# Demystifying Help

## Part 3

By Kevin S. Gallagher

Help topics do not stand on their own, they need to be *linked* to one another. This is commonly known as *jumps*. In the broad sense of the word, linking is the proper verbiage. Beneath *linking* there are sub-categories:

### Jumps

This is the most common method to link topics together. Usually *jumps* will traverse the user to another help topic using the same help window the current topic is being displayed in. If there is a need for keeping the current topic active when jumping to another topic then place the jump into a secondary window. Try to keep the amount of help windows to a minimum, only use secondary windows when absolutely needed to get helpful information across to the users.

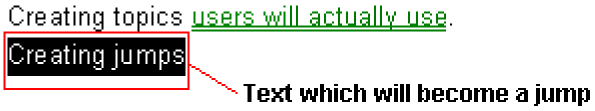
One of the many reasons why applications never had help or limited help came directly from the archaic steps one would need to do to create jumps. Below is what needed to be done to create a simple jump with no third party tool to assist a help author.

1. Select the jump hot-spot text.
2. From the Format menu, choose Character.
3. Select the Double Underline check box, and then choose OK.
4. Position the insertion point immediately after the last letter in the double-underlined jump hot-spot text.
5. From the Format menu, choose Character again.
6. Clear the Double Underline check box, select the Hidden check box, and then choose OK.
7. Type the context string assigned to the topic that is the target of the jump.
8. From the Format menu, choose Character again, clear the Hidden check box, and then choose OK.

The steps laid out here may not seem all that difficult until you are a) short on time b) have a lot of jumps to create. I can remember to this day that debugging took many hours just to insure that a jump worked and also went to the proper topic. Sure you can still create jumps to wrong topics but with the tools available today it is easier to get it right the first time.

Here is how you would create a jump with one of tools available today, although this is done with RoboHelp most competing tools have a similar approach to do the same task.

1. RoboHelp uses Microsoft Word as for an editor simply highlight the text, which will be the text used for jumping to another topic.

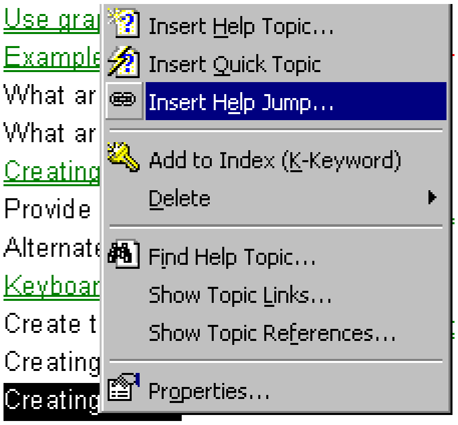


Creating topics users will actually use.

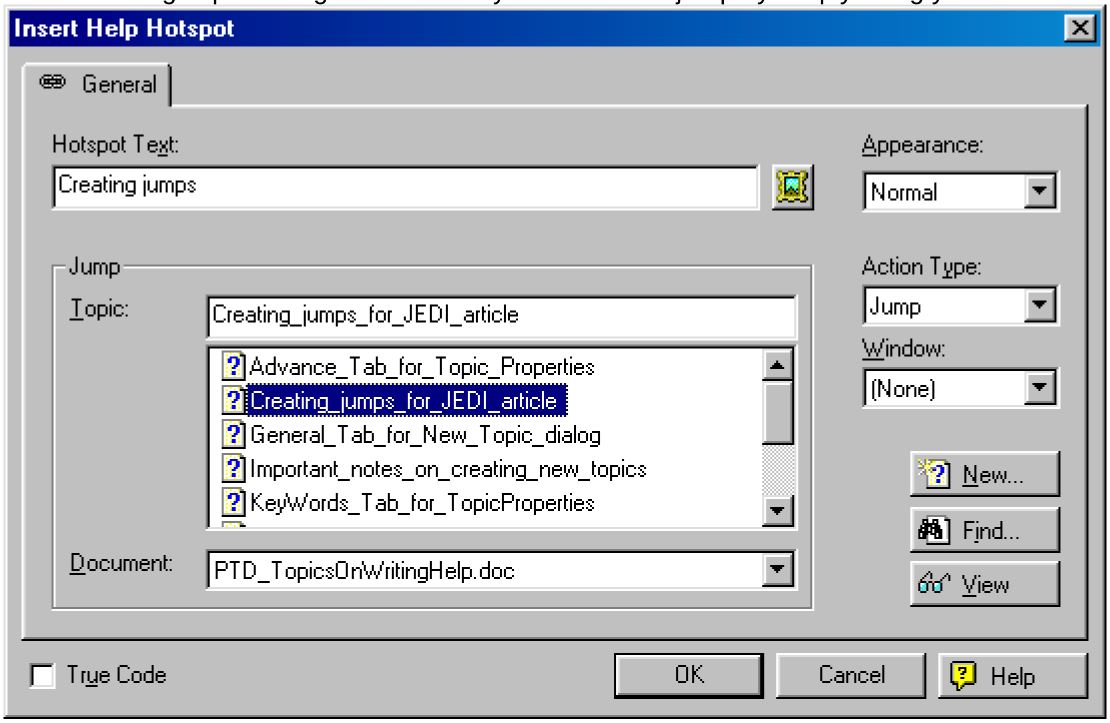
Creating jumps

Text which will become a jump

2. Select “Create new jump” from either pressing a toolbar option, a menu item or a context menu item in MS-Word.



3. This brings up a dialog which allows you to create a jump by simply using your mouse



Hotspot Text: Creating jumps

Appearance: Normal

Jump

Topic: Creating_jumps_for_JEDI_article

Advance_Tab_for_Topic_Properties
Creating_jumps_for_JEDI_article
General_Tab_for_New_Topic_dialog
Important_notes_on_creating_new_topics
KeyWords_Tab_for_TopicProperties

