SoftQuad HoTMetaL for Microsoft Windows

Published by

SoftQuad Inc.
56 Aberfoyle Crescent
Suite 810
Toronto, Canada M8X 2W4
Telephone: (416) 239-4801
Fax: (416) 239-7105
e-mail: hotmetal@sq.com

Document version

SoftQuad HoTMetaL
First Edition (June 1994)

SoftQuad Inc. makes no warranty of any kind with respect to the completeness or accuracy of this book. SoftQuad may make improvements and/or changes to the product(s) and/or programs described in this book at any time and without notice.

Copyrights and Trademarks

Notice

Contents

Getting started

This chapter tells you how to start up HoTMetaL and gives the basic information you need to get going with creating and editing files. If you are new to HoTMetaL or to SGML, you should certainly read this section, as it will help you get acquainted with the product and learn about the components and procedures you'll need to get your work done.

Purpose

HoTMetaL is an editor for creating files that can be read by graphical browsers (such as *Mosaic*) that are connected to the World Wide Web (WWW). The file format for such files is called HTML. The HTML format is actually a type of file format defined using the Standard Generalized Markup Language (SGML). In other words, all HTML files are SGML files (the converse is not true, however—there are many other file formats described by SGML, so most SGML files are not in HTML format.)

You can see what the HTML format looks like by invoking the **View Source...** command in Mosaic's **File** menu. HoTMetaL provides an easy-to-use, graphical, structured editor for creating files in this format.

Text and markup

Like most electronic documents, an HTML file consists of **text** and **markup**. (Markup is special codes inside the file that indicate how part of the file is to be processed: for example, a word-processor file would contain markup indicating typographical features such as the font and font size for various parts of the document.) In an HTML file the markup consists primarily of **elements**. Elements normally consist of a **start-tag** that is placed at the beginning of a section of the text, and an **end-tag** that is placed at the end of that section of text. In HoTMetaL, when you insert an element in the document, you are actually positioning its start- and end-tags. When you want words and phrases to be considered distinct elements, you surround them with tags. For example, a title in an HTML file would look like this:

```
<title>This is a title!</title>
```

(As you will see, when you are editing documents with HoTMetaL, you don't have to deal with tags on this level: for example, the start- and end-tags are represented on the screen by icons so that you don't have to type them literally, and HoTMetaL will insert both tags for you when you select a portion of the text to be surrounded by an element.)

When you are marking up a document in HTML format, you mark up parts of the document according to their function in the structure of the document. For example, there are different elements for divisions, headings, lists, list items, paragraphs, titles, and many other parts of a document's structure. It is important to realize that HTML markup is not primarily intended to indicate how the document is formatted typographically, it is meant to describe the document's structure. One of the reasons for using HTML (and SGML) is that the files can be readily re-processed in a different format by other publishing, browsing, database, etc., applications.

In addition to describing the structure of a document, some elements also describe the links to other documents that are found in an HTML document.

HTML documents are **structured** documents, that is, the elements must be arranged according to specific rules: otherwise, the document is considered invalid. When you are using HoTMetaL, you don't have to keep track of these rules yourself—HoTMetaL does it for you. One of HoTMetaL's most important features is automatic **rules checking**, which ensures that you do not violate the required structure as you are creating a document. As well, when you open or save a document, HoTMetaL checks that the markup is correct and complete.

Many HTML browsers have permitted a very loose, unstructured document format. Therefore, if you are editing existing HTML files, you may find that the structure that HoTMetaL imposes on documents is somewhat constraining. If you need to, you can relax these constraints using the **Turn Rules Checking Off** command in HoTMetaL's **Markup** menu. Because there is an emerging trend toward browsers that require a stricter document structure, we believe that you will find it to your advantage to create all your new HTML documents with HoTMetaL's default rules in force.

It will also be worthwhile to modify existing documents to conform to these rules. When you open a document, HoTMetaL will automatically turn off the rules checking if the document does not conform. If you click on **Turn Rules Checking On** in the **Markup** menu, you will get an error message describing the first error found in the file, and the insertion cursor will move to the location of the error. After the error has been corrected, you can repeat this procedure until all the errors have been removed.

To receive a rules file which is flexible enough to accept almost any HTML file, send e-mail to the following address:

```
loose-dtd-request@sq.com
```

Inserting markup

The most common operations you will carry out in order to add or change markup are:

- The **Insert Element...** command in the **Markup** menu inserts a new, empty element in which you can type text or insert other elements.
- The **Surround...** command in the **Markup** menu lets you surround a selection with an element: if some part of the document should be contained in a particular element, then you can highlight that portion and select this command to choose an element to surround it with.
- The **Change...** command in the **Markup** menu lets you change the markup: if you want to tag part of the document with a different element you can select this command to get a list of valid elements to replace it.
- Deleting markup: if you want to remove an element without deleting its contents, you should do the following:
  - Highlight the contents.
  - Invoke **Copy** or **Cut** from the **Edit** menu.
  - Highlight the element (including the start- and end-tags)
  - Press the Backspace key. This will delete the element.
  - Invoke **Paste** from the **Edit** . This will restore the contents of the element.

Special characters

If you type the characters '<' or '&' (both of which are special characters in SGML files) they will be replaced by small rectangular icons: **lt** and **amp** , respectively.

Rules files

The rules that determine how elements can be arranged in an SGML file are described by a set of declarations collectively known as a **document type declaration**, or DTD. HTML files are no exception to this. HoTMetaL reads a DTD in a special, binary form called a rules files which contains the same information as the DTD but in a different format, one which is more efficient for HoTMetaL to read. It is not necessary for you to know more details about DTDs and rules files. You just need to be aware that HoTMetaL uses a rules file called **html.mtl**: this file is located in a directory called **rules** in the HoTMetaL directory. Normally there will be no need for you to move this file; if for some reason you do, the new location of the rules file must be specified using the **rules_path** configuration variable—otherwise HoTMetaL will not be able to find it. For example:

```
rules_path=c:\wendy\rules
```

See the chapter *The configuration mechanism* for more information on configuration files.

If you are interested in seeing the DTD for HTML files, look at the file **html.dtd** in the **dtds** directory underneath the HoTMetaL directory.

The configuration mechanism

HoTMetaL's **configuration mechanism** allows you to modify certain aspects of HoTMetaL's behavior: find and save options, location of auxiliary files and directories, etc. HoTMetaL reads parameters called **configuration variables** from two configuration files: these are the file **sqhm.ini** in the HoTMetaL installation directory, and the file **sqhm.ini** in the Microsoft Windows directory (usually this is the **c:\windows** directory. Specific configuration variables are discussed in various locations throughout this manual. For full details, see the chapter *The configuration mechanism.*

Links

It is normal for HTML documents to contain links to other documents, which can be located anywhere on the WWW. These links are provided by **Universal Resource Locators** (URLs), which are identifiers that name the location and filename of a document, and the protocol used to access it. There are a number of elements available to you whose function it is to contain URLs: one such element is called "A" (you may think of this name as standing for "Anchor"). Rather than typing in the URL between the tags as text, you edit something called an **attribute** of the element: this is a piece of information attached to an element.

HoTMetaL has three commands for working with URLs:

- **Publish...** in the **File** menu is used if you want to change the URLs in the document from identifiers that refer to files on your local system to identifiers that refer to publicly-available files on one or more WWW servers.
- **Show Link and Context View** in the **View** menu is used to display the URL (if there is one) in the current element.
- **Edit Attributes and Links...** in the **Markup** menu is used to edit the URL (if there is one) in the current element.

More information on each of these commands may be found in that chapters on their respective menus.

Screen formatting

HoTMetaL provides screen-formatting capabilities which facilitate the document creation process by allowing you to assign distinctive styles to the elements in your document. This formatting is in effect only while you are editing the file in HoTMetaL. Another application may (in fact, almost certainly will) format the document differently.

Because SGML files are structured documents, setting a style for an element means setting it for all elements of that type.

The following kinds of typographical properties can be set:

- **Character-based properties**: font family, font size, font style, line height, justification, fill mode, and format type. All of these properties are set using the **Character...** command. (Font style allows you to adjust the font by making it bold, superscript, etc.; the fill mode determines how carriage returns are treated—in **fill mode**, they are treated like spaces, but in **no fill mode** they cause a line break; format type lets you choose whether an element appears inline or starts on a new line.)
- **Separation**: using the **Separation...** command, you can set off elements by adding space on top and on bottom. This command also lets you cause an element to be formatted as if it started with a tab.

Displaying icons

Using the **Show/Hide Tags** command in the **View** menu, you can cause the special character icons and the tag icons that represent elements to be visible or invisible.

Displaying a document outline

The command **Show Structure View** in the **View** menu displays a nested outline view of the document. You can cut, copy, paste, and navigate in this outline. For more information, see this command's documentation in the chapter *The View Menu*.

New lines

Although pressing the Return or Enter will put a "new line" into the file, Mosaic and other browsers follow the SGML whitespace rules, and therefore will ignore new line characters in some circumstances.

An overview of the menus

This section provides a summary of the main features.

- **File** menu: file manipulation, e.g., opening and closing files
- **Edit** menu: cutting and pasting; finding and replacing strings and patterns
- **View** menu: screen formatting; displaying different views of the document structure.
- **Markup** menu: inserting and changing markup; checking document conformance

- **Help** menu: on-line help.

How to run HoTMetaL

The usual way to run HoTMetaL is by double-clicking on the icon labeled **HoTMetaL** in the **HoTMetaL** Program Group. If this group has not been created, you should create it, as explained below.

The icon must have a **command line** associated with it. You can check the command line associated with this icon by using the **Properties...** command in the Windows **File** menu. You may also change the HoTMetaL command line by using the **Properties...** command. You may create a new *program group* using the **New...** command in the Windows **File** menu, then add HoTMetaL to that group (also using the **New...** command). Alternatively, you could add HoTMetaL to an existing program group. When adding HoTMetaL to a group, associate it with a command line like

```
c:\hotmetal\sqhm.exe
```

as appropriate. An icon will be provided for you by HoTMetaL.

