OmniHelp Design Report

OmniHelp is an open-source, cross-platform system for displaying Help information in a manner similar to that used by WinHelp and HTML Help systems, with support for context-sensitive Help and modular Help projects. This document describes OmniHelp Version 0.8 design and formats.

OmniHelp is licensed under the LGPL, which permits its use in commercial products as long as OmniHelp source code, including all modifications, is made available to users. All OmniHelp development work to date has been contributed by Jeremy H. Griffith, Carolyn Stallard, and other staff of Omni Systems, Inc., with the support of some Mif2Go users. Hopefully other vendors will choose to participate in OmniHelp development work. The OmniHelp project is officially hosted on SourceForge, from which the current version of this paper, and all required files, are available for download (as well as on omsys.com).

The OmniHelp viewer consists of a set of HTML and JavaScript files that present the Help content in a tri-pane format, using any browser that meets the following criteria:

- complies with minimum Web standards
- supports framesets
- supports basic CSS1.

Supported browsers include NN4+ and IE4+. Help content is in HTML and JavaScript (data) files.

So far, the only Help Authoring Tool (HAT) that supports OmniHelp is Omni Systems Mif2Go, which produces content and data files from FrameMaker documents. (Mif2Go also supports many other Help formats, as well as XML, XHTML, and Word RTF, from FrameMaker 5.5, 6, and 7.x on any Windows platform.)

OmniHelp display

OmniHelp displays Help in a tri-pane window, using frames. The top navigation pane presents the usual buttons: **Start**, **Prev**, **Next**, **Back**, **Fwd**, and **Hide**/**Show**. These buttons do what you would expect. **Hide**/**Show** controls the display of the left navigation pane.

Four navigation features list topic titles as links

The left navigation pane displays tabbed **Contents**, **Index**, **Search**, and **Related**:

Contents Displays the TOC, linked to HTML topic files, with expanding topic groups. For Nav 4 only, it shows the full TOC or one of two shorter versions that show only entries around the currently displayed topic.

Index Presents an array of letters; displays expanding index entries for the selected letter, each entry followed by titles of the referenced topics, as links.

Search Permits entry of a single search term, or a series of terms separated by Boolean operators AND, OR, NOT. The search result is a list of topic titles, as links. **Search** can also list all pre-indexed search terms, and show the number of topics each term references.

Related Based on ALinks, or Subject markers, this feature displays as links the titles of topics you might expect to see when you click a “Related Topics” button in other systems.

OmniHelp integrates these four features to provide a common, consistent way of using each. In all four, the navigation pane contains a list of topic titles that are linked to the

topics they identify. There are no buttons to look for, and no additional dialogs to work through.

Use current or new browser window

You can view OmniHelp two ways:

- In the current browser window, whatever its size may be.
- In a separate window of settable size, with most of the browser "chrome" removed, leaving a simple menu and status-bar frame. At present, this works in Internet Explorer and Opera, but not in Firefox or other Mozilla-derived browsers.

Browser Detection and CSS

CSS controls navigation displays

On loading, OmniHelp detects the browser in use, and selects the appropriate CSS files for displaying navigation labels and data, and topic content. Navigation displays are controlled by the CSS settings, as specified in `ohctrl.css`, the default stylesheet, and in the browser-specific variants for Netscape Navigator 4 (`ohctn4.css`), Netscape Navigator 6+ (`ohctn6.css`), and Internet Explorer 4+ (`ohctie.css`). The author (or user) is free to edit this CSS, and to produce more variants.

The CSS for displaying topic content is best supplied by the author or user, because it depends on the classes used in topic files. Again, the built-in browser detection offers the same variants listed above for NS4, NS6, and IE.

Popups and Secondary Windows

OmniHelp fully supports the use of secondary windows and full-HTML popups which can contain links of their own. For example, a popup may be handy for a **Note:**. A secondary window may be useful for a large graphic, such as the enlarged form of the OmniHelp Logo, or for a stepwise procedure.

The coding for such display is simple. The link (`<a href=...>`) contains a `target` attribute with the name of the desired window, and an `OnClick` attribute that calls a Javascript function: `sec(this, 200, 100); return false` where the `200` and `100` are optional, the desired width and height for the window. Another parameter may be added to specify JS opening properties, such as `menubar,titlebar`. Mif2Go automates this process, using FrameMaker character formats applied to the link hotspots to identify the target window, with window properties defined just once in mif2htm.ini.

