The VisualWorks Help System

This document describes the document structure and format for the help system provided in VisualWorks 7. This structure is essentially unchanged since this help system was introduced in 5i.1, except for two additions as noted in the text.

Note that the system is essentially a *prototype*, and is subject to change in future releases, without guarantee of compatibility. It is rough, but usable in its current form, even with its bugs and limitations.

Help File Contents

Help files must be well-formed, unstructured XML documents with specific header information and tags.

File Description Header

Each help text file must begin with these lines (though the specific title will differ):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet title="VisualWorks Tools" charset="UTF-8"?>
```

The second element tag may seem odd since help does not use stylesheets. In fact, if you look at the help files, additional attributes, specifying a stylesheet, are included. These are an artifact of how we create our files, and are simply ignored by the system.

However, the help system does look for this element, and uses the title attribute to identify the top level topic in the Help Browser. The title attribute must be there or the topic does not show in the browser and the topics are completely inaccessible.

Note also that the second element is “xml-stylesheet”. If you use FrameMaker to generate help, as we do, it adds this line as “xml:stylesheet” (a colon instead of a dash), which causes the help system to fail. (This is a bug.)

File Body

The remainder of the file occurs between <XML></XML> tags. There is no validation against a DTD or an XSL stylesheet, so the file is only expected to be well-formed.

There are, however, certain assumptions and requirements for certain tags, such as the presence and order of heading level elements.

Elements, in XML style, are specified by beginning and ending tags, in the form:

```
<tagname>some text</tagname>
```

This usage makes the help file format more like HTML, only with custom tags, than good, structured XML.

There tags to identify paragraph elements and character format elements.

File and Directory Organization

Help topics are organized hierarchically to three levels:

```
Topic
    Subtopic
        Help item
        Help item
        […]
    Subtopic
    Subtopic
    […]
Topic
[…]
```

Depth beyond that is generally a poor idea for help.

Each top-level topic and its nested subtopics are contained in a single file, so you need as many help files as you have top-level topics.

Help Directories

Help files are, by default, contained in subdirectories of the VisualWorks **`$(VISUALWORKS)/help/`** directory. The directory is specified in the Settings Tool.

The VisualWorks help files each have their own directory, though this is not necessary; they could have been placed in a single subdirectory.

When the help system is loaded into the image, it scans all subdirectories of **`help/`** to locate XML files, and extracts top-level topic headings. If you add or remove directories and/or files, use **Help → Rebuild Help Library** in the Visual Launcher to rescan these directories and files.

Topics and Heading Levels

A help file's top-level topic is specified twice, and both are required (this is a bug, but is manageable). The top-level entry listing in the Help Browser is specified in the xml-stylesheet header element, as described above. However, the help system starts building the help topic tree for the file from the Heading1 element. So, you end up with beginning lines like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet  title="Help File" charset="UTF-8"?>
<xml>
<Heading1>Help File</Heading1>
```

The Heading1 element can be empty, since the label is provided in the xml-stylesheet line, but it must be there. Leaving it empty, though, would be a mistake, and behavior of the file may be unreliable.

While the file can have more than one Heading1 element, only the first is used; all subsequent Heading1 elements are ignored. This is the reason you need a file for each top-level topic. (This, arguably, is also a bug, and is certainly not necessary, but that's the implementation.)

Subsequent topic levels are specified by Heading2, Heading3, and (added in VW7) Heading4 elements. When used in order, such as:

```
<Heading1>Topic Title 1</Heading1>
    <Heading2>Topic Title 2</Heading2>
        <Heading3>Topic Title 3</Heading3>
        <Heading3>Topic Title4</Heading3>
    <Heading2>Topic Title 5</Heading2>
        <Heading3>Topic Title 6</Heading3>
```

Will display in the browser as:

Topic Title 1
Topic Title 2
Topic Title 3
Topic Title 4
Topic Title 5
Topic Title 6

The system doesn't require that a Heading2 element precede a Heading3 element. This is a bug, and such a Heading3 element is promoted and displayed as if it were a Heading2 element.

Nesting elements (e.g., Heading3 within Heading2) is not supported and will not display properly.

Topic Text

Help topics are populated by text (paragraph) elements following Heading2 and Heading3 elements.

Heading1 elements cannot be populated with text (this is a bug). The Heading1 elements really should be what's shown in the browser, and there's really no reason that general level text should not be supplied.

The most basic text element is Body1, which is just plain text. Others included bulleted and numbered lists, code samples, and definition items and descriptions. These are all listed in a later section.

When a Heading2 or Heading3 item is selected in the Help Browser topics list, any text provided for the topic is displayed in the text pane. A Heading2 element need not have any text, if it is only used to group Heading3 elements. For a Heading3 element, there's no point in having one if there's no text.

Within a text element may be embedded character format elements, such as for bolding or italicizing text. For example, some body text for a Heading3 topic with a bolded word may be:

```
<Heading3>Help Topic</Heading3>
<Body1>This is just some <B>sample</B> help text.<Body1>
```

Character formatting tags are listed in a later section.

Note that if you include a new-line in a Body1 element, it is shown as a line break in the help display. This is a bug, since such line breaks should be ignored.

Topic Sort Ordering

The sort algorithm for help is really not up to snuff.

The only reliable way of sorting the order in which top-level topics are displayed is to do as we have for the VisualWorks help. Put each file in its own directory, and use a numeric prefix. The topics are then displayed in something like numeric order, but an odd one; 055 is read more like 5.5, so that 055 is listed after 05 but before 06.

Links, Anchors, and Such

The help system supports hypertext links, inline graphics (GIF), and code execution.

