Notes on "Fixing Up" HTML Help Created from WinHelp Projects

By Sid Penstone (S_Penstone@computer.org)

Last Edit: August 8, 2010

These notes were written to help others who are using HTML Workshop to convert their old WinHelp projects to HTML Help, based on the problems that I and other users have encountered. The notes suggest ways to recover some functions of the original project that are not preserved by HTML Help Workshop, such as context-sensitive help and popup text,. They are intended to supplement the excellent information already provided by others, such as:

"Converting WinHelp (HLP) to HTML Help (CHM)", which gives step by step instructions for conversion of WinHelp to HTML help. It is available at

http://www.help-info.de/en/Help_Info_WinHelp/hw_converting.htm

Other references are listed at the end.

Topics

Conversion with HTML Workshop - Overview

Microsoft has announced that WinHelp Help files will not be supported in Vista and subsequent operating systems, and recommends that all new Help projects be written in HTML Help format. For many developers, this means that existing Help files must be converted to HTML Help.

Microsoft's HTML Workshop (referred to as "HHW" in these notes) has the ability to convert an existing WinHelp 4 help project to an HTML Help Project. It asks for the path to the old .project (*.hpj) file and the location and project name for the HTML project to be created. It then creates a complete HTML Help project from the WinHelp project files – each topic of the .rtf files is written as a separate HTML (*.htm) file in a /html folder, a contents file (*.hhc) with an item pointing to each topic file is created, and if there were K-words in the WinHelp project an index file (*.hhk) is created. Images are also extracted from the .rtf files (if they are included by the correct method) and written as GIF (*.gif) files in an /image folder.

The resulting HTML project can then be compiled by HHW into a .chm file that retains the original jumps between topics of the original WinHelp project, as well as a complete table of contents. HHW creates an individual HTML file for each topic in the original project., using a name (not related to the original topic id or title) that is based on the name of the .rtf file and a hash code. However, the automatic conversion introduces some problems, since it does not convert all of the properties of the original WinHelp project. This can result in considerable editing of the new HTML Project.

These notes contain suggestions on minimizing the amount of work. We also suggest the use of the HHPMod utility to rewrite the HTML project before starting any editing.

Problems with the Automatically Created HTML Help Project

- Context-sensitive Help, e.g. using Map and Alias files or definitions is lost;
- A-links used for "See Also" buttons are not transferred;
- Images are only converted and included if they were linked, or inserted using the {bmc} etc. method in the original .rtf file. (Microsoft has identified this as a bug in HTML Workshop);
- "Hot-spot" hypergraphics image files (*.shg), if transferred, are converted to inactive *.bmp or *.gif images;
- Images stored in subdirectories are not correctly mapped to the new project;
- Popup text topics (marked by a single underline) in the .rtf file are converted into regular topic files, and so appear as a new topic in the Help window instead of a popup in the current Help window;
- The names of the .htm topic files are not related to the original topic names or topic IDs.

Problems with HTML Workshop's Handling of Images

1. HHW will only extract and convert images that are inserted in the rtf file by the WinHelp macros {bmc}, {bml}, {bmr}; e.g. {bmc myimage.bmp}.

It will accept Windows bitmaps (*.bmp), Windows metafiles (*.wmf) or GIF (*.gif) files. They will be converted to GIF (*.gif) files, written into the /images folder of the converted HTML Help project, and linked in the .htm file with a <img= …> tag.

Microsoft reported on October 16, 2002 (http://support.microsoft.com/kb/267959) the failure to extract embedded images as a bug in HTML Workshop, but there has been no sign (Oct. 2007) of a fix.

If your .rtf file included the images by pasting them (embedding) into the file, you must locate your original images in order to insert them in the HTML help file.

2. There is another problem if you placed the images used in the .rtf file into subfolders to keep them organized. (e.g. mydiags/structures, mydiags/contents, etc.) HTML Workshop will find and convert them to .gif correctly and it will also add the full path to the text in the <img > links in the .htm file. *But* it will actually write all of the .gif files into the same /images folder in the new project, so that they will not be found in the HTML Help. You will have to manually create the new subfolders and move the .gif files to them, or go back and modify the WinHelp project so that all of the images are in a single folder, and then convert the project again.

