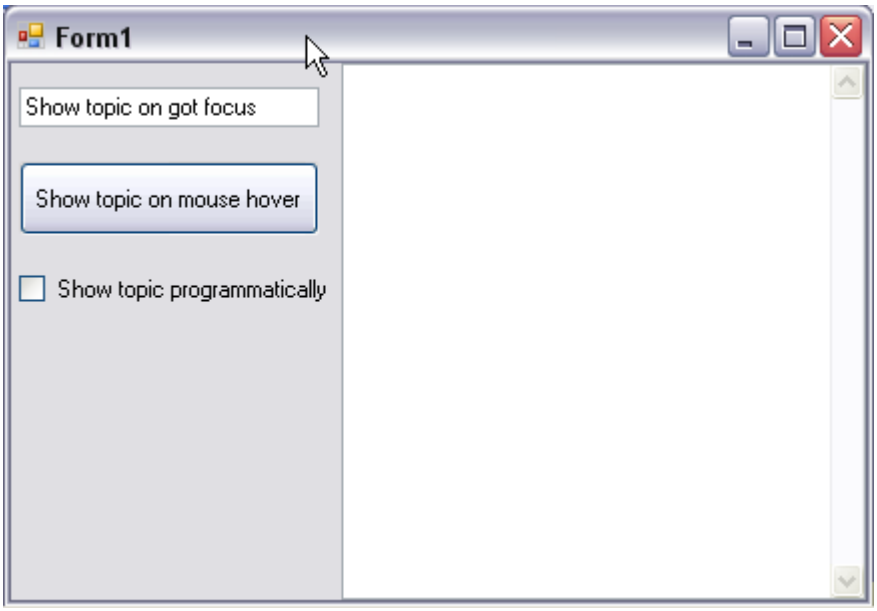
This tutorial explains how to use C1DynamicHelp to map Help topics to controls on a form at design time within Visual Studio. This is usually done by a software developer.

Step 1 of 6: Add controls to the Windows form

In this step, you will add the C1DynamicHelp control to your form, along with a **TextBox**, **Button**, and **Checkbox**.

1. Create a .NET project and add the C1DynamicHelp control to the Toolbox. An extender property, **Help Topic on C1DynamicHelp**, is added to all controls when C1DynamicHelp is added to the form.
2. From the Toolbox, double-click the C1DynamicHelp control. It docks at the right of your form.
3. Add a **TextBox** control to the form:
    1. From the Toolbox, double-click the **TextBox** control to add it to your form.
    2. From the Properties window, set the **textBox1.Text** property to **Show topic on got focus**.
4. Add a **Button** control to the form:
    1. From the Toolbox, double-click the **Button** control to add it to your form.
    2. From the Properties window, set the **button1.Text** property to **Show topic on mouse hover**.
5. Add a **CheckBox** control to the form:
    1. From the Toolbox, double-click the **CheckBox** control to add it to your form.
    2. From the Properties window, set the **checkBox1.Text** property to **Show topic programmatically**.

You have successfully added the controls to your form, which should look similar to the following:



In the next step you will set up the C1DynamicHelp control.

Step 2 of 6: Set up the C1DynamicHelp control

In order to associate topics from your Help file with form controls, you must specify the source Help file.