Two other ways you can launch HoTMetaL from within a Windows session are:

- Using the **File Manager**, move to the directory where you installed HoTMetaL. Now double-click on the file **sqhm.exe**
- Use the **Run...** command in the Windows **Program Manager** with an explicit command line such as:

  ```
  c:\hotmetal\sqhm.exe
  ```

Creating and editing files

This section gives the basic information needed to start editing files with HoTMetaL.

Creating a new file

HoTMetaL comes up with a new, empty HTML document ready for you to use. You can also create a new file as follows:

- Click on the **New...** command in the **File** menu.

HoTMetaL brings up a new, empty file.

Editing an existing HTML file

If you already have an HTML file that you want to edit:

- Click on the **Open...** command in the **File** menu.
- In the dialog box that appears, choose the file that you want to edit.

Once you've done this, HoTMetaL opens the file and you can begin editing.

The File menu

The **File** menu contains commands for creating, opening, closing, saving, and publishing files edited with HoTMetaL. It also contains commands for importing and exporting documents so that they can be used by other publishing and database software.

New...

Creates a new, empty HoTMetaL file.

Open...

Opens a previously saved file.

HoTMetaL presents you with a dialog box allowing you to select a rules file. The dialog box is called a **file selection dialog**; a similar dialog box appears when you select the **Save As...** command.The structure and function of the file selection dialog for the **Open...** command is explained here.

The dialog box has several parts:

- A drop-down list box labeled **List Files of Type**. This list box lets you choose whether to display files with the default file extension (usually **.htm**) or display all files in the current directory.
- a text box labeled **File Name**

  This text box can contain a relative or absolute path name, which terminates in a file name or directory name, and can optionally start with a drive name. A file name (which may contain an extension) can contain the following "wildcard" (special) characters:

  - ***** (asterisk): matches any sequence of characters in a file name
  - **?** (question mark): matches any single character in a file name

  Such a pattern containing wildcard characters is used to filter the file names displayed in the list box directly below. If the **File Name** box contains a path name terminating with a directory name, then all file names in that directory are displayed and the pattern becomes '***.***'. The default pattern in this dialog box is '***.htm**'. You can type a pattern in the box directly or choose it from the **List Files of Type** list.

If the **File Name** box contains a path name that ends in a file name without wildcard characters, clicking on OK will cause that file to be chosen as the rules file for the new file being created.

> **Note: If you want to change the default file extension for this dialog box, you must set the new default extension with the 'import_ext' configuration variable.**

- an information field labeled **Directories**

  The current directory (including the drive name) appears underneath the label. The default directory is the one specified first by the **import_path** configuration variable.

- a list box underneath the **File Name** text box

  This list contains the files in the current directory that match the pattern in the **File Name** box. The list of files is updated whenever a new directory is chosen, and when you click on the OK button

  Clicking once on a file in this list causes the name to go in the **File Name** box. Double-clicking causes the file to be opened. This has the same effect as clicking once on the file name and then clicking on the OK button.

- a list box, on the right side of the dialog, underneath the **Directories** information field

  This box allows you to navigate in the directory structure of a drive by double-clicking on the directories shown in the list. The top level "directory" displayed is the current drive; if you double-click on this directory, its subdirectories are displayed, slightly indented to indicate the nesting relationship. If you click on one of these directories, its subdirectories will be displayed, and so forth. At any particular time the list will display the sequence of directories that you have navigated along, ending with the subdirectories of the last directory you selected.

- A drop-down list box labeled **Drives**. This allows you to choose which drive to navigate.

By pressing Tab or Shift-Tab you can make the **File Name** box, the list of directories, the list of files, or either of the buttons the active item in the dialog box. When either of the lists is active, you can select a list item by pressing repeatedly on the first letter of the item until it is selected. When the **File Name** box is active, you can enter text in it.

In summary: you may select a directory from which a file may be opened by using the list of directories, or by typing the path name in the **File Name** text box. You may choose a file by doing one of the following:

- double-clicking in the list of files
- selecting a file in the list of files and then clicking on OK
- entering the file name in the **File Name** text box and clicking on OK

If the file you selected is already open, you will receive a warning message informing you of this, and the file will become the current document.

> **Note: In this window, the default directory is the one specified first with the 'import_path' configuration variable. The default extension for a file to be opened is specified with the 'import_ext' configuration variable.**

Locating the rules file

There will sometimes be a **document type declaration** (DOCTYPE) at the top of an SGML file, specifying which rules file to use. This declaration is not mandatory with files being opened with HoTMetaL, because all HTML files use the HTML rules file. In fact, if the DOCTYPE declaration is missing, or is present but does not specify the HTML rules file, HoTMetaL will use the HTML rules file in the default location. If the DOCTYPE points to the HTML rules file in another location, that file will be used.

Error checking

As the file is being opened, HoTMetaL checks for fatal SGMLerrors. Fatal errors include start-tags without end-tags and vice versa, element names that do not exist in the rules file, and many other SGML errors. In such cases, HoTMetaL displays a message describing the problem and then opens the text file so that you can correct the errors.

If no errors are found, the file is formatted, checked once more for errors as if the **Turn Rules Checking On** command in the **Markup** menu had been selected for the new file. At this stage, non-fatal errors—SGML errors may be detected. Examples of these are incorrectly placed elements, and text at a point where no text is permitted. Errors of this kind do not prevent the file from being opened.

Finally the file is **validated**: this stage of error checking ensures that the SGML markup is correct and complete. This example illustrates the difference between **rules checking** and **validation**: if you open a file which has an HTML element that does not contain an FRONT element, rules checking will not complain, because you have not yet violated the rules file. Validation, however, will alert you to the fact that the required FRONT element is missing.

Open Template...

This command allows you to work with templates, which are predefined structures for documents. Templates are used as forms or document outlines that you can enter text into without having to insert any of the markup yourself.

Opening a template

To open a template, click on the **Open Template...** command. This brings up a file selection dialog box labeled **Open Template**. If you have a templates directory (see below) the dialog box will display the all files from that directory. Each file corresponds to a template: to open a template, just open it as you would any other file. The document name in the title bar will be the same as the template name, but with a number added to the first part of the filename: for example, the first time you open a template called **fax.htm**, the document will be called **fax1.htm**, the second time the document will be called **fax2.htm**, and so forth.

When the template file is opened you can enter text or elements into it, and later save the file. When you save the file, you will have to choose a file name—the name in the title bar is not automatically adopted. If you save this file in the templates directory, make sure you do not overwrite the original template file.

Creating your own templates

A number of templates are shipped with HoTMetaL, but it is expected that you will normally be working with templates that were created at your own site.

Templates directory

In order to work with templates successfully, a directory must be designated as the **templates directory**. This is a central location containing all of your template files, and it is the directory whose files are displayed when the **Open Templates...** dialog box comes up. By default, this is the directory called **tmplts** in the directory where HoTMetaL is installed.

If you want to use another directory for this purpose, you will have to name that directory using the **templates_path** configuration variable. For example:

```
templates_path=c:\susan\tmplts
```

If the default templates directory does not exist (someone has removed it), and no alternative directory is specified with the **templates_path** variable, the current directory will be used as the templates directory.

Creating templates

To create a template file with HoTMetaL, you should do the following:

- Create a document as you normally would.
- Invoke the **Save As...** command in the **File** menu.
- Choose a directory and filename. You can save the file directly in the templates directory, or move it there later.
- Click on the **Save As** button.

Installing a template

When the template file has been created, it should be saved in the templates directory (or you can save it elsewhere and move it later) so that it will be easily accessible from the **Open Template** dialog box.

Save

Saves the current file (that is, the file that is opened in the to disk.

HoTMetaL saves the document in the file name shown in the title bar at the top of the window.

You will be given a file selection dialog box in which to provide a name for the new file being created.

> **Note: The default directory is the first directory that is specified by the 'export_path' configuration variable.**

Save options

There are several **save options** that you may set by way of configuration variables, if you need to do so:

- You may choose to save a **document type declaration** (DOCTYPE), at the top of the file. The default may be set using the **export_doc_type_dec** configuration variable.
- If your file contains special characters (those outside the ASCII range 0-127), you may choose to have these converted to SGML **character references**. The default setting for this option (ON or OFF) may be set with the **export_convert_special_chars** configuration variable.
- You can choose the character(s) that HoTMetaL uses to mark the end of a line in the saved file. You may choose one of the values MAC, UNIX, and MSDOS, which will cause the end-of-line character to be **carriage return** , **line feed**, or **carriage return and line feed**, respectively The default end-of-line marker can be set with the **export_eol** configuration variable. If you don't provide a value for this variable, the default will be UNIX.
- For elements that are formatted in **fill mode** (see the documentation on the **Characters...** menu item) you can set the length of lines in the saved file by telling HoTMetaL to insert end-of-line characters after a specified number of characters. This option can be turned on or off by default with the **export_add_line_breaks** configuration variable.. The number of characters in a line can be set with the **export_max_line_len** configuration variable. HoTMetaL

will not cause a line break in the saved file to occur between an element and adjoining text.

The file will be validated prior to saving, and you will be notified of any errors that are detected. If you have rules checking turned on, such errors will prevent the file from being saved. If rules checking is turned off, a file with errors may be saved.

Save As...

Creates a **new** file whose content is the same as the current file, and closes the **current** file leaving the new file open.

HoTMetaL gives you the file selection dialog box with which to specify the name of the new file.

You should follow the same instructions for selecting a file or directory as were described in the section on the **Open...** command.

> **Note: In this window, the default directory is the one specified first by the 'export_path' configuration variable. The default extension for a file to be opened is specified by the 'export_ext' configuration variable.**

Note that any changes made to the original file since it was last saved will not appear in that file after the **Save As...** command is used (although they will, of course, be saved in the newly created file).

The save options that were specified for **Save** will also apply to **Save As**.

Close File

This command closes the current file. If the file has had changes made to it since it was last saved, you will be prompted to save the changes before closing it.

Preview

When you invoke this command it will save the current document to a temporary file and then launch Mosaic to display the file.

Publish...

Before a completed HTML document is moved to the server, all URLs should refer to documents that are available on some WWW server. (While the document is being created, they may refer to documents on your local system.) The **Publish...** command gives you the opportunity to edit all the URLs, modifying them if necessary.