Some Things To Do Before the Conversion by HHW

- Make a list of the A Links in your .rtf files – which topics contain the text of the A Link, and which files contain the references to the A Links. You will have to restore these manually.
- Make sure that all of the images are linked or included with the {bmc imagename} method, since embedded images will not be converted. It may be worth while to make new versions of the .rtf files involved that include the images using *both* methods, and a new version of the project (*.hpj) file to call the new .rtf files and create a WinHelp file. Insert each {bmc} version next to the embedded version so that it will appear next to it in the compiled .hlp file. This will allow you to confirm that the images are in fact the same (note that you cannot preview the {bmc} format images in Word!). When you convert this new project version with HTML Workshop, the embedded images will be ignored; only the {bmc} images will be converted. The new version of the .rtf files using the {bmc imagename} will not produce the WinHelp layout that you would like, but HHW will not preserve your image layout perfectly anyhow. It is more useful to have the image files transferred automatically to the new project. You can manipulate the position of the images there.
- If your images are in subdirectories (sub-folders) they will not be located correctly in the converted project – move them all to a single folder before conversion.

- If you want the topic title lines in your .rtf file to be converted into <H1>,<H2>, etc. lines in the HTML files, make sure that their Paragraph format is linked to one of the "Levels" in Word: e.g. Level 1 for Heading 1, etc. and that the font is **bold**.
- If you have multiple Topic IDs in a topic in your .rtf file, record the details in case you need to reconstruct the calls to them later.
- Identify which topics are used as popup text; you will have to extract the text manually and modify the references to them in the new project if you want them to be popups.
- Record the details of the links used in your hypergraphics images; the images will have to be edited in the HTML project using the HTML methods for "hot spots".
- If you have context sensitive ID's defined in map or alias files, print the files out for later reference.
- If you are using map files, make sure that all comments are preceded by semicolon characters (;). Files using C-type comments (/ and /* characters) may be read incorrectly or skipped by HTML Workshop.

The HHPMod Utility Program

HHPMod is a free utility (available from http://post.queensu.ca/~penstone) that reads the HTML project file and HTML files produced by HHW, and (optionally) the original WinHelp4 project file, and builds a complete new HTML help project in a new folder. It performs the following:

Finds the original Topic ID strings used in the WinHelp .rtf files, and renames the HTML files by their original Topic IDs. This makes identifying the files for editing much easier.

Rewrites the new HTML files with all of the internal links pointing to the renamed files.

If there were context-sensitive Help features in the WinHelp project, as defined by map and alias header files, HHPMod parses the original header files and writes new versions that link to the new HTML topic files. When the new project is compiled, your original calls based on program IDs will be restored.

Identifies the topic files that were probably popup text topics in the WinHelp project because they have a Topic ID but no title.

Extracts the text from the untitled topic files to text files that will support popups within the new HTML Help project, and also in application program popups.

Optionally edits the topic files that call popups within the HTML help file to use Javascript popup calls to enable the popups.

For more information, see the topic Using HHPMod.

Using HHPMod to Modify the New HTML Project

The program is available from http://post.queensu.ca/~penstone as a .zip file. If you have Visual Basic 6 installed (or you are updating an earlier version of HHPMod) then you need only download the executable and help files. If you do not, then download the "Full" version.

You will need to know the location of the HTML project created from your WinHelp project. If you have any context-sensitive topics, then you will also need the original WinHelp project files, including the header files (*.h, *.hm).

You should NOT have edited any of the files in the new HTML Help project, because HHPMod parses the files assuming that they are exactly as written by HHW. If you are not sure, then use HHW to create a new HTML Help project from your WinHelp project in a new folder, and apply HHPMod to this project.