Popup windows behave like other secondary windows, except that you get a fresh popup every time you target the popup window. With the other window types, if the targetted window already exists, the new content goes into the existing window replacing the old. Popups also have less "chrome" by default; their JS opening properties are minimalist: "`dependent,scrollbars,resizable`". Secondary windows have that plus quite a bit more: "`title,titlebar,status,menubar,toolbar`".

If you have a plain link in a popup or secondary window, the new content replaces the old in the same window. If you want to put the new content back into the topic window in the frameset, simply use `target="main"` in the link; no `OnClick` is needed in that case.

Different browsers do handle these windows differently. With Netscape, the popups stay on top, and when the main frameset is closed, so are the popups and secondary windows. With IE, however, the windows do not stay on top, and they remain after the frameset is closed. Neither browser gives the WinHelp popup effect where clicking anywhere on the popup or main window closes the popup; you have to close the windows explicitly by clicking the top right X.

Note: This example of popup usage includes the OmniHelp logo:  . It can also include links, such as this one to the Search topic put back in the main window.

OmniHelp Logo

OmniHelp generation

For topic (text) information, OmniHelp uses HTML 4.01 files conforming to current W3C Recommendations. (It can also accept XHTML, but this can cause problems with CSS in some browsers.) OmniHelp requires one addition to the `<head>` of such files:

```
<script language="JavaScript" type="text/javascript" src="ohmain.js">
</script>
```

Synchronized contents

This script tag links in OmniHelp CSS support, and keeps the Contents synchronized when you click links in the topic pane. OmniHelp can load HTML files from any source in the topic pane, via links from an OmniHelp topic. However, if a file is not listed by name in the Contents, the Contents display remains where it was when the previous file loaded; **Prev**, **Next**, **Back**, and **Fwd** buttons have no effect; and the Related topics feature is not available. The Index and Search panes are usable normally, as are the **Start** and **Hide**/**Show** buttons.

Navigation data files

Project-specific navigation information is contained in four data files; a fifth data file is used for context-sensitive help. The data files are named for the project, with the following suffixes:

`.ohc`	Table of contents
`.ohk`	Topic index
`.ohs`	Search index
`.ohl`	Related topics
`.oha`	CSH identifiers

All data files use a similar format: JavaScript two-dimensional arrays that require no parsing by the OmniHelp code; the arrays are used directly. For example, here is part of the `.ohc` data for a sample project:

```
[2,"9 Using markers for HTML markup","htmlmark.htm#Xae1015982"],
[3,"9.1 Working with FrameMaker markers","ae999397.htm#Xae1015686"],
[3,"9.2 Adding custom marker types","ae999029.htm#Xae999029"],
```

And here is part of the `.ohk` data for the same project:

```
[1,"title, HTML attribute"],
[2,"for images, assigning via marker"],
[3,"ae999029.htm#Xae1003420",3],
```

The last index line (last line in the `.ohk` data sample) refers back to the topic in the last contents line (last line in the `.ohc` data sample), but to a different point within that topic, via the final “3” (the index of the contents item).

Start-up and project files

OmniHelp uses two additional files with project-specific content:

`_project``.htm`	Project file: contains the title displayed for the project
`.ohx`	Settings file: contains all customizable JavaScript variables

Mif2Go generates these two project-specific files, as well as the five `.oh*` data files. A Help author can optionally use a template file, `ohtpl.ini`, to customize the *`_project`* and `.ohx` files. The OmniHelp distribution includes sample files `_OmniHelp.htm` and `OmniHelp.ohx` for manual editing, if Mif2Go is not used to generate OmniHelp.

Control files

The rest of the OmniHelp files are fixed in content, although they can of course be modified by anyone willing to invest the time to understand the JavaScript they contain. For the usual HTML output, these files are:

`ohstart.js`	Start-up script, included by reference in *`_project`*`.htm`
`ohmain.js`	CSS-setting script, included by reference in all topic files
`ohframe.js`	Framesetting script, referenced from `ohframe.htm`
`ohtop.htm` `ohtop.js`	Top navigation-pane loader and script
`ohctrl.htm`	Main control file, which loads into an invisible frame; directs most OmniHelp operations with six script files:

`ohctrl.js`	Control
`ohtoc.js`	Contents
`ohidx.js`	Index
`ohfts.js`	Search
`ohrel.js`	Related topics
`ohmerge.js`	Merge

`ohmerged.htm` `ohmerged.js`	Run-time project merging loader and script
`ohlang*.js`	Three language-dependent files that contain all visible UI text
`ohctrl.css`	CSS used for navigation control, with variants named:

`ohctie.css`	Internet Explorer 4 and up
`ohctn4.css`	Netscape 4.x
`ohctn6.css`	Netscape 6 and up

`ohdr.css`	Sample CSS used for main topic panel, with variants as above.