Action Type: Jump

Window: (None)

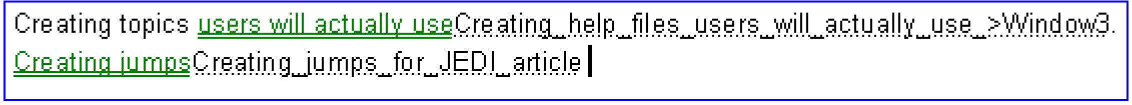
Document: PTD_TopicsOnWritingHelp.doc

□ True Code

The first text box containing "Create jumps" is the text which was highlighted, usually this remains unchange but you can change it if so desire. The combo-boxes provides methods to change the appearance (colorization) and type of jump plus depending of the type of jump, what type of window and which window to place the jump into. A really nice feature is the "View" option, which allows you to view a topic inside of a real help window to make sure that it is the proper topic you would like to jump too.

4. Pressing the OK button creates the link for you and then returns you to MS-Word.

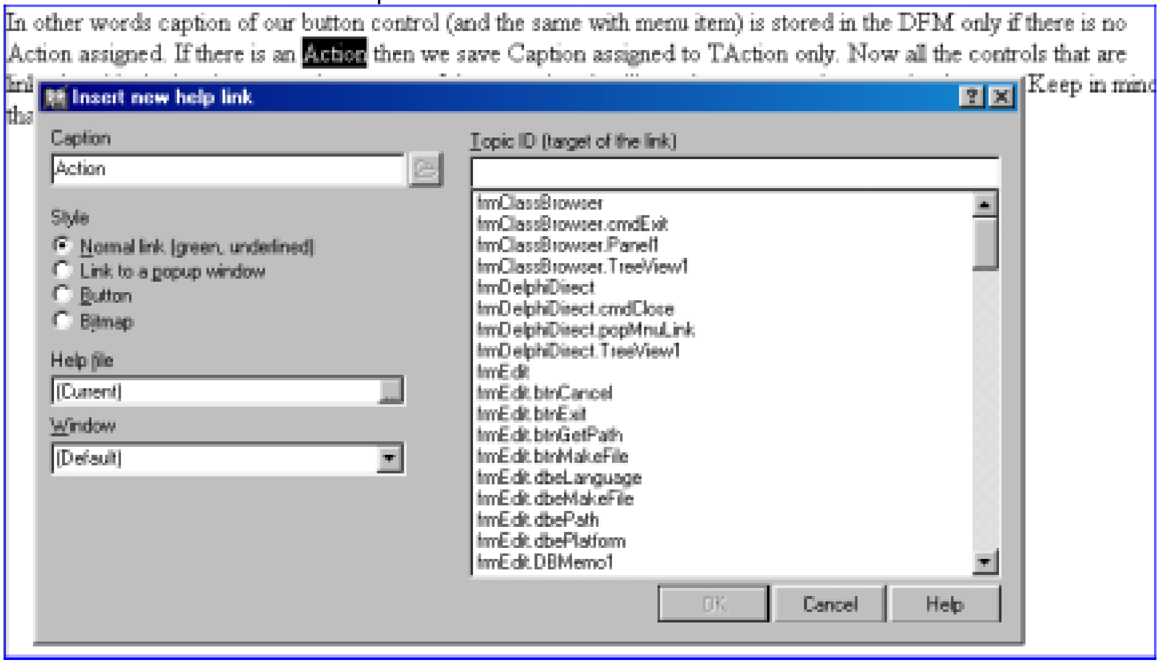
The following is what the completed work looks like with formatting of hidden text turned on.



Creating topics users will actually useCreating_help_files_users_will_actually_use_>Window3.
Creating jumpsCreating_jumps_for_JEDI_article

Normally I do not have hidden text shown (the text with the dotted underline) but have done so here to permit you to see what is needed by the Microsoft help compiler. The double underline attribute of the text which shows the text for the jump is required so that the compiler recognizes the text as a jump, same goes for the hidden text.

Using “Help and Manual” creating a link is similar to RoboHelp, one difference is that “Help and Manual” uses an internal word processor.