1. Start the HHPMod program.
2. The latest version (Version 3.x.x, August 2010) operates aas a "Wizard" with steps to help the user through the rewriting procedures.
3. In the "HTML Project (*.hhp) Step", enter the location of the HTML Help project created by HTML Help Workshop by converting your WinHelp project.
4. If you have Map and Alias data to recover, in the "WinHelp Project (*.hpj) Step", select the original *.hpj project. If there is any Map or Alias information in the project, the checkboxes in the Wizard frame will be checked.
5. Click the button "Next>" to leave this step. HHPMod will scan both project files, all of the .htm files of the HTML Help project, any map header files and alias header files, and create internal lists of the data. You may get messages and questions during this process. The progress bar at the bottom of the form should indicate progress, and finally the text "Successfully scanned all files." should appear, together with any warning messages. HHPMod will have stepped to the "New HTML Project Step".
6. (If you wish to see the correspondence between the original Topic IDs and the .htm files created by HTML Workshop, select the menu "View/Scan Log". This presents a list of all of the .htm files, the Topic ID contained in them, and the Topic Title from the original .rtf files.)
7. In the "New HTML Project Step" , select the location where you wish to create the new HTML project. The new project must be written in a different folder than the HTML Help project being modified, since it will include new versions of the .hhp, .hhk and .hhc files with the same name.
8. When the name and path of the new project are satisfactory, click the button "Next>". HHPMod will write new versions of the project file, the contents file, index file (if present) and all of the .htm files. Each new .htm file name will now be the Topic ID contained in the file. All references in all files will be changed to use the new file names.
9. If there were map and alias data in the WinHelp project, the map data is duplicated in the new project, and new versions of alias files are created and written to the new project folder. If there were images in the automatically generated HTML project, they are simply copied to an /images folder in the new

project folder. If there were untitled topics, HHPMod will create the text files "popups.js" and "cshelp.txt" to support local and application driven popups, respectively.

10. For the record, you should examine the log files from the operation – select the menu "View/ Scan Log File" and "View/ Rewrite Log File" .
11. If the conversion was successful, you are now in the "Popups Activation Step". If you wish to restore all of the popups within the Help files, click "Yes" in the box shown, and then click "Next>". HHPmod will edit the topic files that call local popups; and will save a copy of the unedited topic files in the folder "/html/nopopups/".
12. Open the new HTML project file in HTML Workshop and compile the project. You may test the context-sensitive topics using the "Test/HTML Help API" menu. Select the command "HH_SELECT_CONTEXT", clear the "WINDOW" box, and enter the map number to be tested. The topic corresponding to the number should appear.

If you have problems or comments about HHPMod, please send a message to the e-mail address shown on the "About" box of HHPMod.

Recovering Context-Sensitive Help

Programs using WinHelp call the WinHelp() function of the Windows API to access and display a particular topic in a .hlp file. The WinHelp function requires a numerical argument to define the topic to be accessed. The programmer usually uses a symbolic argument within the program. This leads to WinHelp Map and Alias data files.

HHW does not access any of this data, so the links must be rebuilt for HTML help. WinHelp aliases refer to Topic IDs from the .rtf file, but HTML Help uses the HTML file names in its links.

The Map information (mapping of symbolic names to numerical values) from the WinHelp project can be used without change, by copying the information from the [MAP] section of the WinHelp project file to the [MAP] section of the HTML Help project file. Any *.h map files must be copied to the HTML project folder.

The Alias information must be edited to point to the HTML file containing the original topic to be displayed. The automatic conversion produces file names that are not related to the original topic names or IDs, so this is difficult to do manually.

The utility HHPMod described above can read all of the map and alias information from the WinHelp project file and automatically write the required new versions of the data in a new HTML Help project. This modified project will use the topic id's of the original project.

For manual recovery, see the topic Manual Rewriting of Map and Alias data.

Recovering Hot Spots in Bitmaps

HHW will convert the hypergraphics files (*.shg) in the WinHelp .rtf files to .gif files and write them in the \images folder of the HTML help project. If it does not, check the notes on preparing for conversion. Note that the old image hotspots will be lost.

You must reinsert your hot spots using a Web editor such as Front Page, Expression Web, Dreamweaver, etc. This is done in the text of the HTML file that displays the image; the image file is not modified. Each hotspot will appear as an <area ….. > tag in the file.

