Authoring HydroDesktop Help

September 20, 2010

By Tim Whiteaker

# Introduction

This document describes how to author help for HydroDesktop. It starts with an introduction to the help system and reST, the source file format for the help files. Then it lists conventions to follow and preferred words to use so that our documentation will look consistent even though it has multiple authors. Finally, an example help file that shows how to operate a hypothetical button is provided. The example shows some of the more common features that authors should include in their help files.

HydroDesktop help is implemented as HTML files can be viewed in a Web browser. We manage these files in source control just like we do with the HydroDesktop source code itself. We use software called Sphinx to generate these HTML files from a set of source files. The nice thing about Sphinx is that it reads our source files to create a Web site that includes a table of contents, index, and search capabilities, along with custom styles for elegant presentation of our content.

Most of the team is new to Sphinx, so we’re still figuring out the best protocol to follow when adding new help files. Below is the suggested course of action to take if you want to contribute help files.

First, you should install Sphinx and learn how to write the source files, which are called reST files. Although it’s not required to install or run Sphinx in order to create reST files (a simple text editor is all you need for that), doing so will help you get a sense of how this all works, and you’ll be able to instantly compile your reST files into HTML to see how your content will look.

Next, view the content in source control at Documentation\OnlineHelp\source, particularly the files with the .RST extension. These are the source files, or reST files as they are called. Read **How to Organize Our Content.docx** for guidelines on where to put your source files when you are ready to contribute to the HydroDesktop help.

Finally, contribute some content. Feel free to build the HTML output to see what the result looks like. Once the content is finished, remember to add the content to source control and perform an SVN Commit. Then, your new content will be available to all help authors and will make its way into the official HydroDesktop help system.

**Action items:**

1. Follow the instructions in **Documentation\OnlineHelp\How to Use Sphinx.docx** for using Sphinx.
2. Read the instructions in **readme Help Authors.docx** for rules on documentation conventions that we use to give our help a consistent look and feel.
3. Create and build your own HTML output using Sphinx to see how the system works.
4. Read **Documentation\OnlineHelp\How to Organize Our Content.docx** to learn where to put your source files.
5. Create HydroDesktop help files in the appropriate place and commit them to the source code repository when you are satisfied with them.`
6. Ask your code developers to read **How to Access Help from HydroDesktop.docx** for instructions on how to access your help files within their code.

I recognize that these instructions are sparse. Please feel free to expand upon this documentation to include details that I left out which would have otherwise saved you a big headache.

# How to Create Help Content

Let’s suppose that you’ve created your first Sphinx project or that you’ve downloaded the HydroDesktop help project from the source code repository and want to add some help files. At this point, all you need to do is create or edit some reST files (the file format for Sphinx) and then use the make html command to update the output.

**Tip**The HydroDesktop help project is located in the Documentation\OnlineHelp folder.

Sphinx uses reStructured Text (reST) as its source file format. There is documentation about reST at <http://sphinx.pocoo.org/rest.html> and <http://docs.geoserver.org/trunk/en/docguide/sphinx.html>. You can also see a tutorial at <http://scienceoss.com/use-sphinx-for-documentation/>. I like to set Notepad or some other simple text editor as the program used for opening .rst files. There’s also an online editor at <http://cometdemo.lshift.net:8080>.

To edit a reST file, open the file with a text editor and make your edits. It’s just ordinary text, but with certain rules for markup such as placing dashes underneath words to indicate a title, so any text editor will do. As an example, I like to clean up the default welcome.rst file that was created by sphinx-quickstart, to look like the text below (I removed the module index, shortened the title, and added a subheading called About HydroDesktop).

.. HydroDesktop Help master file, created by

sphinx-quickstart on Tue Jul 20 13:49:57 2010.

You can adapt this file completely to your liking, but it should at least

contain the root `toctree` directive.

Welcome to HydroDesktop Help

============================

About HydroDesktop

------------------

HydroDesktop is a free and open source desktop application developed in C# .NET that serves as a client for CUAHSI HIS WaterOneFlow web services data and includes data discovery, download, visualization, editing, and integration with other analysis and modeling tools.