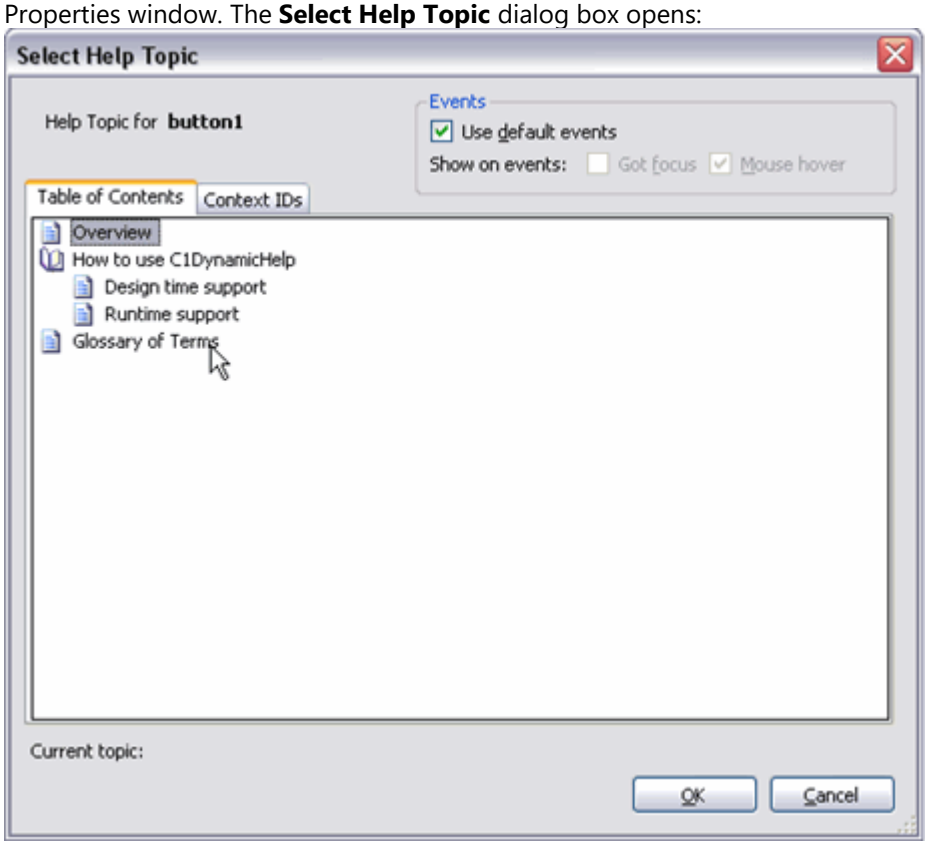
You can do this from the **C1DynamicHelp Tasks** menu by setting the HelpSource property to any *.chm file or NetHelp *.htm file (use the default.htm or the file name that was used as the base URL for the NetHelp target.). In this example, we will use the **C1Sample.chm** installed with **DynamicHelp for WinForms**, by default.

1. Click the C1DynamicHelp smart tag (▶) to open the **Tasks** menu.
2. Click the **ellipsis** button next to the HelpSource property.
3. Locate and select the **C1Sample.chm**. It is located in the Tutorials/Data folder, by default.
4. Click **Open**.

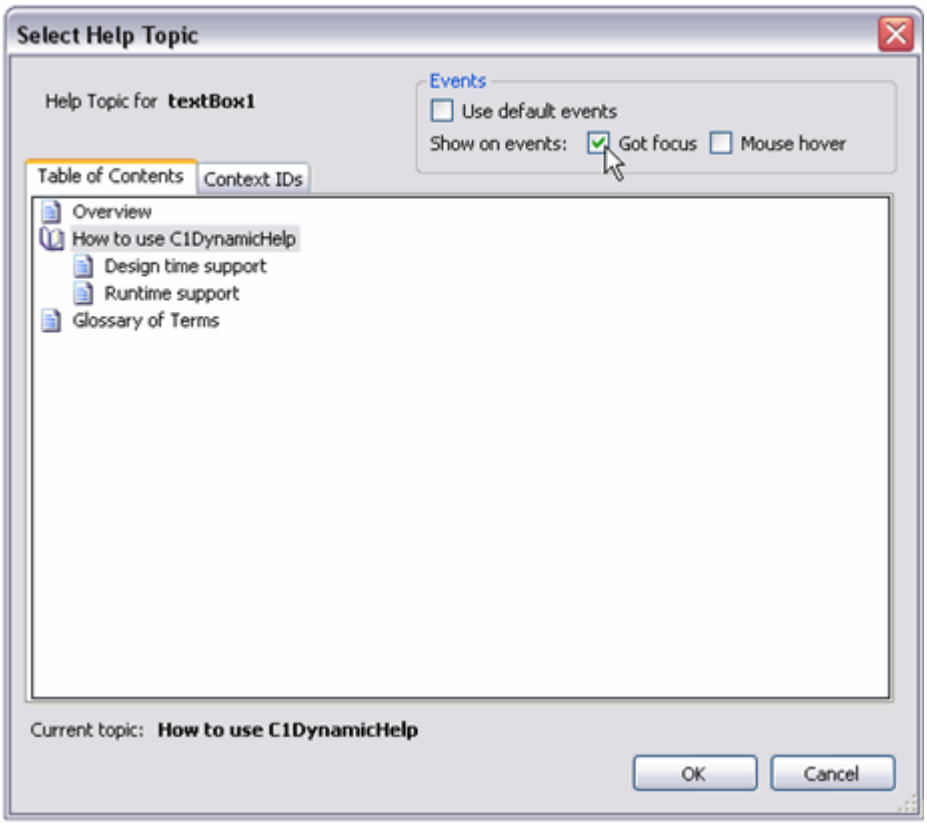
Step 3 of 6: Associate topics with controls on the form