If the original target of the hotspot was a regular topic in the help file, then simply set the "href= " to point to the HTML file that now contains this topic. Use the list you prepared from the WinHelp project. If you are working with a project that has been modified by HHPMod, then select the file with the same name as the original Topic ID used in the WinHelp .rtf file. If you do not use HHPMod, then you must locate the correct HTML file by searching them.

If the original target was a popup topic, it will require some additional work to create the popup text from the topic file; HHW will have converted the popup topic to a regular HTML file. This cannot be forced to display as a popup – the text must be extracted written as part of a JavaScript script, and invoked as a JavaScript popup. See the notes following on recovering this popup text.

Recovering Popup Text Called From Help Topics

Popup text topics in the .rtf file (whether they have a title or not) will be automatically converted by HHW to an HTML topic file. To display these as text popups in the HTML Help window, they will be managed by JavaScript function calls in the HTML files that call these popup topics. The JavaScript call requires that the text be available either within the HTML file that will display it, or in a script file that is referred to in the HTML file. HHPMod will perform this editong automatically (see above).

Setting this up is described in the Help file of the HTML Workshop. HHPMod will do this automatically (see above). Our experience is that if there are several of these popup topics, it is best to prepare a script file containing all of the text strings, and refer to it in the HTML files that display the text. This permits editing of the text with a text editor. HHW says that the script file name and path must be included in the [FILES] section of the HTML project file (we have found that this does not seem to be necessary, as long as it is in the same folder as the HTML files invoking it, but there may be problems if it is not included). The file name can be added to the [FILES] section by a text editor.

If you do not use HHPMod, we recommend that you extract the text of your popups from the original .rtf files and copy it to a text file (JavaScript Script file) with the .js JavaScript extension. Open the .rtf file in Word, and select "View/Footnotes". Open up a new text file in a text editor such as Notepad or TextPad.

Now, go in turn to each of your popup topics and do these two steps:

- In the Footnotes window, select and copy the text of the # footnote (this is the topic ID that is used to access the popup topic) and paste it as a new line in your text file. Type an equal sign, a pair of quotes and a newline after it. (Don't copy the # sign at the start of the footnote.) If the footnote was #idh_popup, then you will have:

  ```
  Idh_popup=""
  ```

- In the main window, select and copy all of the text in the topic and paste it into the text file between the quotes. You will now have an entry of the form:
  `idh_popup1="This is a sample popup."`

Where `idh_popup1` is the ID that will be used to call the popup, and the text in quotes is the text to be displayed in the popup window.

When all of the topics have been copied, save the text file in the HTML Help project /html folder with the extension ".js". Example – "popups1.hs".

Go to the other .rtf files and add their popup text to the file.

See the notes "Invoking Popup Text in HTML Help" below to reconstruct the popup action.

After you have replaced and tested each of the popup calls with their Javascript method, you may delete the .htm files that contain the popup text, unless they are also called as regular topics elsewhere in the project.

Note that Popup text displayed by a program, such as "What's This?" help, uses different methods.

Recovering Popup Text Called from a Program

This requires editing within the program source code to replace the calls to WinHelp with calls to HTMLHelp, and only involves the HTML Help file if topics in the .chm file are required. It requires HH_POPUP structures and HTMLHelp() API calls within the program, not in the HTML help file.

Refer to the instructions in the HTML Help that is part of HTML Help Workshop.

The popup text required to support application popups must be writeen in a text file (usually called "cshelp.txt"), in a format:

```
.topic idh_topic3
This is the text to be shown
```

Where `idh_topic3` is the context ID. This must be available as a numeric value; this is accomplished by including a header (*.h) file that defines the values of the context ID symbols. (Note that the cshelp.txt file can also be written directly with the numerical values, (e.g. `.topic 9823`) but this makes it more difficult for the programmer.)

Topic IDs in WinHelp Files

The "Topic ID" is the string entered as the # footnote in the .rtf topic file. It is attached to the first line of the file. If the topic displays a title, there will be a $ footnote attached to the same line. (If the Topic ID is to be mapped or aliased, it should not contain spaces.)