Contents:

.. toctree::

:maxdepth: 2

Indices and tables

==================

\* :ref:`genindex`

\* :ref:`search`

To create a new page, simply create a new text file and give it a .rst file extension. Some example content for a page describing basic map navigation in HydroDesktop could be:

.. index::

single: navigation

Navigation

==========

Basic Navigation

----------------

#. Click the zoom tool.

#. Use the zoom tool.

Advanced Navigation

-------------------

#. Click the zoom tool.

#. Close your eyes.

#. Spin around three times.

#. Use the zoom tool.

The above text shows numbered lists and different heading levels. It also adds an entry in the topic index for this page under “navigation.” I saved this page as navigation.rst. Next, you could add this page to the table of contents in the welcome.rst file.

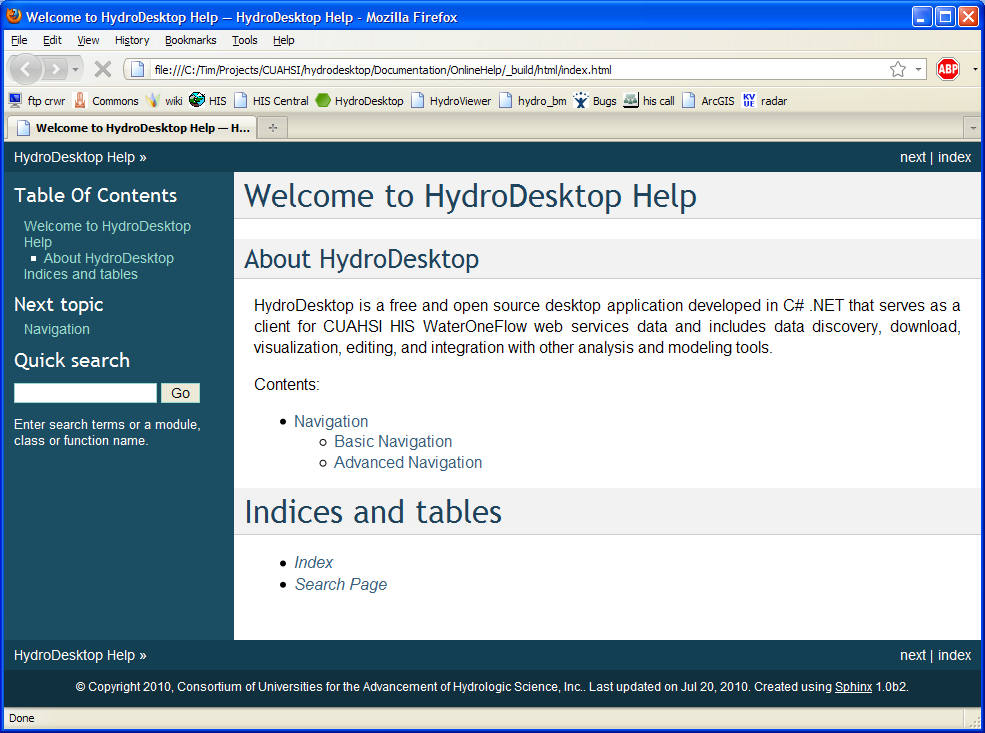
Contents:

.. toctree::

:maxdepth: 2

navigation

Now if you save all files, open a command window in this folder (**Start | Run | cmd**, then navigate to the folder) and enter make html, you’ll create HTML output in the output directory that looks something like the screenshot below.



The presentation of the above content is determined largely by CSS. One way to change this is to create your own CSS that extends or modifies the default CSS created by Sphinx.

**Note**The HydroDesktop Help Content Officer will manage the CSS we use for the help system. This discussion about CSS is included here just in case you were curious about it, or if you are good at CSS and want to contribute to the CSS we use for the help system. Please work with the Help Content Officer when making changes to the CSS we use for the help system.

The example below shows how to change the background color of <h1> elements.

1. Create a CSS file, such as mystyle.css, and save it in the \_static folder that is at the same level as your source .rst files, e.g., OnlineHelp\source\\_static.
2. Add the following text to mystyle.css. This imports the style rules from the default style sheet created by Sphinx, and overrides the background color for <h1> elements within the <div class=”body”> element, setting it to fuchsia.

@import url("default.css");

div.body h1

{

background-color: Fuchsia;

}

1. Save mystyle.css.
2. In conf.py, add or modify the line to set the style sheet to use: html\_style = 'mystyle.css'
3. Save conf.py.
4. Make the html and examine the output.