You can associate Help topics with controls using the **Select Help Topic** dialog box.

1. Select **Button1** and click the ellipsis button next to the **HelpTopic on C1DynamicHelp1** property in the Properties window. The **Select Help Topic** dialog box opens:



2. Select any help topic from the **Table of Contents** tab and click **OK**. In this example, we will select the **Overview** topic.
3. Select **TextBox1** and set its **HelpTopic on C1DynamicHelp1** property to any topic from the **Table of Contents** tab.
4. In the **Events** group of the **Select Help Topic** dialog box, uncheck **Use default events** and check the **Got focus** checkbox next to **Show on events**.



Unchecking **Use default events** allows you to specify the events on a per-control basis. Otherwise, the DefaultTrigger property is used for all controls. Using the **Events** settings makes showing topics automatic when a control obtains focus or the mouse hovers over it, or in both cases. But if you want manual control over what triggers topic display, you can uncheck both check boxes (or set DefaultTrigger=None; this will do it for all controls if it's not overridden on a per-control basis) and show topics programmatically.

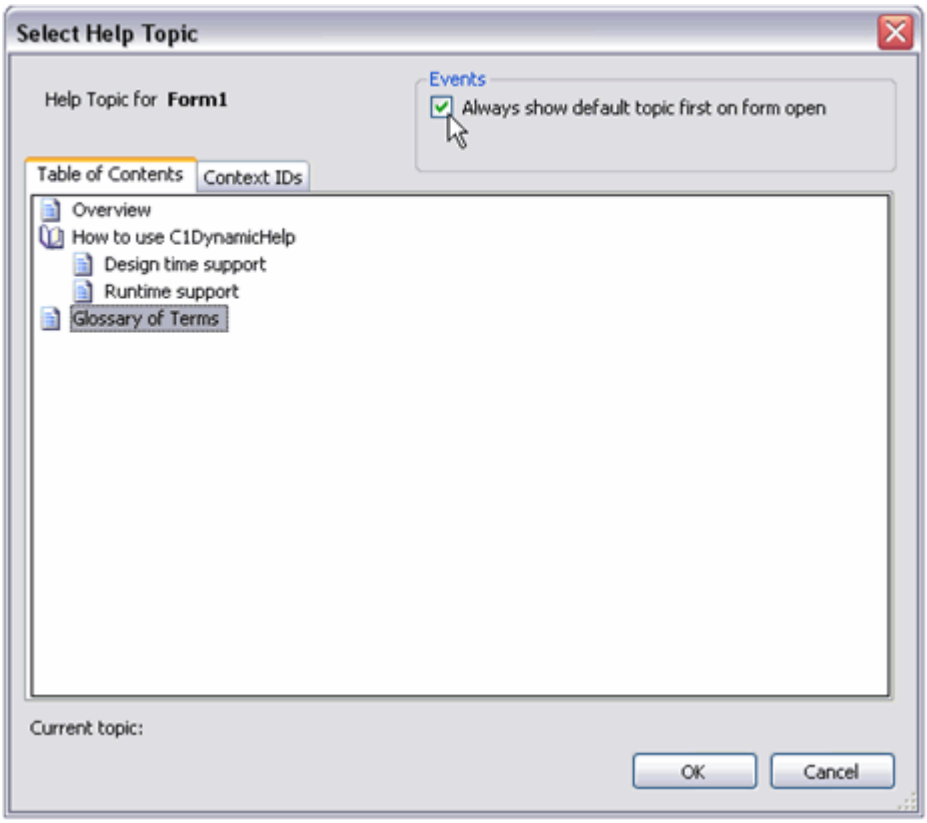
5. Click **OK**.