Topic ID's are expected to be unique in WinHelp projects, since there is no way to distinguish them in the compiled .hlp file. However, it appears that the WinHelp compiler does not check if the same Topic ID string is used in more than one topic in the

project. This is not a problem in HTML Help, because the Topic ID string can be prefixed by the name of the .htm file.

Also, it is acceptable in WinHelp to have more than one # footnote applied to the same topic (they must be unique in this case).

When HHW converts the WinHelp project to a HTML Help project, the Topic ID of each topic is written as an HTML "anchor" <A > in the corresponding .htm file. If there was a title ($ footnote) in the topic file, the anchor appears attached to the title. For example, if the # footnote was "finding_context_ids" and the title of the topic was "Finding Original Context IDs" the .htm file will contain:

```
<H1><A NAME="finding_context_ids"></A>Finding Original Context IDs</H1>
```

If there were multiple Topic IDs (# footnotes), they appear as additional anchors following the first anchor in the same line of the .htm file.

WinHelp Map and Alias Operations

The WinHelp compiler requires that any symbolic arguments of the program be mapped to numeric arguments in the [MAP] section of the WinHelp project file, either directly by a "#define" line in the project file or in a header file that is referred to by an "#include" statement in the project file. The application program's compiler often produces the header file (usually with a .h or .hm extension).

To connect to the specified topic in the .hlp file, an "alias" is required, that defines the Topic ID of the topic to be displayed. This is done in the [ALIAS] section of the project file, either by a line definition "Symbol=Topic ID", or in an alias file referred to in an "#include" line.

HTML Help Map and Alias Operations

The principle is the same as that of WinHelp map and alias operations, except that the alias connects the symbolic constant to a .htm file name, rather than a Topic ID. Because HHW does not retain the original Topic ID's when naming the .htm files, it is difficult to write the new alias files manually. See below.

Manual Rewriting of Map and Alias Data in the HTML project

If you are not using HHPMod to rewrite your project, then follow these steps; but note that HHPMod would perform all of these steps automatically.

Header files from the WinHelp project containing Map definitions can usually be used without change. However, you should open the old files in the WinHelp project and confirm that:

- All comments follow semicolons
- There is only one whitespace character (space, tab) after the word "#define"

The map definition files must be copied to the HTML project folder, and their names added as an #include to the [MAP] section of the HTML *.hhp project file.

Microsoft has advised that there is a bug in HTML Workshop that may cause problems if you add them from the workshop, so it is fairly simple to add them with a text editor. If there is no [MAP] section, then add this text below the end of the [FILES] section:

```
[MAP]
#include filename
```

Where "filename" is the path to the map header file relative to the project file. If the file is in the same folder as the project file then no folder information is required.

If the WinHelp project file already included map definitions in its [MAP] section, you can simply copy its [MAP] section directly to the [MAP] section of the HTML project file.

Alias data must be edited for use in the HTML project. Use a text editor to write new versions of the old alias files and then #include these new alias files in the HTML project file's [ALIAS] section. Do not write alias definitions directly into the [ALIAS] section; while this will work, you should minimize manual editing of the HTML project file.

The WinHelp alias file aliased each symbolic constant to a Topic ID that existed in the .rtf file. In the new alias file for HTML Help, the alias must point to the .htm file that contains the topic with that Topic ID.

So if the original alias line was:

```
IDH_DEFINITION = Definitions
```

The new file will contain the line:

```
IDH_DEFINITION = html\top1g6d8.htm
```

Where "top1g6d8.htm" is the topic file created by HHW from the .rtf topic with the Topic ID "Definitions". (Note that HHPMod would rename the files by their Topic ID, so if you use HHPMod the line would be: `IDH_DEFINITION = html\Definitions.htm`)

After each new file is written, save it to the HTML project folder, and add the name of the file to the [ALIAS] section of the HTML project file.

Map and Alias Formats

Map header files (usually *.h or *.hm) are often written automatically by the program development software. The format acceptable to HTML Help in the header file or as a line in the [MAP] section of the HTML project file is:

```
#define SYMBOLIC_CONSTANT numeric_value
```

Example:

```
#define IDH_HELP_ON_DEFINITIONS 2437
```

Note – there must not be multiple white space after the `#define` or the HHW compiler will report an error.