When you invoke this command you will get a dialog box containing two text boxes. The first box (labeled **Change From**) contains a part of the URL that you want to change; the second box (labeled **Change To**) contains the string that you want to change it to. The default values in these two boxes are specified by two configuration variables: **publish_change_from** specifies the part of the URL that should be changed; **publish_change_to** specifies the new value for this part of the URL. For example, when you are creating a document the URLs may consist of local filenames such as:

file:///c|/rodney/orwell/homage.htm

When the document is placed on your server, you must substitute URLs that refer to documents that are available on your server or some other server. For example:

http://sqrex.sq.com/sqmosaic/orwell/homage.htm

If there were a large number of URLs for which you needed to change a local directory such as **file:///c|/rodney**to a a string such as **http://sqrex.sq.com/sqmosaic**, you could set your configuration variables as follows:

publish_change_from=file:///c|/rodney
publish_change_to=http://sqrex.sq.com/sqmosaic

This would cause the **Change From** text box to contain "file:///c|/rodney" and the **Change To** text box to contain "http://sqrex.sq.com/sqmosaic".

Finding and Replacing URLs

When you click on the Find button, HoTMetaL finds the next element that has an HREF attribute (this attribute represents a URL). The search starts at the insertion point (or selection) and wraps around to the top of the document if necessary.

When an element with an HREF is found, the insertion point is placed inside that element, and the document scrolls to its location. The field labeled **URL** at the top of the dialog box will display the value of the HREF (this field is not editable).

If the URL contains the text in the **Change From** box, clicking on the Replace button will change it to the text in the **Change To** box. The **URL** field will be updated to reflect the change.

Clicking on the Replace All button will make this change for all URLs in the document that contain the **Change From** text. A field at the bottom of the dialog box will indicate how many changes were made.

You can edit the **Change From** and **Change To** text if you want to perform substitutions other than the default one.

Exit

Quits HoTMetaL. If an open file has been changed since the last time it was saved, you will be prompted to save the file before exiting.

The Edit menu

The **Edit** menu contains commands to do the basic editing: cut, copy, and paste a selection, undo your last action, and perform find and replace operations.

Commands in this menu make extensive use of the *clipboard*. The clipboard is an area that holds parts of a document that have been cut or copied. It may contain markup. The clipboard may be used to transfer information between documents as well as within a document. Note that if markup is involved, it must comply with the rules file for the document into which the material is to be pasted.

Undo

Allows the effect of the **last** operation to be undone.

Most commands can be undone. There are, however, some HoTMetaL actions that cannot be undone:

- any command from the **File** menu
- scrolling and windowing commands
- text selection
- **Undo** itself
- **Show Structure View**, **Show Context View**
- Any actions performed prior to the last time the document was saved cannot be undone.

If you imagine a sequence of undo-able commands as a list, successive **Undo** commands will proceed through the list, starting at the most recent. Therefore, if you execute two **Undo** commands in a row you will undo the most recent action, and then undo the second most recent action. Note that since **Undo** is not itself an undo-able command, one **Undo** cannot undo another. This function is reserved for **Redo**, which is the inverse of **Undo**. (See the section on **Redo**.)

If you undo a **Copy** or **Cut** command, the previous contents of the clipboard will be restored.

Undo limit

The number of previous commands which can be undone is not limitless but is controlled by an **undo limit**. Once this limit is reached, each successive command will cause the oldest undoable command to be committed, that is, it will no longer be possible to undo that command. For example, if the undo limit is set to 1 and you **Cut** a selection and then **Paste** what you just cut, you will be able to undo the paste but not the cut. If the undo limit were two or greater, both the cut and paste could be undone. The default value for the undo limit is 10, but this can be changed by setting the **undo_limit** configuration variable.

Redo

Allows the most recent undone command to be redone.

Redo operates in a way similar to **Undo**: a sequence of **Redo** commands re-does the most recent redo-able commands (i.e., commands that have been undone) in reverse order. **Redo** and **Undo** are inverses of each other: the net effect of an **Undo** and its corresponding **Redo** is to cause no change to the document.

If an undoable action is performed after a series of one or more **Undo** commands then the **Undo** commands will no longer be redo-able.

To illustrate how **Redo** works, suppose you **Cut** a selection in a document, and then **Paste** that selection somewhere else. If you perform two **Undo** commands, first the **Paste** and then the **Cut** will be undone. If you then execute a **Redo**, the **Cut** will be redone. A second **Redo** will then redo the **Paste**.

Cut

Removes the current selection from the document and places it in the clipboard. Any previous contents of the clipboard will be erased. The selection can then be pasted.

This command is used when you want to remove a section of text that will probably be pasted in elsewhere, in the same or another document.

Copy

Copies the contents of the current selection into the clipboard and erases any previous contents. The document is left unchanged.

This command is used when you want to duplicate a portion of the document without erasing it. The copied selection can be inserted elsewhere using the **Paste** command.

Paste

Transfers the contents of the clipboard to the document. If the document contains an insertion point, the clipboard is pasted at that point; if it contains a selection, then the contents of the clipboard replace the selection.

The clipboard may contain markup. If the paste would result in an incorrectly marked-up document, you will be prompted to either cancel the paste operation or continue with rules checking turned off.

Find and Replace...

Allows text, elements, and patterns to be found and replaced.

You are presented with a dialog box that allows you to enter various values and parameters.

Specifying the search and replace strings

The **Find** text box allows you to specify a **search string** of text characters, tags, entity references, or patterns. If the document contains a selection when you invoke **Find and Replace...** the selected text will automatically become the search string. If the selected text is longer than 255 characters, it will be truncated. If it contains a markup icon, e.g., an element, comment, or entity reference, it will be truncated at the last character before the icon. A selection that starts with a markup icon will become a null search string, and therefore an existing selection cannot be used to cause an element to be the search string.

The **Replace** text box allows you to specify a **replace string** consisting of text characters, tags, or patterns with which you want to replace the search string.

The **Find In** text box allows you to restrict your search to a particular element.

The **Find**, **Replace**, and **Find In** strings are described in more detail below.

Command buttons

There are five buttons (including [Cancel]) along the bottom of the **Find & Replace** dialog box that allow you to carry out several search and replace operations.

Find

[Find] causes HoTMetaL to search through the document for the search string according to the various search parameters chosen. If you click on [Find] and the search is successful, HoTMetaL activates the document window, selects the text and/or elements that were found, and scrolls to the selection. If the search fails, HoTMetaL will beep. Also, a **Not found** message will be displayed at the bottom of the **Find & Replace** dialog box.

Text searches will not match if parts of the search string are found within different elements. If you are searching for *Fred and Barney*, but the word *and* is in a separate element (emphasized, for example), the search string will not be matched.

Replace

Replace replaces the current selection in the document with the replace string. This command is enabled only when you have a selection.

Replace then Find

Replace then Find replaces the current selection in the document with the replace string, and then resumes the search procedure. This command will be enabled only if you have a selection.

You would use this option if you wanted to manually examine each occurrence of the search string before doing a replacement: if you decide to perform the replacement, click on **Replace then Find** again; otherwise, click on **Find** to go to the next occurrence of the search string.

Replace All

Replace All replaces all occurrences of the search string with the replace string. This command automates the whole find and replace process, not giving you the opportunity to choose individual cases. It is possible that some of the occurrences of the search string which are found cannot be replaced, because this would cause an incorrectly marked-up document. If so, these occurrences are skipped over. After the operation has been completed, a message will be displayed in the **Find & Replace** dialog box showing how many occurrences of the search string were found, and how many were replaced. The insertion point will now be at the end of the last replacement.

Specifying search patterns as regular expressions

When the **Find Patterns** option is on (see below), the characters you type in the **Find** text box are interpreted as patterns by HoTMetaL: that is, the search string can contain certain special search characters which allow the search string to match a class of strings, or markup constructs. If your search string does not contain any special search characters, HoTMetaL will search for exactly the string you have typed. On the other hand, if the search string does contain special search characters, it is interpreted as a **regular expression** which defines a pattern of characters to be matched. For example, the special search character '**.**' (period) is used in the pattern

```
m...y
```

to match a sequence of five characters beginning with 'm' and ending with 'y', e.g., the words *money*, *marry*, *murky*, etc. A complete description of Author/Editor regular expressions and special search characters appears below.

Regular expressions and special search characters

The following characters are special search characters in a search pattern:

```
\ . * ? + ^ $ [ ]
```

In addition, the characters ‘&’ and ‘<’ are special when one or the other appears as the *first* character of the pattern.

If you want to search for any of these as ordinary characters when **Find Patterns** is turned on, it must be preceded by a backslash. For example,

```
\.
```

is used to match a period.

Regular expressions are sequences of ordinary characters and special characters, combined according to certain rules.

The following list summarizes how these special search characters are interpreted, and how regular expressions are formed.

- An ordinary character represents itself.
- A string beginning with a ‘<’, immediately followed by an element name, is used to match a whole element (which may include attributes and content, as discussed below).
- A period or dot, ‘.’, represents a single, arbitrary character (including a blank). So

  ```
  fo.d
  ```

  would match *food*, *ford*, *fond*, *fold*, etc. Similarly,

  ```
  s.o.
  ```

  matches *stop*, *shot*, *snow*, and so on.
- A single character, or a string enclosed in parentheses, followed by an asterisk, ‘*’, matches zero or more occurrences of that character or string. For example,

  ```
  l*ama
  ```

  would match *ama*, *lama*, *llama*, *lllama*, etc.

  ```
  b(an)*a
  ```

  would match *ba*, *bana*, *banana*, and so on. It is possible to combine the ‘*’ with ‘.’ to match arbitrary strings of characters. So

  ```
  s.*ch
  ```

  matches *search*, *such*, *stretch*, *stopwatch*, as well as *sch* and ‘*skip lunch*’. The search pattern represents strings that start with ‘*s*’ followed by zero or more occurrences of an arbitrary single character (it doesn’t have to be the same character over and over) followed by the characters ‘*ch*’. Since the period can match a blank space, this pattern can match a multi-word string.