Step 4 of 6: Associate a topic with the form

You can also use the **Select Help Topic** dialog box to associate a topic with your form.

1. Select **Form1** and click the **ellipsis** button next to the **HelpTopic on C1DynamicHelp1** property in the Properties window. The **Select Help Topic** dialog box appears, but notice the **Events** group is different; in this case, it has one check box **Always show default topic first on form open**.

   The topic associated with the form is the default topic. It is shown when the control that displays the current topic due to a focus/mouse hover event loses focus/mouse hover. If the check box **Always show default topic first on form open** is checked, the default topic is also shown when the form is opened.



2. Select the default topic from the **Table of Contents** tab.

Step 5 of 6: Show topics programmatically

You can show a specific topic programmatically by using the ShowTopic method.

Create a **CheckedChanged** event handler for **CheckBox1** and add code so it looks like the following:

To write code in Visual Basic

Visual Basic

```
Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CheckBox1.CheckedChanged
      C1DynamicHelp1.ShowTopic("WordDocuments/glossaryofterms.htm")
End Sub
```

To write code in C#

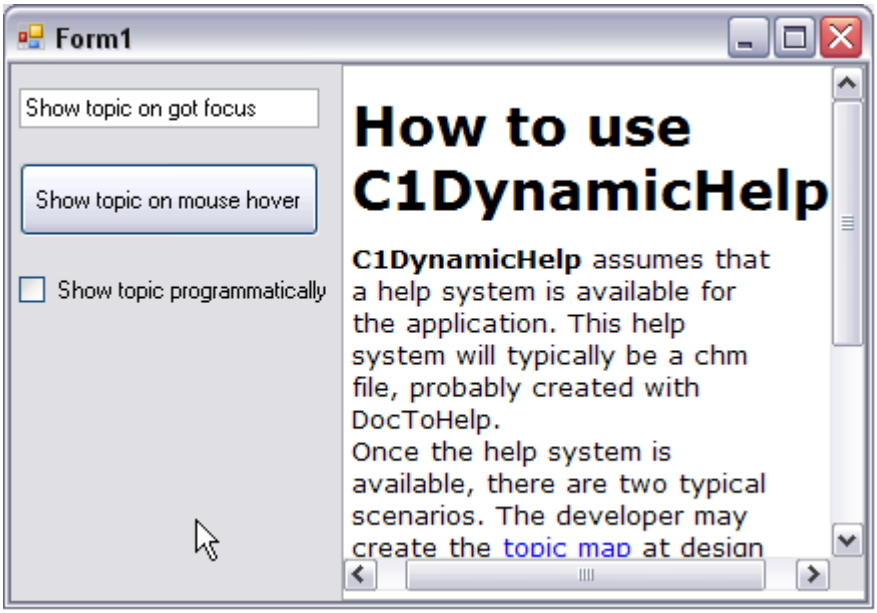
C#

```
private void checkBox1_CheckedChanged(object sender, System.EventArgs e)
{
    c1DynamicHelp1.ShowTopic("WordDocuments/glossaryofterms.htm");
}
```

Step 6 of 6: Run the application

To run the application, click the **Start Debugging** button and notice the following results:

- When Form1 is loaded, the default topic is shown.
- When the mouse is over the button, the topic is displayed.
- Check or uncheck **Show topic programmatically** to display its associated topic.
- When **TextBox1** gets the focus, it displays the specified topic.



Congratulations, you have successfully completed Tutorial 1! In this tutorial you have learned how to:

- Add and set up the C1DynamicHelp control.
- Associate topics with controls on a form at design-time.
- Show topics programmatically.

Tutorial 2: Mapping Help Topics in Authoring Mode

This tutorial shows the second method of assigning Help topics to controls on a form. It is done entirely by Help authors; once a software developer sets up his application to use the C1DynamicHelp **authoring mode**, he doesn't need to do anything to map topics to controls! This eliminates the error-prone process of passing the topic/control map back and forth between Help authors and software developers.