In other words caption of our button control (and the same with menu item) is stored in the DFM only if there is no Action assigned. If there is an Action then we save Caption assigned to TAction only. Now all the controls that are lnl Keep in mind the

Caption: Action

● Normal link (green, underlined)
○ Link to a popup window
○ Button
○ Bitmap

Help file: (Current)

Window: (Default)

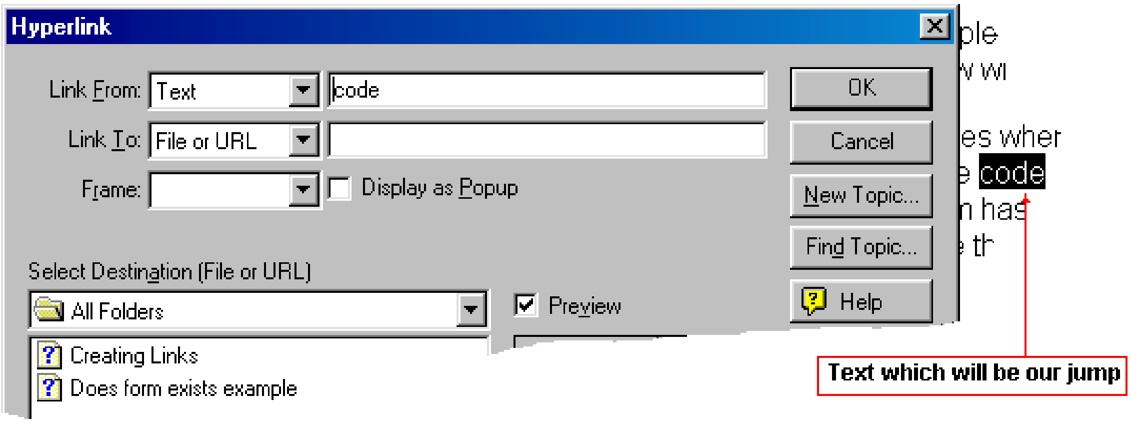
Topic ID (target of the link):

frmClassBrowser
frmClassBrowser.cmdExit
frmClassBrowser.Panel1
frmClassBrowser.TreeView1
frmDelphiDirect
frmDelphiDirect.cmdClose
frmDelphiDirect.popMnuLink
frmDelphiDirect.TreeView1
frmEdit
frmEdit.btnCancel
frmEdit.btnExit
frmEdit.btnGetPath
frmEdit.btnMakeFile
frmEdit.dbeLanguage
frmEdit.dbeMakeFile
frmEdit.dbePath
frmEdit.dbePlatform
frmEdit.DBMemo1

“Help and Manual’s” internal word processor is not as powerful as using Microsoft Word as in RoboHelp but I see no reason that you need what Microsoft Word provides. Remember you are entering simple text for the most part. *I should mention (if I have not done so before) that I use RoboHelp because of my day job were this is the chosen help authoring tool. If I were to select a help-authoring tool today most likely it would not be RoboHelp because of the cost of RoboHelp. It all depends on how much money you have and you must ask yourself, do I need all of the features RoboHelp provides or will a lower price tool get the job done for me.*

- Both WinHelp and the new HTML style help both supports creating jumps to help topics in other help files. Again like creating simple jumps without a third party tool can make this process difficult. Yet another reasons to use a third party tool, everyone I have used provide visual method to create inter-file jumps.

  The following shows an example of creating a jump in an HTML style help file using the HTML version of RoboHelp.



Link From: Text | code

Link To: File or URL |

Frame: | □ Display as Popup

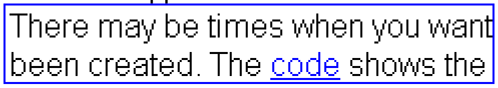
Select Destination (File or URL)

All Folders | ☑ Preview

Creating Links
Does form exists example

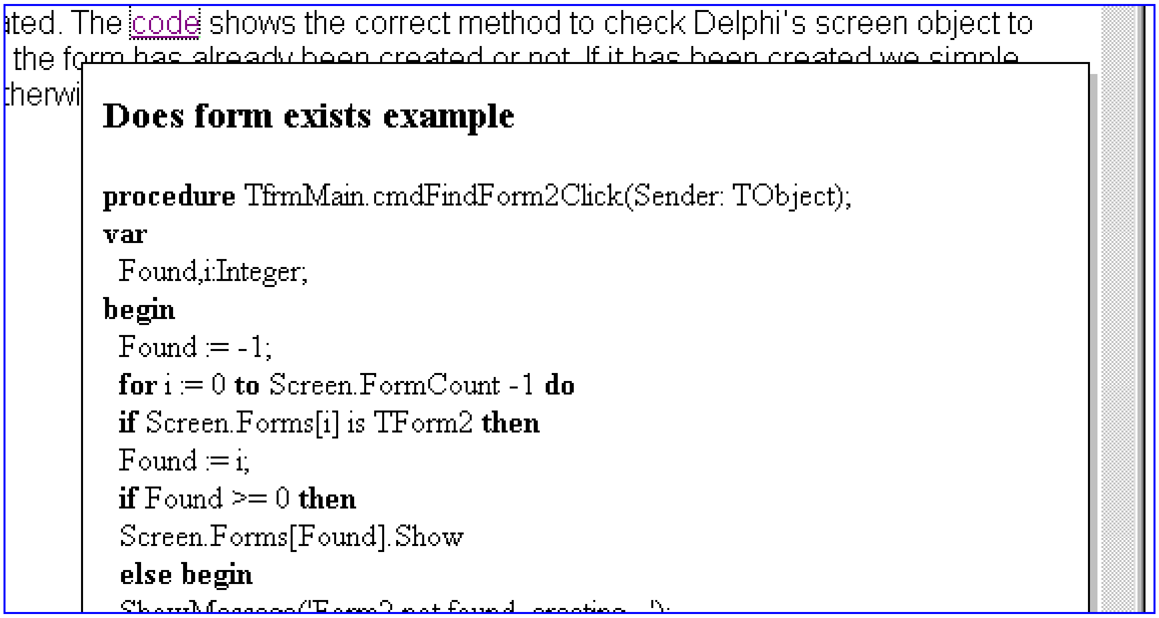
Text which will be our jump

The results appear as follows:



There may be times when you want
been created. The code shows the

The text used for the jump/link is colored in blue. When single clicking the mouse on the jump word the user will see the assigned help topic in the same window. You could also choose to have the topic displayed in another window or as a popup.



ated. The code shows the correct method to check Delphi's screen object to
the form has already been created or not. If it has been created we simple
therwi

**Does form exists example**

```
procedure TfrmMain.cmdFindForm2Click(Sender: TObject);
var
  Found,i:Integer;
begin
  Found := -1;
  for i := 0 to Screen.FormCount -1 do
  if Screen.Forms[i] is TForm2 then
  Found := i;
  if Found >= 0 then
  Screen.Forms[Found].Show
  else begin
  ShowMessage('Form2 not found, creating...');
```

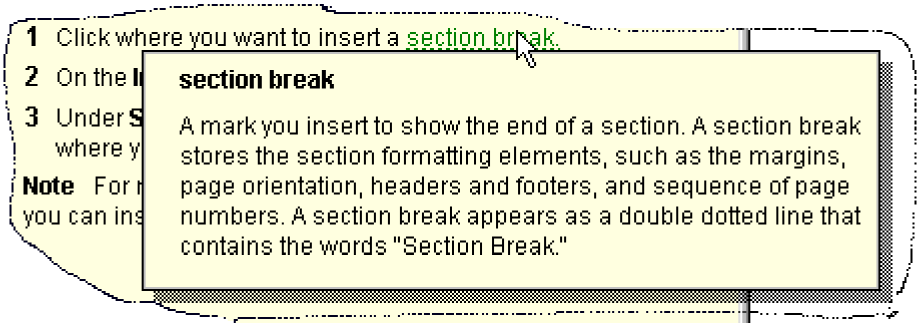
The code to accomplish the above (generated using RoboHTML help-authoring tool).

```
<A HREF="javascript:BSSCPopup('Does_form_exists_example.htm');"
onMouseOver="if (parseInt(navigator.appVersion) >= 4)
BSPSPopupOnMouseOver(event)" class="BSSCPopup">code</A>
```

- You are not limited to using text for jump markers as described above, graphics can also serve as jump points for jumping to other topics.

**Popups**

Pop-up's can be described in two different ways, in the context of *What's this* style of help and as tiny windows hovering over some text in a help topic. For this discussion we are not talking about *What's this*. Pop-up's provide you with a way to display things like definitions of word(s) and phrases. The example below shows a popup explaining "section break" in Microsoft Word.



1 Click where you want to insert a section break.

2 On the I

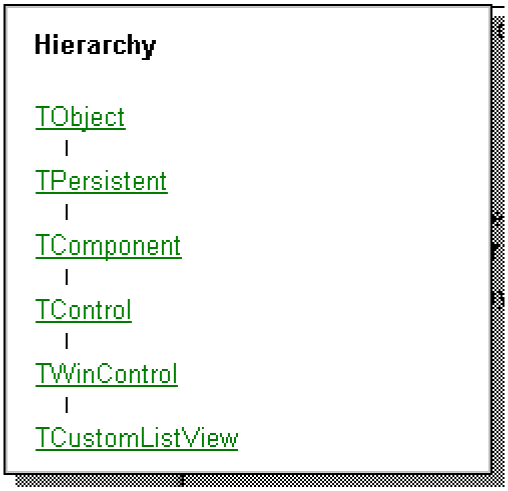
3 Under S where y

**Note** For i you can ins

**section break**

A mark you insert to show the end of a section. A section break stores the section formatting elements, such as the margins, page orientation, headers and footers, and sequence of page numbers. A section break appears as a double dotted line that contains the words "Section Break."

Popup links can have *links* to other topics. For example, Borland does this in Delphi help as shown below

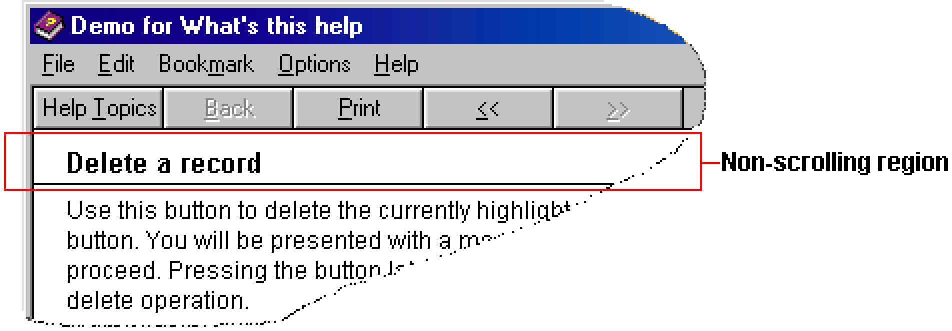


**Hierarchy**

TObject
|
TPersistent
|
TComponent
|
TControl
|
TWinControl
|
TCustomListView

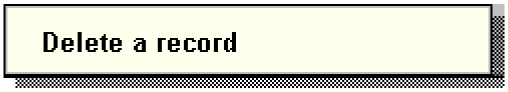
I would suggest only using links within popup's when you are dealing with people you know will not become lose in jumping around. Some users will easily become lost while others will not, you need to test this out with your testers prior to implementing.