The alias format either in an alias header file or in the [ALIAS] section is:

```
SYMBOLIC_CONSTANT = htm_file_path
```

Example:

```
IDH_HELP_ON_DEFINITIONS = html\Definitions.htm
```

Comments can be included with the map or alias data, prefixed by a semicolon (;). Note that files containing C- type comments may be read incorrectly, or skipped entirely. If you test your .chm file with the “Test/htmlAPI” menu in HTML Workshop and get an error message “HELP_CONTEXT without a MAP File” error, it may mean that your map file format was unreadable.

Invoking Popup Text in HTML Help

Here the user is presented with a link in the topic that when selected pops up a box in the current window containing text, such as a definition. Setting this up is described in the Help document that accompanies the HTML Workshop, but a summary of the steps (note that HHPMod can now do this automatically) is:

1. Extract the text of the popups and their Topic IDs from the original .rtf topic files and create a JavaScript script file, as described above.
2. Insert an instance of the HTML Help Active-X control into the .htm file that will call the popup. (Only one instance is required per .htm file.) The object should be inserted between the <head> and </head> tags. The id of the instance will be used in calls. For example, assume that this instance is called "mypopup":

```
<object id=mypopup type="application/x-oleobject"
classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11">
</object>
```

3. Also inside the <head> and </head> tags of the .htm file, insert a reference to the script file that contains the text blocks to be displayed :

```
<script type="text/javascript" language=JavaScript
src="FormPopups.js">
<!—
Comments can go here.
The file "FormPopups.js" contains a font definition:
Popfont="Arial,9,,plain",
and all of the text blocks to be displayed together with their identifying
Topic ID.
-->
</script>
```

4. Insert a call to the "TextPopup" method of the control instance into the <body> where the popup is to be invoked. This example uses a hotspot in an image, and displays the text with the Topic ID "HTMLDATA" in a popup window over the image. It invokes the control instance "JavaScript:mypopup".

```
<map name="FPMap0" id="FPMap0">
<area shape="rect" coords="11,47,309,131"
href="JavaScript:mypopup.TextPopup(HTMLDATA,Popfont,9,9,-1,-1)" >
</map>
<img alt="HHPMod Main Form" src="../images/RunForm.bmp" width="422"
height="519" style="float: left" usemap="#FPMap0" class="style1"></p>
```

The argument "Popfont" refers to a string in the script file "FormPopups.js" that defines the font to be used in the popup text (example: Popfont="Arial,9,,plain").

Miscellaneous HTML Help Workshop Notes

- If you use the HHW menu "Test" to test your context items, the .chm file may lock up and prevent HHW from compiling a new version. This sometimes happens when there is an error in the test. Closing and reopening the project file usually works. If not, then exit and restart HTML Workshop.
- Make sure that when you select “View Compiled Help File” or the View button that you have set the drop-down window to point to the same .chm that you just compiled. HHW remembers the last one that you looked at before you closed HHW, which may be different than the one that you just compiled !

References and Sources

There are many useful references available at http://www.help-info.de

"Converting WinHelp (HLP) to HTML Help (CHM)"
Step by step conversion of WinHelp to HTML help -
http://www.help-info.de/en/Help_Info_WinHelp/hw_converting.htm

"Creating HTML Help with Microsoft's HHW"
by Char James-Tanny
available as a PDF document by download from
http://mvps.org/htmlhelpcenter/htmlhelp/hhtutorials.html

"HTML Help Beginner's Guide" by Simon Law
available as a download from
http://www.help-info.de/download/HelpOnHTMLHelp.zip

"Converting WinHelp to HTML-based help"
By Neil Perlin
Article reviewing the problems and options for conversion
http://www.stc-nne.org/Noreaster/Nov01/converting.htm

Bug in HTML Workshop handling of images:
http://support.microsoft.com/kb/267959

Bug in HTML Workshop when adding map or alias data:
http://support.microsoft.com/kb/188444/en-us

Rob Chandler's FAQ page
http://helpware.net/FAR/far_faq.htm