Hyperlinks and Anchors

Help uses the usual XML hypertext anchor elements to specify hyperlinks. To specify a target anchor, specify an ID:

```
<A ID="Launcher File Menu"></A>
```

To link to it, use an href attribute specifying the file name and anchor ID:

```
<Body1>Click <A href="filename.xml#id(Launcher File Menu)”>here</A>.
</Body1>
```

The parenthetical syntax for specifying the link target is an artifact of our use of FrameMaker, but is required by the help system. This required idiosyncracy reasonably construed as a bug, since it is not a standard XML (or URI) requirement.

Additional attributes, such as those contributed by FrameMaker, are acceptable, but unnecessary and ignored.

Inserting Graphics

You can insert a GIF graphic in the help file by specifying the graphic in a IMAGE element. For example:

```
<Body1><IMAGE href="bluespider.gif" /></Body1>
```

Not that this tag can include the closing mark at the end, rather than a separate closing tag. But, both work so you can also use:

```
<Body1><IMAGE href="bluespider.gif"></IMAGE></Body1>
```

Launching or Browsing Code

You can reference a class, and either browse it or launch it, by inserting a CodeExample element. If the class is an application (subclass of ApplicationModel) that responds to #open, it will be launched. Otherwise, a class browser will be opened on it.

For example:

```
<Body1> <CodeExample href="" >Random</CodeExample></Body1>
```

opens a class browser on class Random. The CodeExample element tags enclose the name of the class. The href=”” attribute is required, or the link will not show. On the other hand:

```
<Body1> <CodeExample href="WalkThru.pcl">
WalkThru.RandomNumberPicker</CodeExample></Body1>
```

loads the WalkThru parcel (WalkThru.pcl) and launches the RandomNumberPicker application. The parcel must be on the VisualWorks parcel path.

This feature is nice as far as it goes, but needs to be expanded for greater control and flexibility.

Contextual Help

A limited facility for adding contextual help has been added in VW 7, that allows you to invoke the help browser on a book and topic. For example, a menu item can execute:

```
HelpBrowser openOnBook: 'someBook' topic:' topicID'
```

The following command works in the current system:

```
HelpBrowser openOnBook: 'VisualWorks Tools' topic: 'Inspector'
```

The book title string is as specified in the file description header (see “File Description Header”):

```
<?xml-stylesheet title="VisualWorks Tools" charset="UTF-8"?>
```

The topic ID is inserted in the help file as a hyperlink anchor:

```
<A ID="Inspector"></A>
```

This facility will be expanded in the future.

XML Element Tags

This section lists and describes the supported help element tags. A lot of the specific tags are artifacts of our FrameMaker template. Alternative tags can be defined, and certainly will be when we impose a DTD.

Heading Elements

Element Tag	Description
Heading1	Top-level topic marker. One only, per help file. The top of the topic tree for the file. No topic text allowed at this level.
Heading2	Second level topic marker. Displays indented when Heading1 element is expanded. Topic text may follow this level.
Heading3	Third (last) level topic marker. Displays indented when Heading2 element is expanded. Topic text may follow this level.

Body Elements

Element Tag	Description
Body1	Plain text, full width paragraphs.
Bullet1	Unordered list, items preceded by a “bullet.”
CodeSample	Extended code sample. Indented and colored dark magenta. (alts: Smalltalk-Code-Table, Smalltalk-Code)
DefDscrp	Description of definition item, following DefItem.
DefItem	Item called out for further description.
StepCont	Numbered list, subsequent items
StepStart	Numbered list, first item.

Character Format Elements

Element Tag	Description
B	Bold
Code	Inline code sample. Dark magenta.
CodeEmp	Code, plus bold.
Emphasis	Bold
Glossary	Highlight for inline terms defined in the glossary. Red and italic.
NewTerm	Emphasis for a new term. Bold. (alt: New-Term)
Platform	Emphasis for non-Smalltalk code or entries. Bold. (alts: PlatformElement, PlatformElement-Table)
UIEntry	Emphasis for what is entered into a UI widget. Bold (alt: UI-Entry)
UILabel	Emphasis indicating a UI label. Bold. (alt: UI-Label)

Link Elements

Element Tag	Description
A	Hyperlink anchor.
IMAGE	Inline GIF image.
CodeExample	Launch or browse class.

Extending the Tags

All of the help element tags are defined as subclasses of class HelpElement. Browsing the elements already defined will make it clear how to add others.

The class name need not be the same as the tag. A concrete tag class defines a class method, #tag, that returns either a tag label or a shared variable holding tag labels.

For example, class Body specifies the tag 'Body1' as a String:

```
tag
    ^'Body1'
```

Class Bold, on the other hand, retuns the name of a shared variable which is initialized to an List of Strings:

```
tag
        ^BoldTags

initialize
        "self initialize"
        BoldTags := List new.
        BoldTags add:'B';
                add:'Emphasis';
                add:'NewTerm';
                add:'New-Term'.
```

Formatting specifications are given in several instance methods. The main methods are:

color
Specifies the text color of a paragraph or character format.

emphasis
Specifies the emphasis feature (e.g., bold or italics) of a paragraph or character format.

indentEmphasis
Specifies how much indent a paragraph element gets.

paragraphSpacing
A Boolean indicating whether or not space (leading) is appended below the paragraph.

There are additional, special-purpose methods as well, which you may browse.

Working with Help Files

While a help file is in use by the help system, you can edit it in VisualWorks, using the file editor. VisualWorks doesn't release the file, so you cannot edit it outside of VisualWorks without exiting.

For small changes to a file, though, you can do this:

1. Edit it in the File List tool or the File Editor, and save the changes.
2. Close the Help Browser.
3. Select **Help → Rebuild Help Library**.
4. Reopen the Help Browser and test.