**Tip**  
Sphinx only rebuilds files that it thinks has changed. This is a handy feature, but sometimes it doesn’t realize things have changed such as CSS files. If Sphinx refuses to rebuild the content because it thinks no files are out of date, deleting the .buildinfo file in the output html folder will force Sphinx to rebuild everything.

**Note**  
Do not manually edit the HTML files generated by Sphinx. These files will be regularly regenerated from source, so any changes you make to the HTML will be erased by the next build. Rather, you should make your changes in the source reST files. If there’s a fancy widget that you want to add to your HTML page that reST does not accommodate, then you’ll have to find a way to live without the widget.

## Working with the Table of Contents

### Creating a Table of Contents

The toctree directive creates a table of contents (TOC). For example, if welcome.rst has a TOC, and you want to reference three files in your TOC (About.rst, MapNavigation.rst, and SavingProjects.rst), you can enter the following text in welcome.rst.

Here’s my Table of Contents:

.. toctree::

About

MapNavigation

SavingProjects

The resulting HTML will have the main headings of those three files in the TOC, as well as any subheadings. Sphinx reads the main headings and subheadings from those files and not only places the text at the correct nesting level in the TOC, but also links directly to each heading or subheading. Working from the above example, suppose About.rst looks like the following.

About HydroDesktop

==================

HydroDesktop is cool.

History

-------

Vision

''''''

HydroDesktop started a couple of years ago.

Prototype

'''''''''

Beta

''''

Open Source Development

-----------------------

Open source is cool.

The output TOC in welcome.html might look like the following.



### Controlling How Deep the Nesting Goes

Suppose your files have headings, subheadings, and sub-subheadings. Your TOC can get very long in that case, so you might want to limit the TOC to show only headings and subheadings (but not sub-subheadings or anything smaller). You can do this by setting a maxdepth option, in this case to 2.

.. toctree::

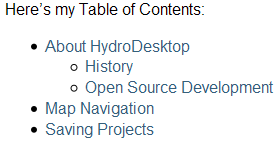
:maxdepth: 2

About

MapNavigation

SavingProjects

The output TOC in welcome.html might look like the following. Notice the sub-subheadings under About HydroDesktop are no longer visible.



### TOCs within TOCs

Another useful feature is the ability to read TOCs within those documents and add those items to the main TOC. Suppose the MapNavigation file has a TOC that links to separate pages for Zooming, Panning, and Refreshing, saved as mapZoom.rst, mapPan.rst, and mapRefresh.rst, respectively. Below is MapNavigation.rst.

Map Navigation

==============

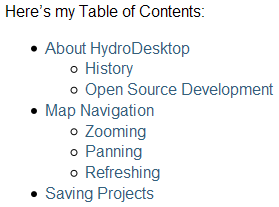
.. toctree::

mapZoom

mapPan

mapRefresh

The output TOC in welcome.html might look like the following. Notice subheadings under Map Navigation.



Sphinx treats each item in the MapNavigation TOC as its own heading. The key thing to remember is that those TOC items are considered to be subheadings below whatever is above the TOC in the rst file. In this example, notice the Map Navigation heading (with the equals signs underneath) which appears before the TOC. If a subheading of Map Navigation appeared before the TOC, then the TOC items would be considered sub-subheadings and thus they wouldn’t show up in our main TOC in welcome.rst because we set a maxdepth of 2.

# Documentation Conventions

This section lists conventions that help authors should use so that our help product will seem consistent across all topics. The conventions and their implementation in reST are described below.

Use emphasis when you want to highlight a word in paragraph text. To indicate emphasis in reST, surround the term with \*. For example:

HydroDesktop stores data in a special database called the \*Data Repository\*.

Use *strong* to indicate where a user clicks in the user interface when providing procedural steps. To indicate strong in reST, surround the term with \*\*. For example:

#. In the Ribbon, in the \*\*Table\*\* tab, in the \*\*Database\*\* panel, click \*\*Change\*\*. The location of the current database is shown in the dialog that opens.

Use numbered lists for procedural steps. In reST, each item in a numbered list begins with #, e.g.,

#. In \*\*Map Layers\*\*, right-click \*\*U.S. HUC\*\*, and then click \*\*Properties\*\*.