- A single character, or a string enclosed in parentheses, followed by a question mark, '?', matches zero or one occurrences of that character or string. For example, to search for instances of both *color* and *colour*, you would use:

  ```
  colou?r
  ```

- A single character, or a string enclosed in parentheses, followed by a plus sign, '+', matches one or more occurrences of that character or string. For example, the following expression matches *ben*, *been*, *beeen*, and so forth, but not '*bn*'.

  ```
  be+n
  ```

- Regular expressions may be enclosed within parentheses for grouping.
- Two regular expressions separated by a vertical bar, '|', match any string that matches either of the regular expressions. For example, if you wanted to search for either *love* or *money*, you would use the expression:

  ```
  love|money
  ```

  In a more complex example, you could combine two regular expressions given above:

  ```
  s.*ch|fo.d
  ```

- A caret, '^', at the very beginning of a search pattern means that text will match the pattern only if it immediately follows markup. Such text must not be separated from the markup by white space. Anywhere else, the caret is not treated as a special search character (except in sub-strings, see below). For example, if you wanted to search for the word *Note* immediately following markup, you could use:

  ```
  ^Note
  ```

- A dollar sign, '$', at the very end of a search pattern means that text will match the pattern only if it is immediately followed by markup. Such text must not be separated from the markup by white space. Anywhere else, the dollar sign is not treated as a special search character. For example, if you wanted to search for the word *sub* immediately preceding markup, you could use:

  ```
  sub$
  ```

- A pair of square brackets

```
[ ]
```

around any string of characters defines a **sub-string** that matches any **one** of the characters between the brackets. For example,

```
an[dy]
```

matches *and* and *any*.

By contrast, a string of characters preceded by a caret, '^', within brackets, matches any character **not** in the string. For example,

```
th[^ei][a-z]*
```

matches any word that begins with *th*, which is not followed by an *e* or *i*. It would match *that* and *thought* but not *therefore* or *this*.

- A sub-string, within square brackets, of the form

```
[char1-char2]
```

matches any character in the range of ASCII characters beginning at *char1* and ending at *char2*. For example, the sub-string

```
[e-p]
```

matches any lowercase letter between 'e' and 'p', inclusive. The substring

```
[A-z]
```

matches any upper- or lower-case letter. Note that if searching is not in case-sensitive mode (see below), no distinction between lower case and upper case letters is made in character ranges. In this case, for example, the character range

```
[a-z]
```

would match any upper- or lower-case letter.

- An expression of the form

```
[^char1-char2]
```

matches any character *not* in the range of ASCII characters beginning at *char1* and ending at *char2*.

- A range can occur inside a sub-string. For example, the pattern:

```
[ac-fh]
```

matches any of 'a', 'c' through 'f', and 'h'.

- If you wish to use any of the characters '^', ']', or '-' as a regular character within a sub-string rather than as a special search character, there are certain rules you must follow.
  - The '^' character is special only if it occurs as the first character in a sub-string. Otherwise, it's treated as a regular

character. E.g.,

```
[joy^]
```

will match any of the characters ‘j’, ‘o’, ‘y’, and ‘^’.

- The character ‘**-**’ is a special search character if it occurs between other characters in the sub-string. If it occurs at the beginning or end of the sub-string, it is a regular character. For example, the sub-string

```
[a-]
```

will match ‘a’ or ‘-’.

- The character ’**]**’ terminates a sub-string unless it occurs as the first character. For example,

```
[]ab]
```

will match ‘a’ or ‘b’ or ‘]’. But

```
[ab]]
```

matches ‘a’ or ‘b’ followed by ‘]’.

None of the otherwise special search characters, including **‘\’** and **‘[’** has special meaning within a sub-string.

- If you surround a sub-expression in the search string by parentheses, ’(’ and ’)’, you can refer in the replace string to whatever this sub-expression matches. In general, an expression in the replace string of the form **\n**, where *n* is a number from 1 to 9, means “replace this expression with whatever the *n*th expression in brackets in the search string has matched”. For example, if the search string is

```
(.*)read
```

and the replace string is

```
\1ox
```

then if the search string matches *bread*, the found text will be replaced by *box*. This is because the sub-expression **(.*)** matched the letter *b*; the expression **\1** in the replace string means “replace this expression with whatever is matched by the *first* expression in parentheses in the search string”. Therefore *b* is substituted for **\1** and the replace string becomes *box*.

Here is a more complicated example: suppose the search string is

```
(v.*e) (v.*a)
```

and the replace string is

```
\2 \1
```

Now the search string may match the words *vice versa*. The first sub-expression, **(v.*e)**, matches *vice* and the second sub-expression, **(v.*a)**, matches *versa*. In the replace string, HoTMetaL replaces **\2** by what the second sub-expression in the search string matched, and replace **\1** by what the first sub-expression matched. Therefore the replace string becomes *versa vice*. The net effect of the operation is to replace an occurrence of *vice versa* with *versa vice*.

It is possible to nest sub-expressions. In this situation, the sub-expressions are numbered according to the order of occurrence of their left parentheses. For example, if the search string were

```
(a(bc)d)
```

and the replace string

```
\2 \1
```

the effect would be to find *abcd* and replace it by *bc abcd*.

The expression **\0** in a replace string refers to the entire string that was matched by the search string. E.g., if the search string were

```
fish
```

and the replace string were

```
gone \0ing
```

then an occurrence of *fish* would be replaced by *gone fishing*.

Elements as search patterns

A search string which begins with an open angle bracket, '**<**', followed by a valid element name matches an element of that name. If the search succeeds, the insertion point is positioned to the left of the start tag. It does not matter here or in other search options whether tags are visible or not (see **Show/Hide Tags** in the **View** menu). The name in the search string can optionally be followed by a closing angle bracket (**>**).

For example,

```
<META
```

matches the element that has the name META. Element names are not case sensitive in HoTMetaL, so

```
<meta
```

and

```
<META
```

would match the same elements.

In a replacement operation, if the search string and the replace string are both elements, one or more occurrences of the element in the search string will be changed to the type specified in the replace string. The contents of the element will be unchanged. The tag in the replace string cannot be followed by text; if it is, an error message will be displayed and the operation will not be performed.

If only the replace string is an element, the text that is found will be removed and replaced. Replacement will not occur unless the document contains a selection.

Searching for text within an element

It is possible to follow an element in a search string with a regular expression which represents a pattern which must be matched within the element. In this case the element must end with a closing angle bracket. For example,

```
<P>The
```

would match the word *The* anywhere within the element P. This is similar to the kind of restrictive searching that can be done using the **Find In** string but it can be used in conjunction with that feature to further restrict the search. In the last example, if the **Find In** string is set to

```
<OL
```

the word *The* would be matched if it appeared in a paragraph in a ordered list but not if it appeared in a paragraph in another context.

Error messages

If you have a badly-formed search or replace string, HoTMetaL will inform you of this with an error message at the bottom of the **Find & Replace** dialog box. This message will consist of a description of the error and the character position in the string at which the error occurred. Errors that will be reported include: invalid element or character names; unmatched parentheses and brackets in regular expressions; '?', '*', or '+' not preceded by any character; invalid character ranges.

For example, if you use the search pattern:

```
<QUAGMIRE
```

you will get the error message:

In Find string: Invalid element name at position 2

because the rules file does not contain an element called QUAGMIRE. This message also indicates that the error was detected at the second character in the search string.

Other search options

There are five options that can be set in searches. You may want to search forward or backward through the file, match only whole words, match upper- and lower-case exactly, employ wrapping, or perform pattern searching. These options can be used in combination and are turned on or off by clicking in the five check boxes in the **Find & Replace** dialog box. The defaults for these options can be set with the appropriate configuration variable. In this case, the boxes will be selected, or not, according to the configuration setting. Values set with configuration variables may be overridden during your editing session by clicking in the check boxes in the dialog box.

Whole Words

A search string may be part of a word or it may represent an entire word. Turning on **Whole Words** means that the search will match a sequence of one or more whole words only. For example, if HoTMetaL were told to look for *red* it would not find that string in *Fred*.

With this option turned on, the pattern:

```
a.*z
```

will match *a tough quiz* but not the first 12 characters (including the spaces) of *a tough buzzard*. The default for this option can be set with the **find_whole_words** configuration variable.

Case Sensitive

When case sensitivity is turned on, HoTMetaL will look for the search string exactly as you've typed it—matching upper case to upper case and lower to lower. With case sensitivity off, the program will find any variation: searching for *alice*, it would find *ALICE*, *alice* and *AliCE*. This option applies to patterns as well as text. The default for this option can be set with the **find_case_sensitive** configuration variable.

Backwards Search

HoTMetaL normally starts its searches at the insertion point (or the end of the selection) and moves *towards the bottom* of the file. **Backwards Search** indicates that you want the search to move from the insertion point (or the start of the selection) *back to the top* of the file. If wrapping is not enabled (see below), HoTMetaL does not wrap around the beginning or end of the file, so you should always make sure that you begin your search from the appropriate place in the file. The default for this option can be set with the **find_backward** configuration variable.

Wrap

When **Wrap** is turned on, HoTMetaL will wrap around the top or bottom of the file, depending on whether you are doing a forward or backward search. The default for this option can be set with the **find_wrap** configuration variable.

Find Patterns

This option allows you to turn on or off HoTMetaL's ability to find patterns. If **Find Patterns** is turned off, any special characters that you type in the find or replace strings will be treated as ordinary characters. The default for this option can be set with the **find_patterns** configuration variable.

Find In

One of HoTMetaL's more powerful search features is its ability to restrict a search to the contents of particular type of element. This means, for example, that you could use **Find and Replace...** to check that a word is in uppercase letters whenever it appears in a paragraph, but in upper and lowercase when it's part of a title.

The **Find In** text box is used to specify the element that you want to restrict searching to. The format for this string is identical to that for the search string when searching for an element, except that the element name can't be followed by text.

Find Next

Performs a search for the search string specified in the previous **Find & Replace** dialog box. Once a search string has been specified, this command has the same effect as clicking the Find button in the **Find & Replace** dialog box.

The View menu

Overview

The **View** menu contains various commands to format and structure the window when editing an HoTMetaL document.