Note that when creating popup topics there can be no *non-scrolling region* for that topic, otherwise only the heading/title of the topic will display. Using the following example topic you will see how it shows up as a popup topic using *What's this* help in an application.



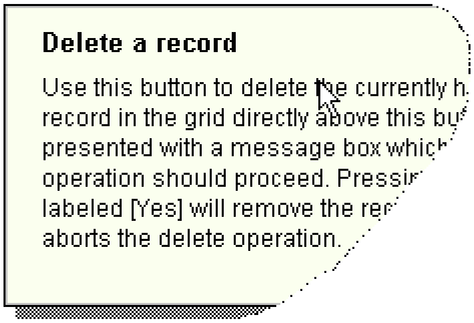
**Delete a record**

Use this button to delete the currently highligh
button. You will be presented with a me
proceed. Pressing the button l
delete operation.



**Delete a record**

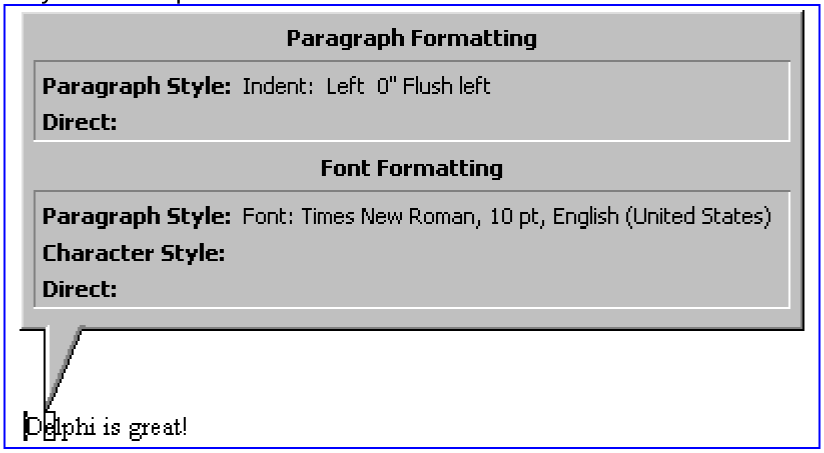
By removing the *non-scrolling region* from the topic you would then see the following rather then simply the heading/title of the topic. The image of the help topic below is now displayed correctly since the *non-scrolling region* has been removed.



Delete a record

Use this button to delete the currently h
record in the grid directly above this bu
presented with a message box which
operation should proceed. Pressir
labeled [Yes] will remove the rec
aborts the delete operation.

If you want to get creative with popup topics you could do something like MS-Word's popup help for formatting of documents (as shown below). This style of help tells the user exactly what item they asked help for.



Paragraph Formatting

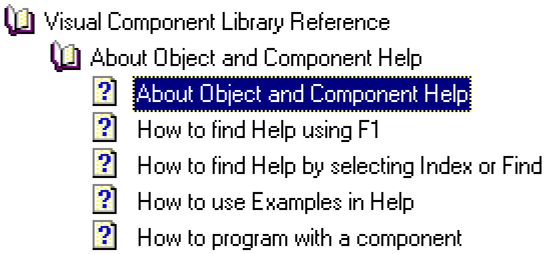
**Paragraph Style:** Indent: Left 0" Flush left
**Direct:**

Font Formatting

**Paragraph Style:** Font: Times New Roman, 10 pt, English (United States)
**Character Style:**
**Direct:**

Delphi is great!

**Browse sequences**
Browse sequences are the two buttons, which appear in many WinHelp files and appear as (Previous << and Next >>). They allow users to safely navigate through related topics. A good example appears in Delphi help as shown below:



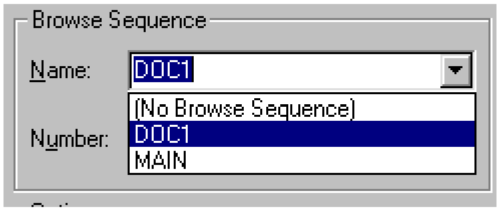
Visual Component Library Reference
About Object and Component Help
About Object and Component Help
How to find Help using F1
How to find Help by selecting Index or Find
How to use Examples in Help
How to program with a component

(The image above is from the Table of Contents)

By clicking on the first topic you get the (Next >>) button, clicking on the last topic and only the (Previous <<) button is available. Any topic within these two topics will show both the Previous and Next buttons.