#. Click the \*\*Detailed Properties\*\* tab.

Don’t inundate the documentation with screenshots. This makes the documentation hard to maintain when screenshots need to be updated. Instead, use clear, concise text, and only include screenshots when text alone is not sufficient to communicate the message.

Use pipes (|) to separate menu and submenu commands, e.g.,

#. In \*\*Map Layers\*\*, right-click \*\*U.S. HUC\*\*, and then click \*\*Labeling | Label Setup\*\*.

For tips on how to write good tutorials, see <https://www.utexas.edu/its/style/written/procedure.php> and <http://docs.geoserver.org/trunk/en/docguide/tutorial.html>.

# Word List

Here are words that often have multiple spellings or capitalizations. Let’s use the following:

right-click

Web service (capital W)

Web site

e-mail

For more suggestions, see <http://www.utexas.edu/visualguidelines/tech.html>.

# Example reST File

This example shows the source reST file (exporting-shapefiles.rst) for a help topic describing how to export spatial data to a shapefile.

.. index::

single: exporting shapefiles

single: shapefiles, exporting

Exporting Shapefiles

====================

Spatial data can be exported from the map layers in HydroDesktop to a shapefile on your computer. This is useful when you want to use the data in other Geographic Information Systems software.

To export spatial data to a shapefile:

#. If you don't currently see the map in HydroDesktop, click the \*\*Home\*\* tab to show the map.

#. In \*\*Map Layers\*\*, right-click the name of the layer that you want to export, and then click \*\*Data | Export Data\*\*.

#. Chose export options and the location of the output shapefile, and then click \*\*OK\*\*.

The data are exported to the specified location. When the export is complete, you will be prompted about adding the exported shapefile to the map.

Let’s now dissect key portions of that file. At the top of the file is an index directive that tells Sphinx to add entries to the index page that it creates.

.. index::

single: exporting shapefiles

single: shapefiles, exporting

Two entries are added:

* exporting shapefiles
* shapefiles, exporting

This entries will appear in alphabetical order in the index. Two entries were added in this case to handle the two most likely keywords that a user might search for when trying to learn how to export shapefiles. When the user clicks one of those topics in the index, they will be brought to the text that appears just below the index directive (in this case, the heading “Exporting Shapefiles”). See the links above for reST documentation for more options when specifying index entries. Indexes are one of the primary ways users find help for a given topic, so don’t forget to include index entries for your topics!

**Tip**  
Always view the index in the output html after you have added entries in the index. Make sure your entries appear as you expected.

Back in our reST file, we see that the main heading for this help topic appears after the index directive.

Exporting Shapefiles

====================

Note how the heading is marked by using equals signs underneath the heading. Similarly, we could’ve added subheadings to this document by using characters such as ----------- or ‘’’’’’’’’’’’’.

The next snippet of reST contains several items of interest. It is part of a numbered list containing instructions for exporting shapefiles.

#. In \*\*Map Layers\*\*, right-click the name of the layer that you want to export, and then click \*\*Data | Export Data\*\*.

Key things to note about that text are:

* The “#.” Characters denote a numbered list.
* Names of buttons or menu items that the user should click use *strong* styling, indicated by \*\*.
* The correct term from the word list is used to refer to the command “right-click” (vs. right click or Right Click).
* The pipe character delimits the **Data** menu and **Export Data** submenu.

The remainder of the reST file contains ordinary paragraph text. When this file is compiled into HTML, an attractive style will be applied to turn this content into clean, presentable, and indexed help content.

**Once this reST file is complete, you must add an entry for this file in the table of contents of another file in the project.** Otherwise, users will have no way of navigating to this topic using the table of contents. In this example, an entry for this topic is added to the table of contents in the guide-books.rst file.

Below is a snippet from guide-books.rst showing the entry added to its table of contents for exporting shapefiles. Recall that the name of the reST file for the topic about exporting shapefiles is exporting-shapefiles.rst. Note that this table of contents has been simplified for this example and has only three entries in it.

.. toctree::

:maxdepth: 2

using-the-orb

working-with-databases

exporting-shapefiles

At this point, the reST file describing how to export shapefiles is complete and an entry for this file has been added to the table of contents in another file in the help project. The author could now run make html to build the help output and inspect the result. Always be sure to examine your output help topic as well as its entries in the table of contents and the index!