Styles

Because HTML files are structured documents, setting a style for an element means setting it for all elements of that type.

Styles files

All formatting information set with the **Character...**, command is stored in the file **html.stl**. This file is called a styles file. The styles file usually has the same name as the rules file, but with the filename extension replaced by the styles file extension. By default this is ‘ **.stl**’, but you can change it with the **styles_ext** configuration variable.

When an HoTMetaL file is created, HoTMetaL looks for the styles file in the **styles path**, that is, the set of directories named by the **styles_path** configuration variable. If it finds the styles file, then the formatting information contained in that file is used for the current HoTMetaL file. If the styles file is not found, then a new styles file, containing the default formatting information, is created in the first directory in the styles path. Information in the styles file is updated whenever the HoTMetaL file is saved.

The same styles file is used for all HoTMetaL files created with a particular rules file: when formatting information is changed for one such HoTMetaL file, it changes for all HoTMetaL files created with the same rules file.

Numerical parameters

The **Character...** command allows you to set numerical parameters. These may be set from a menu or entered directly by the user in a text box. As appropriate, these values may be absolute, relative, or expressed as a percentage of some base value (this is explained below). The following units may be used:

- centimeters (cm)
- inches.
- machine units
- millimeters (mm)
- picas
- pixels
- points

A pixel is the same as a point, and a machine unit is 1/16 of a pixel.

You may use any unit wherever you are allowed to enter values. For example, point size may be expressed in points, inches, picas, etc. Descriptions of individual commands will indicate which kinds of values (absolute, relative, percentage) may be used.

Units may be specified by giving the full unit name, the abbreviation (mm or cm), or the first few letters of the unit name, as long as that specification is unambiguous.

Examples

The following are examples of valid settings:

- 1 inches
- 1.45 i
- 2 mm
- 2 milli
- 3 pix
- 6 points
- 2 po
- 2 pica
- 2pica

Relative and percentage settings

Relative settings specify an amount to be added to or subtracted from a base setting: they have the same format as absolute settings, but are prefaced by a ‘+’ or ‘-’ sign. As well, you may set a value to be **Adopt Current**, which means that the value is to be inherited from the surrounding element. Some valid settings are:

- -2.67 inches
- +3 picas
- Adopt Current

– ac (same as Adopt Current)
– +0 (same as Adopt Current)

Values may also be expressed as a percentage by affixing a percent (%) sign, or the word 'percent' or a suitable abbreviation e.g.,

– 100% (same as the default value)
– 150% (1.5 times the default value)
– 243 per (2.43 times the default value)

Show/Hide Tags

If you select **Show Tags**, the start- and end-tags in your document will appear on the screen as small tag icons, and special characters will be represented as rectangular icons containing the character name; if you select **Hide Tags**, the tag icons will be hidden, and the special characters will appear in text form. This command is toggled: if you choose **Show Tags**, the menu item changes to **Hide Tags**, and vice versa. The tags which point to the right are **start-tags**, indicating the beginning of an element, while those which point to the left are **end-tags**, indicating the end of an element. When you create a new HoTMetaL document, tags will be visible by default.

Show Link and Context View

This feature allows you to display a window showing the sequence of nested elements which terminates at the current insertion point or selection.

This window will not show the structure of the entire document (see **Show Structure View** below) only the hierarchical context containing the current position. That is, it displays the sequence of open tags at the current position. Any attributes, representing links, that have been set for elements in the sequence, will be displayed.

You can bring up a different context view for each open HoTMetaL document.

Show Structure View

This command brings up a window that shows the structure of the entire document.

You can bring up a different structure view for each open HoTMetaL document. Once a structure view is displayed, it remains visible, even if you change documents, until it is dismissed.

The structure view shows the hierarchy of the document in a plain, unformatted manner. Each line in the structure view represents one element in the document. Each line shows a start tag, an end tag and possibly some text between them. The indention of the line indicates the level at which the element is nested. The text cannot be edited and typing is not recognized, but you can use all the editing commands from the **Edit** and **Markup** menus.

Editing in the structure view

Since text entry is not permitted in the structure view, almost all editing is done in the formatted view. The structure view is useful when rearranging whole elements or when creating an outline for your document.

To select an element in the structure view, you can either click to the left of the start tag icon and drag the cursor across the icon, or, click *twice* to the left of the start tag icon. Multiple elements can be selected by clicking the mouse to the left of a start tag and then dragging down to select successive elements.

You may insert, remove, etc., elements while in the structure view. Any of the valid commands that you select will apply to the contents of the structure view.

Any editing you do in the structure view will appear in the formatted view provided it is displaying the same portion of the document.

Expanding the view of elements

Clicking on an element icon allows you to open and close elements, that is, show and hide their contents. When an element is open, all the elements it contains (but not their sub-elements) are shown as indented lines following that element's line. (To see the contents of sub-elements, open these in turn.) When an element is closed, access to its contents is not possible because the contents are not displayed in the window. Since you can insert elements in the structure view, you can use this feature to build an outline (structure) for your document before you begin to type in the text.

Show Image

This command allows you to display a file (usually a bitmapped graphic file) referred to by a URL in the current element. To view a file, invoke this command when the insertion point or selection is inside the desired element.

Show Image uses the following mechanism to determine how to process the file—in this explanation, assume that the URL is:

file:///c|/rodney/orwell/george.gif

1. HoTMetaL reads the filename (the last part of the URL). In this case it's "george.gif"
2. HoTMetaL reads the file extension (the part after the dot). In this example the extension is "gif". This extension should indicate the format of the file: we would expect this file to be in GIF format.
3. HoTMetaL scans the defined configuration variables, looking for one called **view_***extension*, where *extension* is the file extension read in the previous step. In this case, it would look for a variable called **view_gif**. The value of this variable (if it's defined) should be a command that processes the file, for example:

view_gif=c:\windows\pbrush.exe

If this variable is not found, **Show Image** terminates with an error message.

4. If the variable is found, the filename specified in the URL is appended to the command, and the resulting command line will be executed on your system. In this case, the command line would be:

   c:\windows\pbrush.exe george.gif

   (This invokes the Windows *Paintbrush* program.)

If you intend to use **Show Image**, you should ensure that before you invoke HoTMetaL, the necessary configuration variables (**view_gif**, **view_tif**, **view_jpg**, etc., are defined, and any programs that these variables refer to are available on your system (in particular, ensure that they are in your executables path unless you use a full path in the command name).

Since this command uses only the filename part of a URL, the URL can specify a file on any server, provided that a file of the same name exists on your local system.

Load Styles...

This command allows you to load a styles file in text format. The same kinds of screen formatting parameters as are stored binary styles files will be read from this file. When this command is invoked, a file selection dialog box appears, allowing you to choose a styles file. The current directory is the one listed first by the **ascii_styles_path** configuration variable.

> **Note: The binary styles file is updated with the formatting information loaded from an ASCII styles file when the HoTMetaL file is saved.**

> **Note: If you copy an ASCII styles file from one environment to another, you may experience formatting anomalies if the environments support different fonts. In this case, you may wish to open the ASCII styles file as a text file, change the font names as appropriate, and then reload the file.**

Character...

This item allows you to change character-related formatting properties such as font, font size, and justification.

You can change the format assigned to an element in the document at any time. When you invoke the **Character...** command, HoTMetaL gives you a dialog box allowing you to set formatting parameters for the current element (the one the insertion point or selection is in). When you change these parameters, they will change for all occurrences of the current element. If the insertion point is not inside any element, HoTMetaL gives you a dialog box to set the default format. If you move the cursor to a different element while the **Character** dialog box is on the screen, the dialog box changes to reflect the formatting of the new element.

The name of the element currently being formatted appears in the lower left corner of the dialog box. (The element is PARA in the illustration.)

Font Family

The drop-down list box for **Font Family** shows the current choice. You can specify the font family by name or by adopting the font of a surrounding element. To select a new font family, position the cursor on the **Font Family** arrow. Click and drag down the list of the font family names available on your system. Release the mouse button over your choice.

Adopt Current

Sometimes you may want to change the font family you're using on the screen for many related elements. Changing the **Font Family** for every element would be time-consuming and tedious. You can allow changes to be inherited by taking advantage of HoTMetaL's ability to pass format settings from one element to the next.

For example, if the insertion point is surrounded by the following tags:

HTML BODY P

and you set the **Font Family** to **Adopt Current** in the BODY and P elements, then whatever font family you establish in the HTML element will ripple through the entire structure.

Font Size

Selecting the type size is similar to selecting the font family. You can open the **Font Size** drop-down list box and choose any size (including **Adopt Current**) shown. You may also enter a size in the text box to the left of the arrow. This size may be an absolute or relative value. HoTMetaL will choose the closest available font size. Relative values are added to the point size of the surrounding element.

Font Style

The check boxes under the **Font Style** heading allow you to add style variations to the font family. The check boxes can be set individually or in any combination. If you want the style to be the same as the style for the surrounding element, click on the **Adopt Current** check box only.

Bold, Italic

As an example, consider a document with three emphasis elements: B , I , and EM . So that the contents of these elements will stand out, you could set the **Font Family** to **Adopt Current** and set the **Font Size** to be the same for each of these elements, make them all **Inline** (see below) and then set B to **Bold**, I to **Italic**, EM to **Bold** and **Italic**. These emphasis elements would then cause their contents to be the same font and size as the surrounding text but each would have a different style.

Superscript, Subscript

Superscript and **Subscript** raise and lower, respectively, the baseline of the font. Superscripted text is raised so that its baseline is one-third of its ascent above the baseline of the containing element. (The *ascent* is roughly the distance from the baseline to the top of an uppercase letter.) Similarly, the baseline of subscripted text is lowered to be one-third of the ascent below the baseline of the surrounding element. To make effective use of these style options, either choose a font size that is smaller than the size of the containing element or ensure that the line spacing of the containing element is large enough that the tops of the super-script and bottoms of the sub-script characters are not cut off.

Toggle

When **Toggle** is selected as part of the font style, the other style settings are turned off if they are turned on in the containing element. For example, an inline element whose font style is set to **Bold** and **Toggle** will appear as bold text within plain (Roman) surrounding text and as plain within bold surrounding text.