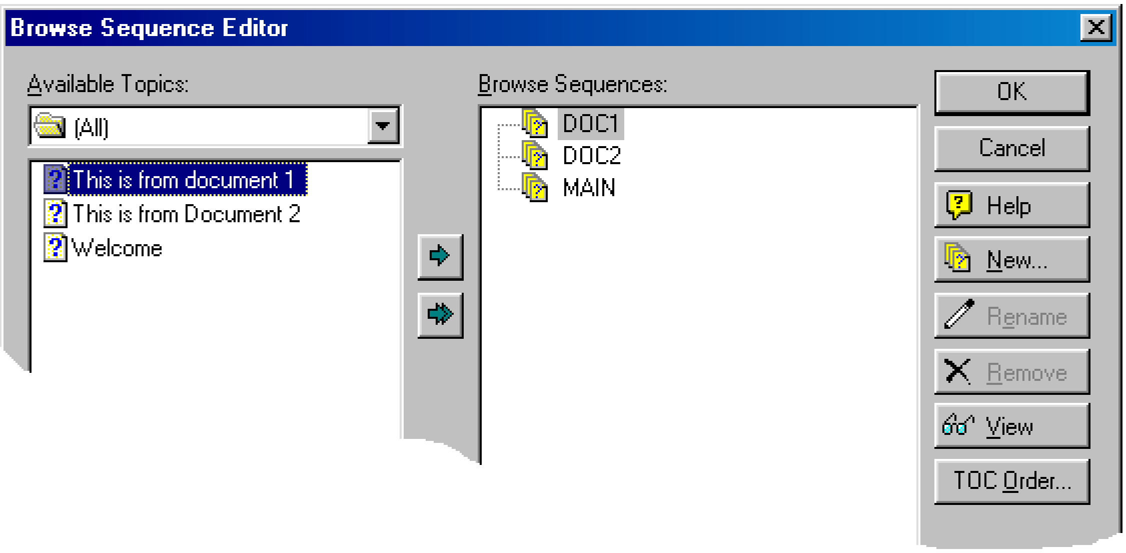
To implement browse sequencing in the old days (and also today if you are not using a help authoring tool) involved creating names for browse sequences for each series and then list the topics and a unique number for each topic, lastly a special footnote needed to be inserted into each topic in a browse sequence. Using modern day help authoring tools is much easier. Below is an example of how RoboHelp makes the task easy.

After creating a new help topic you can place the topic into a browse sequence through the property dialog of the topic, or exclude the topic.

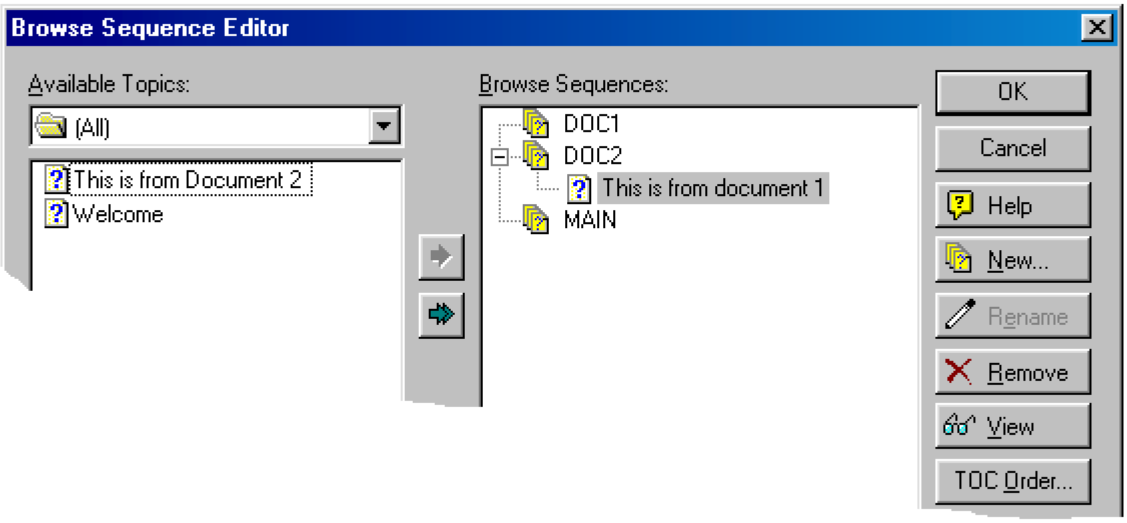


Name: DOC1

If you want, the browse sequencing can be done using RoboHelp's Browse Sequence editor. The following image shows the editor with three browse sequences with no topics assigned to any of them.



Dragging a topic in the left pane to the right pane move the topic to the desired sequence as shown below:



Pressing the OK button at this point places a special footnote into the topic.

Below is the format of a browse sequence footnote, the plus symbol denotes it is a sequence followed by the sequence name (DOC2) and lastly by the sequence number (10).

$^{+}$ DOC2:000010

To be of any use you need at least two topics in the browse sequence. Once there are more then one topic in a sequence they will provide the proper navigation.

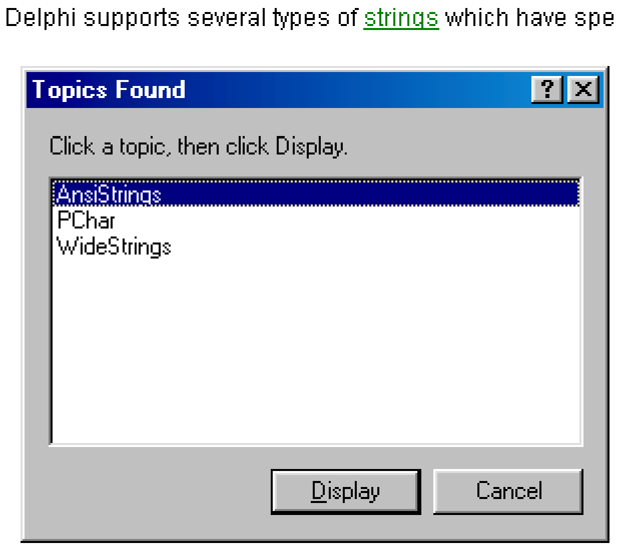
Note that help topics within browse sequences can be spread out through several source documents or any order you wish in one document. Also, the Microsoft help compiler can automatically assign sequences for you by specifying “auto”. If this is done the sequence order is the same as the order the help topics are placed in the source document.