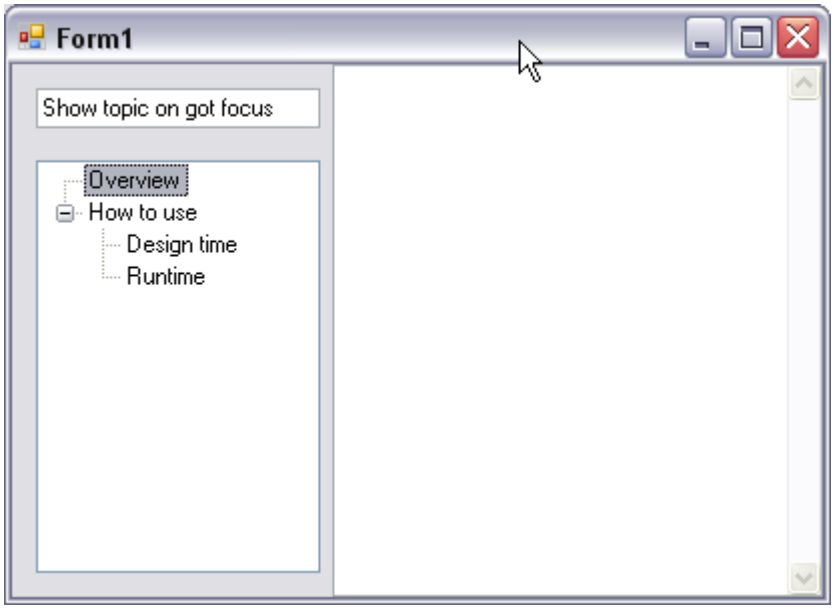
The first two steps of this tutorial explain how software developers can set up their applications for **authoring mode**. Step 3 explains how Help authors can assign topics to controls in **authoring mode**. The final step shows how the Help appears in the application.

Step 1 of 4: Add controls to the Windows form

To set up your new Form with the C1DynamicHelp control, a **TextBox**, and a **TreeView** control, complete the following steps. These steps should be performed by the software developer.

1. Create a .NET project and add the C1DynamicHelp control to the Toolbox.
2. From the Toolbox, double-click the C1DynamicHelp control. It docks at the right of your form.
3. Add a **TextBox** control to the form:
    1. From the Toolbox, double-click the **TextBox** control to add it to your form.
    2. From the Properties window, set the **textBox1.Text** property to **Show topic on got focus**.
4. Add the **TreeView** control to the form:
    1. From the Toolbox, double-click the **TreeView** control to add it to your form.

2. From the Properties window, add nodes to the **treeView1** according to the following image:



In the next step you will set up the C1DynamicHelp control.

Step 2 of 4: Set up the C1DynamicHelp control

In order to associate topics from your Help file with form controls, you must specify the source Help file.

You can do this from the **C1DynamicHelp Tasks** menu by setting the HelpSource property to any *.chm file or NetHelp *.htm file (use the default.htm or the file name that was used as the base URL for the NetHelp target.). In this example, we will use the **C1Sample.chm** installed with DynamicHelp for WinForms, by default.

Once a source Help file is specified, you can prepare the C1DynamicHelp control to be used in **authoring mode**. This step should be performed by the developer.

1. Click the C1DynamicHelp smart tag (▶) to open the **Tasks** menu.
2. Click the **ellipsis** button next to the HelpSource property.
3. Locate and select the **C1Sample.chm**. It is located in the Tutorials/Data folder, by default.
4. Prepare the C1DynamicHelp control for using in **authoring mode**:
    1. From the Properties window, set the **Form1.KeyPreview** property to **True**.
    2. Override the **OnKeyDown** method by adding the following code:

To write code in Visual Basic

Visual Basic

```
' toggle authoring mode when the user hits Ctrl+Shift+A
Protected Overrides Sub OnKeyDown(ByVal e As
System.Windows.Forms.KeyEventArgs)
    If (e.KeyCode = Keys.A And e.Control And e.Shift) Then
        C1DynamicHelp1.AuthoringMode = Not C1DynamicHelp1.AuthoringMode
    End If
    MyBase.OnKeyDown(e)
End Sub
```