Adopt Current

Adopt Current means that the font style options of the containing element are adopted in addition to those explicitly set for this element type. For example, if **Adopt Current** is the only option selected, then the font style for an element of this type will be identical to the font style of its containing element. Another example would be an element whose font style is set to both **Adopt Current** and **Italic**. Then the element's text will appear with a font style of italic within an element containing plain text and as bold-italic in bold text.

Underline

Underline allows you to underline the text of the element.

Line Height

The **Line Height** specifies the vertical height from the bottom of one line to the bottom of the next line in the same element. This value should normally be set slightly larger than the font size so that the lines won't appear clipped.

You may select single, double, or triple spacing from the drop-down list box labeled **Line Height**. Double spacing gives twice the line height of single spacing; triple spacing gives three times the line height of single spacing. A value may also be entered directly in the text box to the right of the arrow. This value may be absolute, relative, or a percentage.

A percentage line height is understood as a percentage of single spacing. I.e., 100 percent is the same as single spacing, 150 percent is 1.5 times as high as single spacing, and so on.

If you give a relative value then the line height is equal to the single spaced line height plus or minus the amount specified.

An absolute line height should be at least as large as the point size: otherwise, the lines will overlap. A value of about 1.2 times the point size would be normal.

If you select **Adopt Current** for the line height, then the element you are formatting will assume the absolute line height of its surrounding element. So if the surrounding element has point size 12 and single line spacing, and you choose, say, a point size of 24 for the current element, it will have the same line height as if it were only in 12 point type, because that is the line height of the surrounding element.

Line height defaults to **Adopt Current**.

Justification

HoTMetaL offers four styles of text justification plus the ability to inherit the justification style from the surrounding element. You choose the one you want by clicking on the appropriate icon. The styles are mutually exclusive: you may choose just one.

The choices are:

- **Left** (flush left, ragged right): All the spaces between words are the same width, and the text is aligned on the left leaving the right-hand edge uneven.
- **Right** (flush right, ragged left): All the spaces between words are the same width and the text is aligned on the right.
- **Centered**: all the spaces between words are the same width and the lines are centered on the midpoint of the line length.
- **Both** (flush right and left—often called **justified**): Spaces between words are adjusted to force both left- and right-hand edges of the text to be even.
- **Adopt Current**: the justification style is inherited from the surrounding element.

Fill

This group of choices allows you to specify whether HoTMetaL should treat return characters as a space or as the end of a line. When **Fill** is selected, the lines are broken at the last space before the right indent. If you type Return a return character is inserted but is treated as though it were a space. When **No Fill** is selected, HoTMetaL will not fill the lines—when you type Return , a new line will be started.

For example, poetry and centered headings would normally be in **No Fill** mode, while the contents of a paragraph would be in **Fill** mode.

Saving filled elements

When you save a file, you may choose to have line breaks inserted after a certain number of characters in all elements that are in fill mode. See the description of the **Save** command in the chapter on the **File** menu for more details.

Format Types

For the purposes of screen formatting, the elements are divided into two broad categories: **Block** and **Inline**. You set the category for each element by clicking on the appropriate option button in the **Format Type** group.

Block

Block elements:

- always start on a new line if there are any end-tags or text on the current line.
- cause any element or text which follows them to begin on a new line.
- can have all the formatting options set: justification, font family, size and style, vertical and horizontal spacing, and tab settings.

Inline

Inline elements:

- cause no line break either before or after the element. This format might be used for an emphasized word or phrase or a short quotation.
- can have only certain formatting options set: font family, size, and style. The other options are inherited from the surrounding element.

Separation...

This command lets you set the amount of vertical space that separates an element from neighboring elements. Move the insertion point inside an instance of an element you want to format, and click on the **Separation...** menu item, or click inside the desired element when the **Separation** dialog is already on the screen.

Top and bottom space

Top and bottom space control the vertical separation between elements. They are available only for block elements.

Top Space determines the minimum amount of vertical white space that must precede the element. If the element before this one has a **Bottom Space** value, the actual separation will be the greater of the current element's top space and the preceding element's bottom space.

The top space can be specified as an absolute amount or as a percentage of one line height at the current line height. For example, if you want 1 1/2 lines of white space between paragraphs, set the top space to 150%. The top and bottom space separations are added to the normal line spacing, so if two successive elements have a bottom separation of 0 and top separation of 0, respectively, the baseline of the last line of the first element and the first line of the following element will be one line height apart.

Bottom Space determines the minimum amount of vertical white space that must follow the block. The actual separation is the greater of the current element's bottom space and the next element's top space. The value is specified in the same way as for **Top Space**, above.

Tabbed elements

An element which has **Tabbed** turned on is formatted as if it started with a tab. (See **Ruler...**, below, for information on setting tab stops.) This command does not actually insert a tab into the text. Both block and inline elements may be designated as tabbed. This option is useful for displaying simple tabular material.

The Markup menu

The **Markup** menu contains commands to insert and edit markup.

Insert Element...

This command creates a new element by inserting a new pair of start- and end-tag icons.

When you invoke this command, you are given the **Insert Element** dialog box, which contains a list of names of the elements permitted by the rules file at the location of the insertion point (or selection) in the document.

The list of elements

The left column of the list box contains an alphabetical list of element names; the right column contains a phrase describing the corresponding element. The first element (if there is one) **required** by the HTML rules file at this point in the document will be followed by the '<' character.

Inserting an element

To select an element, click once on the line containing the element name, or type the first character of the element name repeatedly until the correct element is selected. Then click once on the **Insert Element** button to insert it. Alternatively, you can double-click on the line that contains the element name in the scroll box.

If the document contains a selection rather than an insertion point, the selection will be replaced by the inserted element.

Restrictions

Insert Element... will be disabled and the menu item grayed-out if there are no elements that can be inserted at the insertion point or current selection without creating an incorrectly marked up document. Often the command will become enabled if rules checking is turned off (see **Turn Rules Checking On/Off**), but there are some circumstances in which you will never be allowed to insert elements. These include trying to insert an element in:

- an element whose content type is CDATA or RCDATA, which do not allow element start-tags to be recognized.
- an element which is defined in the rules file as being empty.

The characters '<' and '&', which could be interpreted as markup by the HoTMetaL parser, will be converted to special characters.

Required elements

The **Insert Element** dialog box includes a check box labeled **Include Required Elements**. When this box is selected, inserting an element causes the first required subelement (if there is one) to be inserted as well. This option may also be set with the **include_required_elements** configuration variable. This process is recursive, so that if the required subelement itself has a required subelement, this element will also be inserted, and so forth. Note that this does not mean that all required subelements of the current element will be inserted, only the first required subelement (and its required subelements).

Change...

Changes the type of the current element.

HoTMetaL presents you with a list of elements that can replace the current element and still leave the document correctly marked up. You may choose the appropriate element from the list in the ways described above for the **Insert Element...** and **Surround...** commands.

The *current element* is the innermost element containing the insertion point or selection. If the selection consists of an entire element (that is., including its start- and end-tags) then the current element is the one *containing the selected element*, not the selection itself.

The command will be disabled and the menu item grayed-out if there are no elements that can replace the current element without creating an incorrectly marked up document. Often the command will become enabled if rules checking is turned off (see **Turn Rules Checking On/Off** under the **Markup** menu), but there are some circumstances in which you will never be allowed to change the current element. If the new element requires a content of CDATA, RCDATA, or EMPTY, then, even when the rules aren't being checked, the replacement will not be allowed unless the existing content matches those types.

Edit Attributes and Links...

This command lets you view and edit URL and other attributes of the current element.

To edit the attributes of an element, set the insertion point inside that element (i.e., make it the current element). Then use the **Edit Attributes and Links...** command. You will be presented with a dialog box that can be used to edit the attribute list.

The dialog box will change depending on the type and number of attributes that have been defined for this element in the rules file. In every case, however, the dialog box will consist of several lines, one per attribute. Each line will consist of:

- The **name** of the attribute.
- A drop-down list box or a text box, depending on the type of attribute. If the attribute value is defined as being one of a list of values, then the line in the dialog box line for that attribute will containa drop-down list box giving the possible values. Otherwise, you will see a text box.

If you are editing the URL you should modify the element's HREF or SOURCE attribute, as appropriate. Insert the value of the URL in the text entry box for this attribute and click on **OK** or **Apply** .

Turn Rules Checking On/Off

This command toggles the state of rules checking in HoTMetaL.

When rules checking is on, HoTMetaL uses the rules file to ensure that the document being edited will be correctly marked up. While this checking is not complete, it will nevertheless catch and prevent most markup errors.

HoTMetaL prevents markup errors in a number of ways.

- The commands which could cause errors are disabled. For example, the **Surround** command in the **Markup** menu will be grayed-out if the document would not be correctly tagged after the selected content was surrounded by any element.
- A restricted list of elements is presented. For example, the **Insert Element...** command will only present a list of those elements which will leave the document correctly tagged after the insertion.
- An opportunity is given to cancel a command before any damage is done. For example, if a **Paste** operation would leave the document incorrectly tagged, HoTMetaL will present a warning giving the choice of canceling the paste or completing the command after first turning rules checking off.

Rules checking is normally desirable, since it greatly reduces the chance of making markup errors. However, there are occasions when checking can get in the way of the job at hand. Most commonly this happens when the operation that you are performing involves two or more steps, and one of those intermediate steps will leave the document temporarily incorrectly marked up.

When the rules are not being checked, the commands which were previously disabled will usually become enabled; for the exceptions, see the documentation on the individual commands. This means that you will be able to create an incorrectly tagged document, and therefore you should leave the rules off only as long as you need to.

When you choose **Turn Rules Checking On** from the menu, the menu item changes to **Turn Rules Checking Off** to indicate that this state can be toggled. When you turn rules checking back on, HoTMetaL will quickly scan your document to make sure it is correctly tagged. If it isn't, HoTMetaL will present a warning describing the problem and the insertion point moves to the location of the error. Rules checking will remain off. Select **Turn Rules Checking On** again after the problem is corrected.

The Help menu

About HoTMetaL...	Displays the HoTMetaL copyright notice.
Upgrade Information...	Provides details of how to upgrade from this product to HoTMetaL PRO.