My preferred method for implementing browse sequences is to write each topic, then prior to writing the contents of the topics use the browse sequence editor to assign topics to sequences. After the topics have been written the testers know they need to traverse the sequences to insure they not only work properly but also move to intended topics.

**See-Also**

These are similar to references in paper books, they provide easy access to other help topics. They are implemented using “A-keywords”.  The benefit of *See-also* other jumps and popup links is that rather then simply jumping to one topic you can provide the possibility to jump to other similar topics. The usual area that *see-also* is shown is beneath the a help topics heading

Example of a *See-also* within a help topic



Delphi supports several types of strings which have specific uses.

To implement (using RoboHelp) the above *See-Also* an A-Link was first created via a drag-n-Drop operation then by using a menu item in MS-Word the A-Link just created was assigned to the word “string” in the sentence above. Yet another reason for using a Help authoring tool, without it the process is long and prone to errors.

**Buttons**
Usually buttons are used to get the user from topic to topic. The common buttons are the Index and Find button found in most help files. Buttons placement is not limited to the area below a help windows caption area, they can be placed directly into topics.

Add Print and Exit buttons to the button bar

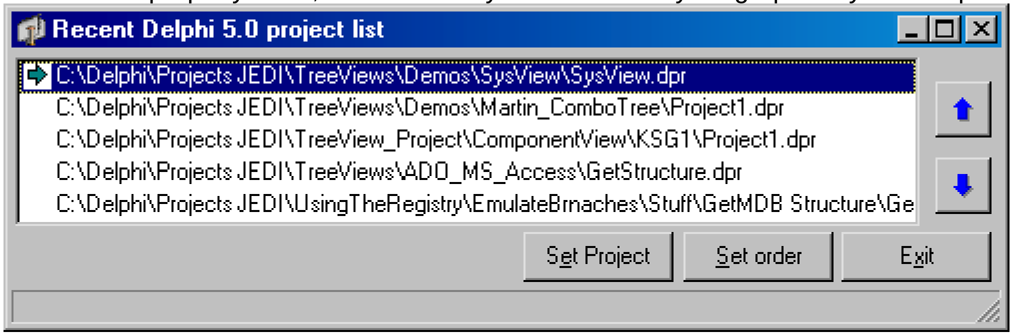
```
CreateButton("ID_PRINT","&Print Topic","Print()")

CreateButton("ID_EXIT","E&xit Help","Exit()")
```

**Hotspot images**
These are images with clickable areas defined with a utility called SHED (Segmented Hypergraphics Editor). An image might have one area across the entire image or multiple areas. When the mouse hovers over a predefined clickable area the mouse cursor changes to a hand. This alerts the user that they can click on the area under the mouse to get help.
By definition, a hotspot image is any image that contains one or more "clickable" areas. When the cursor is over a hotspot area on an image, it turns to a hand indicating that when clicked some sort of action occurs. When using images be sure to test them with various monitors and resolutions to insure they are not distorted in one way or another.

When should hotspot images be used? When text can not, remember humans will respond better to images rather then text. This is not to say you should use clickable images everywhere, just when it appears that the topic may benefit from the use of graphics.

Example
In the window below there are two buttons labeled *Set Project* and *Set Order*. We could place the image into a help topic and create hotspots for each of the two buttons. When the user clicks one of the buttons a help topic would appear. They have touched the item they want to know about and have gotten help for the item. Sure, this could be done with only text in a *What’s this* style of showing help or as text in a help topic. You need to evaluate the users mind-set! I deal with users who may times will not even think of using *What’s this help* simply because they are really not

computer folks. They tend to look into a help file instead, this is were we need to make sure indexes are properly done, otherwise they will never see your graphically done topic.


Recent Delphi 5.0 project list

C:\Delphi\Projects JEDI\TreeViews\Demos\SysView\SysView.dpr
C:\Delphi\Projects JEDI\TreeViews\Demos\Martin_ComboTree\Project1.dpr
C:\Delphi\Projects JEDI\TreeView_Project\ComponentView\KSG1\Project1.dpr
C:\Delphi\Projects JEDI\TreeViews\ADO_MS_Access\GetStructure.dpr
C:\Delphi\Projects JEDI\UsingTheRegistry\EmulateBrnaches\Stuff\GetMDB Structure\Ge

The following image is from RoboHelp setting the topic to use for the *Set Project* button.