To write code in C#

C#

```
// toggle authoring mode when the user hits Ctrl+Shift+A
override protected void OnKeyDown(KeyEventArgs e)
{
    if (e.KeyCode == Keys.A && e.Control && e.Shift)
    {
        c1DynamicHelp1.AuthoringMode = !c1DynamicHelp1.AuthoringMode;
    }
    base.OnKeyDown(e);
}
```

3. Create a handler for the **Form_Load** event and insert the following code to instruct the C1DynamicHelp control to subscribe the controls to the events for showing topics:

To write code in Visual Basic

Visual Basic

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    C1DynamicHelp1.TopicMap.Refresh()
End Sub
```

To write code in C#

C#

```
private void Form1_Load(object sender, EventArgs e)
{
    c1DynamicHelp1.TopicMap.Refresh();
}
```

5. Create a handler for the **Form_Closing** event and insert the following code to ask the user whether to save changes made in the topic map:

To write code in Visual Basic

Visual Basic

```
Private Sub Form1_FormClosing(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles MyBase.FormClosing
    If (C1DynamicHelp1.TopicMap.HasChanges) Then
        Dim dr As DialogResult
        dr = MessageBox.Show("Would you like to save the changes you made to
control/topic map?", "C1DynamicHelp Tutorial", MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Question)
        If (dr = DialogResult.Yes) Then
            C1DynamicHelp1.TopicMap.Save()
        ElseIf (dr = DialogResult.Cancel) Then
            e.Cancel = True
        End If
    End If
End Sub
```

To write code in C#

C#

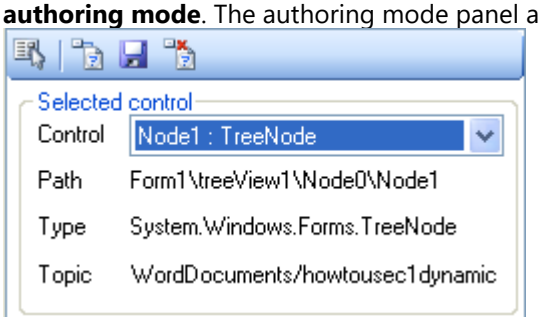
```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (c1DynamicHelp1.TopicMap.HasChanges)
    {
        DialogResult result = MessageBox.Show("Would you like to save the
changes you made to control/topic map?", "C1DynamicHelp Tutorial",
MessageBoxButtons.YesNoCancel , MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
            c1DynamicHelp1.TopicMap.Save();
        else if (result == DialogResult.Cancel)
            e.Cancel = true;
    }
}
```

No controls have topics associated with them yet. The application can now be used by a Help author to map topics to controls.

Step 3 of 4: Using authoring mode

This step explains how the Help author can activate **authoring mode** and associate topics with controls.

1. Run the application from the software developer and press the **Ctrl**+**Shift**+**A** keys together to activate **authoring mode**. The authoring mode panel appears.



2. Map a topic to the **TextBox** control:
    1. Click **Select control** button and select **TextBox1**.
    2. Click the **Attach topic to control** button. The **Select Help Topic** dialog box appears.
    3. On the **Table of Contents** tab, select the *Glossary of Terms* topic.
    4. Uncheck the **Use default events** and **Mouse hover** checkboxes so that only **Got focus** is checked.
    5. Click **OK**. Note that a topic is now displayed only when **TextBox1** gets focus - not when the mouse hovers over it.
3. Map a topic to the **TreeView** nodes:
    1. Click **Select control** button and select the **Overview** node.