The Window menu

This menu contains commands for moving between open files and for arranging document windows on the screen.

Next

This command lets you cycle through the open files. HoTMetaL orders the open files according to the time they were opened. When you invoke **Next**, the file that was opened most recently after the current file, and has not yet been closed, becomes the current file. If the current file is the most recently opened file, then the least recently opened file becomes the current file.

You may also make a file active by clicking on its document window, or by selecting it from the list of open files that appears at the end of the **Window** menu (see below).

Previous

This command is similar to **Next**, but it makes the file that was opened most recently before the current file, and has not yet been closed, the current file.

Cascade

This command causes all the document windows (including those for context and structured views) to be cascaded, in the order in which the files were most recently active. That is, the active file is displayed at the front, the file that was next most recently active is displayed behind it, slightly above and to the left, and so forth.

Tile

This command causes all the document windows (including those for context and structured views) to be tiled. That is, the windows will be arranged in a column of non-overlapping, equal-sized windows down the frame. The active document will be on top. If there are more than three open documents, the windows will be arranged in several columns, with the active document at the top left.

Tile vertically

This command is similar to **Tile**, but causes all the document windows to be arranged in a row of non-overlapping, equal-sized windows across the frame. The active document will be on the left. If there are more than three open documents, the windows will be arranged in columns, with the active document at the top left.

Arrange icons

If some document windows have been iconified, this command will cause the icons to be arranged in a row at the bottom of the frame.

Filenames in the Windows menu

Every open document (including context and structure views) will cause a menu item consisting of the document name to be added to the **Windows** menu. Selecting the document name will cause that document to become active. Each document name is preceded by a number. This number is the *mnemonic* for the menu item, that is, typing the number while the **Windows** menu is open is the same as selecting the menu item.

The configuration mechanism

HoTMetaL's configuration mechanism (sometimes called the *sqconfig* mechanism) lets you modify HoTMetaL's behavior to suit your needs and those of your site.

There are two types of information in the configuration files. There are variables which control HoTMetaL's behavior, and there are variables which give the location of files or directories.

Configuring HoTMetaL

There are many aspects of HoTMetaL's behavior that you can configure to your personal needs or those of your site. For example, you can control default options in the **Find and Replace...** dialog box, set options for the **Save** command, set headers and footers for printed pages, and specify the locations of various auxiliary files. Configuration is done using a mechanism called the *sqconfig* mechanism, or just the "configuration mechanism". You can run HoTMetaL without any problems using the default configuration, but at some point you may prefer to customize. This will be particularly true if several people will be using HoTMetaL on the same PC.

Configuration files

The configuration files read by HoTMetaL are the file **sqhm.ini** located in the directory where HoTMetaL is installed and the file **sqhm.ini** in the Microsoft Windows directory (usually **c:\windows**). These files contain configuration parameters called *configuration variables*. Variables set in the file in the Windows directory take precedence.

If a variable is not set in any configuration file, but is set in the environment, then the setting from the environment is used. If there is no setting in the configuration files or in the environment, then the built-in default value (if there is one) is used. If there is no default value, the variable is undefined. In summary, the value of an *sqconfig* variable is taken from the following sources, in the order given below:

1. The configuration files.
2. The environment.
3. The built-in default.

A suggestion: base and personal configuration files

The following arrangement is one suggestion about how you can use configuration files. It involves using two files: a **base** file for variables that don't change very often, and a **personal** file for variables that change more frequently, are experimental, or are employed by a single user.

Base file

The base file should be used as a system configuration file. It should contain the settings that you want to be used as defaults. A parameter should be changed in this file only if you decide that its default value should change for everyone using HoTMetaL on a specific PC. The file **sqhm.ini** in the HoTMetaL directory should be used for this purpose.

Personal file

The personal configuration file should be used if you need to override some of the settings in the base file without modifying the base file. For example, you may want to make temporary changes to some of the parameters, or to specify parameter values that are used only by you (or another individual user), rather than by everyone who uses HoTMetaL on the computer. If your PC has only a single user, you may choose not to use the personal file at all. Or, you could use it only for making temporary changes to the configuration. The file **sqhm.ini** in the Windows directory should be used for this purpose.

Setting parameters in the configuration files

You do not need to change any configuration variables unless you wish to customize the HoTMetaL configuration.

A variable is just a name which is assigned some value. You can change these variables by simply editing the configuration files, as appropriate, and making the desired changes. **The configuration files are read by HoTMetaL on start-up, so the changes you make will take effect the next time you run HoTMetaL—they will have no effect on a currently-running HoTMetaL. If you need them to take effect immediately, you will have to exit HoTMetaL and restart it.**

Basic format for setting variables

Variables are assigned values by putting lines of the following form in the configuration files:

```
variable = value
```

For example:

```
undo_limit=50
```

undo_limit is a configuration variable that specifies the number of successive commands that can be undone or reversed with HoTMetaL's **Undo** command. The default built in to HoTMetaL is 10: to raise this to 50, you would set the variable as in the example.

You may put spaces or tabs on either side of the equal sign for readability. Also, if you prefer, you may substitute a colon (:) for the equal sign:

```
undo_limit:50
```

The effect is the same.

You should *not* have any “white space” (spaces or tabs) at the end of the line.

If you don’t want to set a particular variable, then you can do this by omitting or deleting any settings of that variable in the configuration files. Alternatively, you can “comment out” settings in these files by inserting the ‘**#**’ character as the first character on all lines containing such settings: HoTMetaL will ignore such lines. For example:

```
#undo_limit=50
```

You should **not** try to give variables a null value, e.g.,

```
undo_limit=
```

or

```
tag_font_name=""
```

You should take care to use legal values for all the configuration variables. Otherwise, HoTMetaL may behave in unexpected ways.

Referencing one variable from another

You can use the value of one configuration variable when assigning the value of another variable. For example:

```
my_name_is=rodney
rules_path= d:\${my_name_is}\rules
styles_path = d:\${my_name_is}\styles
```

The expression:

```
${my_name_is}
```

is equal to the current value of the configuration variable (**my_name_is**) between the ‘{’ and ‘}’. So this expression is equal to “**rodney**”. When HoTMetaL evaluates **.rules_path** in the last example, it substitutes **rodney** for **${my_name_is}**, so that the value of **rules_path** becomes “**d:\rodney\rules**”. HoTMetaL performs this substitution when it *uses the variable*, not when it reads the configuration files at start-up

You can use the same notation to cause HoTMetaL to read a DOS environment variable. For example, you could set an environment variable (at the DOS prompt or in the **autoexec.bat** file):

```
set MY_NAME=rodney
```

A configuration file could have the following settings:

```
rules_path= d:\${MY_NAME}\rules
styles_path = d:\${MY_NAME}\styles
```

When **rules_path** is evaluated, **rodney** is substituted for **${MY_NAME}**, so that the value of **rules_path** becomes "**d:\rodney\rules**". As before, this substitution occurs when the variable is used by HoTMetaL, not at start-up.

The '$' symbol is used as a special character in configuration variables, so if you want to put a '$' in the value of an variable, you have to represent it with '$$'. For example:

```
print_middle_header = $$64 question
```

assigns **print_middle_header** the value "**$64 question**".

Appending and prepending to a variable

If a configuration variable has already been assigned a value, you may wish to **append** or **prepend** some characters to it. For example:

```
rules_path=c:\hotmetal\rules;
rules_path += d:\jennifer\rules;

styles_path=c:\hotmetal\styles;
styles_path =+ d:\jennifer\styles;
```

In the first example, the variable **rules_path** is first given the value "**c:\hotmetal\rules;**". The '+=' in the expression

```
rules_path += d:\jennifer\rules;
```

causes "**d:\jennifer\rules;**" to be *prepended* to the current value of **rules_path**. The value of **rules_path** becomes "**d:\jennifer\rules;c:\hotmetal\rules;**".

In the second example, the variable **styles_path** is first given the value "**c:\hotmetal\styles;**". The '=+' in the expression

```
styles_path =+ d:\jennifer\styles;
```

causes "**d:\jennifer\styles;**" to be *appended* to the current value of **.styles_path**. The value of **styles_path** becomes "**c:\hotmetal\styles;d:\jennifer\styles;**".

(In these examples, the semi-colon, ';', between the file names, is *not* inserted automatically by HoTMetaL.)

If a variable does not already have a value, then assigning it a value using '=+' or '+=' has the same effect as just assigning a value using '='. For example, if **rules_path** does not currently have a value, then

```
rules_path+=c:\rules;
```

has the same effect as

```
rules_path=c:\rules;
```

Control variables

These variables control various aspects of HoTMetaL's behavior: export options, find and replace options, etc. Many of these variables take values of **YES** or **NO**; please note that **YES**, **true**, and **1** (one) are synonymous here, as are **NO**, **false**, and **0** (zero).

Save options

The following variables allow you to choose default **save options** for the **Save** and **Save As...** command. .

export_doc_type_dec

By default, HoTMetaL will save the **document type declaration** (**DOCTYPE**) when it saves a file. If this variable is set to **NO**, then HoTMetaL will *not* include the **DOCTYPE** with an exported file. If it is set to **YES**, or if it is not set at all, then the DOCTYPE will be saved with the file.

export_add_line_breaks

By default, HoTMetaL will not impose any limit on the length of a line in an saved file, i.e., it will not add any explicit line breaks. If this variable is set to **YES**, HoTMetaL will add line breaks after the number of characters specified with the **export_max_line_len** variable (see below). If **export_add_line_breaks** is set to **NO** or omitted, no line breaks will be added.

export_max_line_len

If **export_add_line_breaks** is set to **YES**, lines will be broken after a certain number of characters. You can set this number with the variable **export_max_line_len** or in the dialog box, e.g.,

```
export_max_line_len=60
```

If this variable is not set, the default is 80 characters.

export_convert_special_chars

If this variable is set to **YES**, then HoTMetaL will convert any special characters in your document to SGML **character references**. (Special characters are those outside the ASCII range 0-127). This form is recognizable by other applications and therefore makes your file more portable. If the variable is omitted or set to **NO**, then special characters are not converted.

export_eol

This variable lets you choose the end-of-line marker that will be generated in your saved file. There are three choices: UNIX, which causes the end-of-line marker to be a line feed, MSDOS, which causes it to be a carriage return followed by a line feed, and MAC which sets the marker to be a carriage return. The default value for this variable is **MSDOS**.