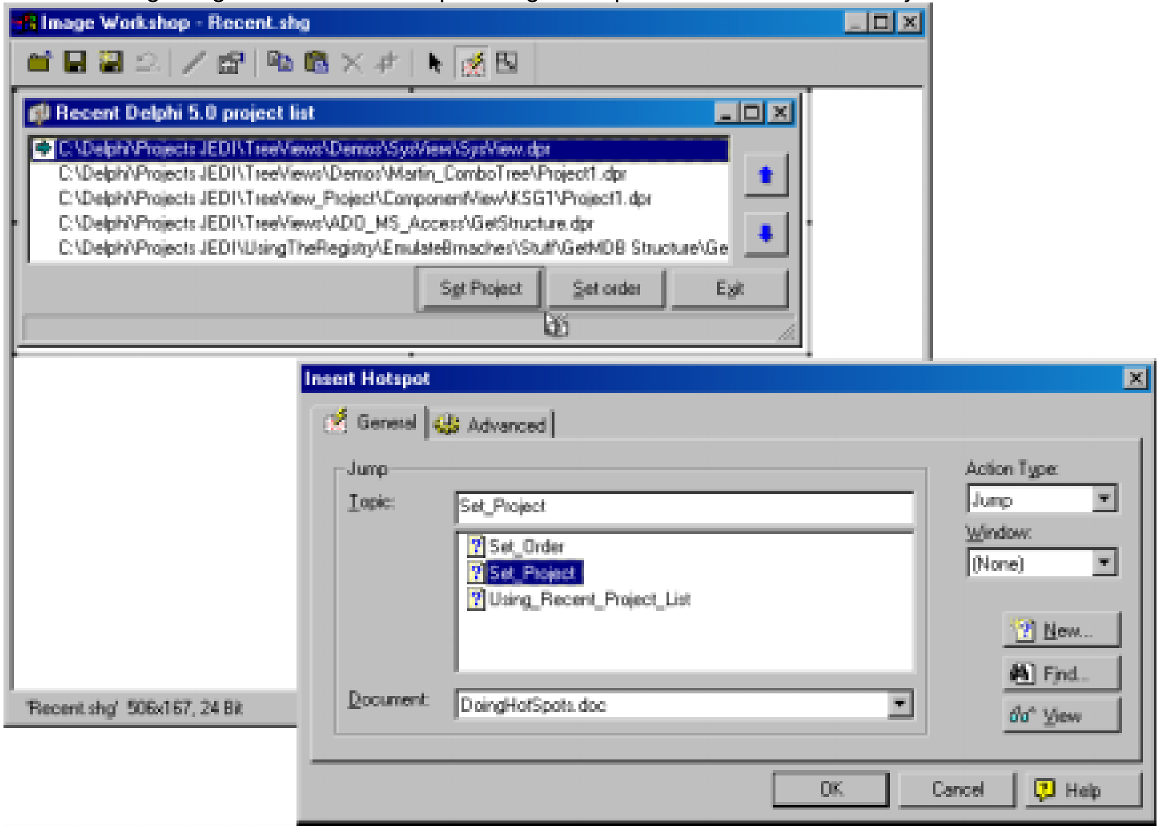


Image Workshop - Recent.shg

Recent Delphi 5.0 project list

C:\Delphi\Projects JEDI\TreeViews\Demos\SysView\SysView.dpr
C:\Delphi\Projects JEDI\TreeViews\Demos\Martin_ComboTree\Project1.dpr
C:\Delphi\Projects JEDI\TreeView_Project\ComponentView\KSG1\Project1.dpr
C:\Delphi\Projects JEDI\TreeViews\ADO_MS_Access\GetStructure.dpr
C:\Delphi\Projects JEDI\UsingTheRegistry\EmulateBrnaches\Stuff\GetMDB Structure\Ge

'Recent.shg' 506x167, 24 Bit

Insert Hotspot

Jump
Topic: Set_Project
Set_Order
Set_Project
Using_Recent_Project_List
Document: DoingHotSpots.doc
Action Type: Jump
Window: (None)

**TIPS**

Although today's help authoring tools hide most of the complexity of creating help it is a good idea to understand what goes on under the covers. For WinHelp there is a help file called *HCW.hlp* which is installed with Delphi, this file is under *YourDelphiInstallPath\Help\Tools*. Take sometime to read through it so that if you encounter problems you are better equipped to debug the problem. For HTML style help download from the API conversion section of JEDI the help file which explains working with HTML help in your Delphi applications. Marcel van Brakel has done an outstanding job here.

Now if you really want to make life easier I suggest taking a look at the help component library called ProHelp. This is a library, which consist of several components for implementing *What's this help* in your application. I take adding online documentation seriously and find this library a must have if you take online help as an important part of your application.

**Features**

- Standard Hints and Help Topics
  ProHelp comes with a set of automatically installable, fully customizable hints (45 items) and help topics (42 items) for standard objects and operating modes of an application. Given below, is a list of objects for which standard hints and help topics are available. Click hotspots to see standard help topics. You may use OnStdTopicHC event handler to assign context numbers on the fly.
- Assigning Help Topics to Nonwindowed Controls
- Assigning Separate Hints to Menu Commands
  Using the Hint property of the TMenuItem controls, you can assign separate hints for various states of menu commands.
- Displaying Help Windows and Hints for Standard Dialog Buttons; ProHelp makes it possible to assign hints and help windows to the standard buttons, which operate as Apply, Help, Cancel and OK action buttons without additional coding.
- OnUnknownMenuItemHC Event; OnUnknownMenuItemHC event occurs when igHelpMonitor receives a request for displaying the help window of a menu command that is neither the TMenuItem control nor the standard menu's command.

By dropping ProHelp's main component on a form you have given your application the ability to show hints in a TStatusBar which the component places on the form when it is place on the form. Another component handles the *What's this* help for the application.

Products

ProHelp
To see the latest information about ProHelp or download the most current version
http://www.igweb.pair.com

Help & Manual
http://www.ec-software.com

RoboHelp
www.blue-sky.com