    2. Click the **Attach topic to control** button. The **Select Help Topic** dialog box appears.
    3. On the **Table of Contents** tab, select the *Overview* topic and click **OK**.
    4. Assign the *How to use C1DynamicHelp* topic to the **How to Use** node using steps **a**-**c**.
    5. Assign the *Design time support* topic to the **Design time** node using steps **a**-**c**.
    6. Assign the *Runtime support* topic to the **Runtime** node using steps **a**-**c**. Note that the corresponding topics are displayed when the mouse hovers over each of the **TreeView** nodes.

    > **Note:** In **authoring mode** you have the ability to associate topics with objects whose class is not

> derived from **Component**, such as **TreeNode** or **ListBox** items. This cannot be done at design time, because there is no **Component** to set for the **HelpTopic on C1DynamicHelp1** property.

4. Click the **Save control/topic mapping**  button to save the changes to an .xml file that serves as the topic map.

   > **Note:** If you close the application without saving the control/topic mapping, a dialog box will appear, prompting you to save the changes.

   The topic map is saved to the same location as the Help file specified in the HelpSource property. It will also have the same name as the specified Help file, only with an .xml extension. In this example, the topic map can be found in the **C1Sample.chm.xml** file. It should now look like this:

   XML code

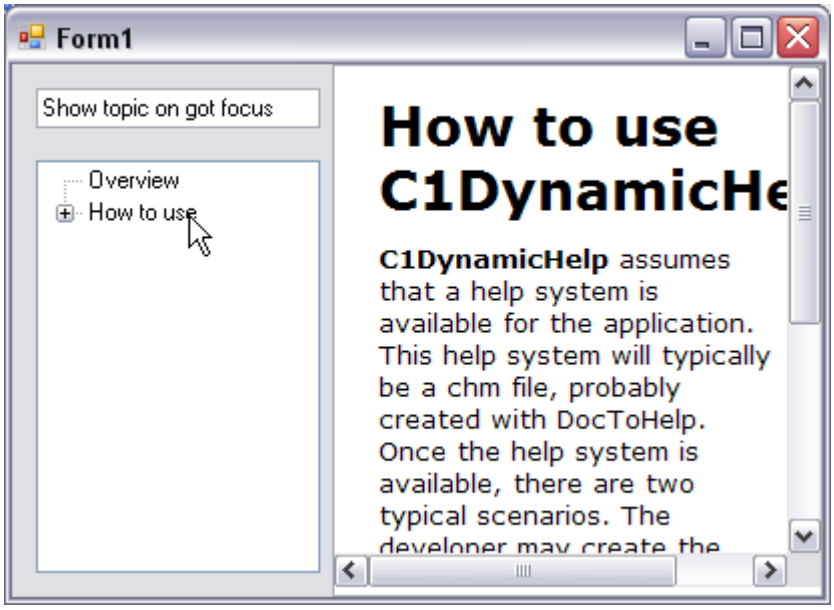
```
<Dictionary>
  <item>
    <key>Form1\textBox1</key>
    <value>WordDocuments/glossaryofterms.htm</value>
    <events useDefaultEvents="False">1</events>
  </item>
  <item>
    <key>Form1\treeView1\Node1</key>
    <value>WordDocuments/howtousec1dynamichelp.htm</value>
    <events useDefaultEvents="True">3</events>
  </item>
  <item>
    <key>Form1\treeView1\Node0</key>
    <value>WordDocuments/overview.htm</value>
    <events useDefaultEvents="True">3</events>
  </item>
  <item>
    <key>Form1\treeView1\Node1\Node2</key>
    <value>WordDocuments/designtimesupport.htm</value>
    <events useDefaultEvents="True">3</events>
  </item>
  <item>
    <key>Form1\treeView1\Node1\Node3</key>
    <value>WordDocuments/runtimesupport.htm</value>
    <events useDefaultEvents="True">3</events>
  </item>
</Dictionary>
```

5. Give the **C1Sample.chm.xml** file, along with the updated Help file(s), if changed, to the software developer.

Step 4 of 4: Run the Application

To run the application, click the **Start Debugging** button and notice the following results. This step should be performed by the developer.

- When the mouse hovers over the **TreeView** nodes, the specified topics are displayed.
- Only when **TextBox1** gets the focus will it display a topic.



Congratulations, you have successfully completed Tutorial 2! In this tutorial you have learned how to:

- Add and set up the C1DynamicHelp control.
- Activate/deactivate **authoring mode**.
- Associate topics to controls on a form in **authoring mode**.
- Save changes in the topic map.