Find

The next group of variables allow you to control the behavior of the commands in HoTMetaL's **Find** menu. You may override all these settings from the dialog box that accompanies the **Find and Replace...** command.

find_whole_words

By default, the **Whole Words** option is turned off in the **Find & Replace** dialog box. If this variable is set to **YES**, this option is turned on; if it is set to **NO** or undefined, **Whole Words** is turned off in the dialog box.

find_case_sensitive

By default, the **Case Sensitive** option is turned off in the **Find & Replace** dialog box. If this variable is set to **YES** this option is turned on; if it is set to **NO**, or undefined, **Case Sensitive** is turned off in the dialog box.

find_backward

By default, the **Backwards Search** option is turned off in the **Find & Replace** dialog box. If this variable is set to **YES**, this option is turned on; if it is set to **NO**, or undefined, **Backwards Search** is turned off in the dialog box.

find_wrap

By default, the **Wrap** option, which causes searches to encompass the entire file, starting at the current position, is turned on in the **Find & Replace** dialog box. If this variable is set to **NO**, **Wrap** will be turned off. If the variable is set to **YES** or not defined, wrapping will be turned on.

find_patterns

By default, the **Find Patterns** option is turned on in the **Find & Replace** dialog box. If this variable is set to **NO**, **Find Patterns** is turned off. If it is set to **YES**, or not defined, the **Find Patterns** option is turned on.

Markup options

These variables govern aspects of the markup process.

include_required_elements

This variable controls whether the **Include Required Elements** option is turned on for the **Insert Element...** command. If this variable is set to **NO**, the option will be turned off. If the variable is set to **YES** or undefined, the option will be on. For more information, see the section on **Insert Element...** in the **Markup** chapter.

prompt_for_attrs

This variable controls whether the **Edit Attributes and Links** dialog box will be displayed each time an element with attributes is inserted in the document. If it is set to **NO**, or undefined, then users will be prompted with this dialog only if the element being inserted has **REQUIRED** attributes. If the variable is set to **YES**, then the user will be prompted every time an element with attributes is inserted.

Display variables

The variables in this section pertain to HoTMetaL's screen display: invisible characters, fonts for icons, and colors.

tag_font_name

This variable lets you choose the font used to display the element names in the tag icons in an HoTMetaL document. The list of available fonts is dependent on your system and on the font source file that you are using. The best way to find out which fonts you can use is to invoke HoTMetaL's **Character...** command and click on the arrow next to the drop-down list box labeled **Font Family**. The menu which appears contains the names of the available fonts. The value assigned to the **tag_font_name** variable should be the font name, not surrounded by quotes, exactly as it appears in the **Font Family** menu. The default font is **System**.

tag_font_size

This variable lets you choose the font size used to display the element names in the tag icons in an HoTMetaL document. The list of available font sizes is dependent on your system and on the font source file that you are using. The best way to find out which font sizes you can use is to invoke HoTMetaL's **Character...** command and click on the arrow next to the drop-down list boxlabeled **Font Size**. The menu which appears contains the available font sizes. The value assigned to the **tag_font_size** variable should be the font size in points, not surrounded by quotes. The default is 12 points.

Other options

undo_limit

This variable sets the maximum number of commands that can be undone with the **Undo** command. By default, this value is 10. The maximum value is 65535. The minimum value is one; if you set it to a value less than one, it will be set to one anyway.

publish_change_from

This variable specifies the text that appears in the **Change From** text entry box in the **Publish...** command's dialog box. The default value is file:///

publish_change_to

This variable specifies the text that appears in the **Change To** text entry box in the **Publish...** command's dialog box. The default value is "http://".

view_gif

This variable specifies a program which the **Show Image** command will invoke to display a file whose name ends with the **.gif** file extension (normally this file would be expected to be in GIF format).

You can add variables of your choice of the form **view_***extension*, where *extension* is the file extension of the file you want to display: for example, you could have **view_tiff**, **view_jpg**, etc., variables.

html_browser

This variable names an HTML browser which can be invoked by HoTMetaL's **Preview** command. The default value is **c:\mosaic\mosaic.exe**.

Location variables

The variables described below give the locations of directories or files that are used by HoTMetaL, and specify default file extensions.

Paths and directories

Paths are lists of directories which HoTMetaL searches to find files it needs to read, or uses to store files. The value of a variable that describes a path consists of a number of directory names, or paths, separated by semi-colons '**;**'. As well as giving specific directory names, it is possible to specify HoTMetaL's *working directory* by putting a period, '**.**', in the path.

Note: The working directory is set with the 'Properties...' command in the Microsoft Windows 'File' menu.

When giving a directory name (other than the working directory), you need to give the full DOS path. This may be fully or partly represented by a reference to another configuration or environment variable, as explained earlier in this chapter.

The following example shows how to set a path variable, in this case, **ascii_styles_path**:

```
ascii_styles_path=${SQDIR}\styles:.:c:\donald\styles
```

This setting is interpreted as follows:

- The expression '${SQDIR}' is replaced by the name of the HoTMetaL directory, so the expression '${SQDIR}\styles' will cause a directory such as **c:\hotmetal\styles** to go into the ASCII styles path.
- The current working directory when the program was invoked, signified by ".", is included in the path.
- Lastly, the directory **c:\donald\styles** is specified explicitly.

The default value for all path variables, with the exception of **save_path**, is ".".

ascii_styles_path

This variable gives a default directory for storing and loading ASCII styles files. When you invoke the **Load Styles...** command in the **View** menu, the file selection dialog box which appears has a list labeled **Directories**. This list displays, by default, the directory which is named first with the **ascii_styles_path** variable.

export_path

This variable names the default directory for saving SGML files. When you invoke the **Export...** command, the default directory that appears in the **Directories** list box is the first directory that is listed on the **export_path** variable.

import_path

This variable gives a default directory for opening files. In the file selection dialog box that appears when you invoke the **Open...** command, the **Directories** list box displays, by default, the directory that is listed first with the **import_path** variable.

rules_path

This variable describes the rules path, a list of directories which HoTMetaL will search through when it needs to find a compiled rules file: for example, when you're opening an HTML file. The order in which the directories appear is significant: when looking for a rules file, HoTMetaL searches the directories in the order in which they appear on the **rules_path** variable. It stops looking as soon as it finds a file with the right name.

styles_path

This variable describes the styles path, a list of directories where styles files are located. These are files used by HoTMetaL to describe the formatting for a document when it is displayed on the screen. HoTMetaL will create a styles file for a rules file the first time that rules file is used. It places the styles file in the first directory listed by the **styles_path** variable. On subsequent uses of the rules file, HoTMetaL will look for the styles file in the directories named by the variable. If the rules file has been changed since the styles file was created, HoTMetaL will ask you if you want to create a new styles file.

The format of this variable is the same as for **ascii_styles_path**, above.

templates_path

This variable gives the directory for storing files that can be used as document templates with the **Open Template...** command.

Files

These variables give the names of specific files that HoTMetaL uses.

Except as noted, when giving a file name you need to give the full DOS path. This may be fully or partly represented by a reference to another configuration or environment variable, as explained above.

rgb_txt

This variable names the color map file, a file which associates color names with the red-green-blue values required to tell HoTMetaL how to produce the colors. The value of this variable should be a file name. If an absolute path is prepended to the file name, then that file is used; if the file name has a relative path, or no path, prepended to it, then HoTMetaL looks for a file relative to the SQDIR directory. The default value is ${SQDIR}**rgb.txt**.

File extensions

These configuration variables determine the default file extensions that appear in the file selection dialog boxes for different kinds of files. The file extensions appear in the **Filename** text box. The file extension consists of a dot followed by a sequence of characters (usually three). When you set a file extension, you must include the dot in the corresponding variable's value. E.g.,

```
rules_ext=.rls
```

ascii_styles_ext

This variable sets the default extension for ASCII styles files. The default extension is **.asf**. The text box labeled **File Name** in the **Load Styles** dialog box will, by default, contain the ***.asf** pattern, causing the **Files** list in the dialog box to contain all files having that extension in the currently selected directory.

export_ext

This variable sets the default extension for SGML files being saved with the **Save** and **Save As...** commands. The default extension is **.htm**. The text box labeled **File Name** in the **Save As** dialog box will, by default, contain the ***.htm** pattern, causing the **Files** list in the dialog box to contain all files having that extension in the currently selected directory.

import_ext

This variable sets the default extension for SGML files being opened. The default extension is **.htm**. The text box labeled **File Name** in the **Open** dialog box will, by default, contain the ***.htm** pattern, causing the **Files** list in the dialog box to contain all files having that extension in the currently selected directory.

styles_ext

This variable sets the default extension for binary styles files. When you create or open an HoTMetaL file, HoTMetaL will look in the styles path for a styles file which has the same name as the rules file's compiled-in system identifier, but with the file extension replaced by the styles extension. The default styles extension is **.stl**.

Tracing configuration variables

HoTMetaL allows you to trace exactly how the configuration settings are used. When tracing is turned on, you will be presented with a warning box when HoTMetaL reads the configuration files, or accesses any of the variables. Tracing is controlled by the SQTRACE environment variable. The possible values are ON (which is the same as **true** and **1**), OFF (which is the same as **false** and **0**) and FULL. The variable must be set before starting up HoTMetaL.

If SQTRACE has the value ON, you are notified whenever any of the following things happen:

- HoTMetaL reads the configuration files.
- the final value (i.e., after all configuration files have been read) of a variable is set. Furthermore, you will be told which of the files the value comes from.
- HoTMetaL looks for the value of a variable but does not find it in any configuration file.
- a duplicated variable is detected. If a variable is set in more than one configuration file, or more than once in the same file, you will be notified, and given the previous and new values and which files these values came from.
- an absolute file name or path name is encountered as the value of a variable.

If SQTRACE has the value OFF, or is not set, tracing will not be invoked.

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W