`ohnav.htm`
`ohnavctrl.htm`
`ohmain.htm`
	Placeholder HTML used during frameset construction to prevent browser difficulties (particularly with IE).

`ohtc1.gif`
`...`
`ohtc13.gif`
	Icons used for expanding Contents and Index, not used for Netscape 4.x

For the alternative XHTML output, the following `ox*.htm` files replace `oh*.htm` files: `oxctrl`, `oxframe`, `oxmain`, `oxmerged`, `oxnav`, `oxnavctrl`, and `oxtop`. In addition, `oxlang.js` replaces `ohlang.js`.

OmniHelp design

The OmniHelp project file, *`_project`*`.htm`, serves as the initial loader of the frameset. The `ohstart.js` script checks the `.ohx` to determine whether a new window is wanted (`newWindow`), and if so opens one with `titlebar,title,status,resizable` properties, plus whatever is specified for `frameOptions` (comma-separated with no spaces), sized according to `frameHigh` and `frameWide`. The list of potential options is very long, and is browser dependent; refer to JavaScript docs for the possibilities.

If a new window is specified, but cannot be opened (perhaps because popups are disabled in the current browser), the existing window is used instead. Otherwise, if `closeWindow` is also specified, an attempt is made to close the original window; the success of this is very browser-dependent. NS 7.1, for example, refuses to close the original window because it was not opened by the same script that is trying to close it.

The project file can have any name you please; not just *`_project`*`.htm`, which was chosen so that the name would appear at the top of an alphabetized directory. In some cases, the name `index.html` may be best, so that an OmniHelp doc (perhaps on a Web site) can be referenced solely by the name of the containing directory.

Frameset

OmniHelp frameset design complies with the W3C HTML frameset DTD, except for one attribute in the main frameset: `border="0"`. Unfortunately, there is no other way to remove the frame borders. If you require strict compliance with the W3C DTD, and do not mind borders, you can remove this attribute by setting `frameBorder=true` in the settings file `.ohx`.

Three alternative framesets can be constructed by `ohframe.js`:

- The normal tri-pane with the top pane all the way across, the navigation-control and navigation panes below it to the left, and the topic pane below it to the right.
- An alternative tri-pane in which the left frameset (navigation-control and navigation) extends all the way to the top of the window, and the top navigation and topic panes are next to it on the right. This is used instead of the normal tri-pane if `topFirst=false` in the `.ohx`.
- A two-pane set in which the navigation-control and navigation panes are gone, and the top navigation and main panes are the full width of the window, which does not shrink horizontally. This set is loaded if `showNavLeft=false` in `.ohx`, and replaces the other tri-panes if you click the top navigation **Hide** button during operation. In either case, the **Show** button restores the chosen tri-pane version.

In all cases, the frameset includes two hidden frames above the top navigation pane: a control frame and a merge frame. The control frame contains all the JavaScript code and `.oh*` data needed for OmniHelp operations. The merge frame is used during subproject loading.

The height of the top navigation pane, the width of the left panes, and the height of the navigation-control pane are all settable in pixels by variables in `.ohx`:

- For the top navigation pane: `topHigh=50`
- For the two left panes, navigation-control and navigation: `leftWide=220`
- For the navigation-control pane: `midHigh=90`

It is not a good idea to reduce any of these sizes. The top navigation pane must accommodate the buttons; and the navigation-control pane must fit the table used for navigation tabs. The default sizes are the very minimum that work in all browsers. However, for the navigation-control pane, the set size is overridden to allow room for any of the optional components used in each different navigation pane.

Top navigation pane

The top navigation pane is created by project-specific file `ohtop.htm`. The modifiable information in this file is mostly “branding” information, which might be more company-specific or product-specific than project-specific. This HTML code is specified by two settings in `.ohx`, one for the top left and the other for the top right. Sample file

OmniHelp.ohx contains “OmniHelp Example” for the left, and the W3C HTML compliance icon for the right. Both occupy individual table cells, and can be replaced by more suitable content by Help authors. When a Help author uses Mif2Go to generate OmniHelp, the changes would be made in configuration file mif2htm.ini, and passed on to the .ohx file.

The rest of the top navigation pane contains sets of buttons. The sample files include all buttons currently supported by OmniHelp code; all are removable. Settings file .ohx, and Mif2Go template file ohtpl.ini, contain variables useBackForward and useHideShow, to determine whether **Back**/**Fwd** and **Hide**/**Show** buttons are displayed.

Start button

Start selects the normal starting topic to display in the topic pane. If a value for mainName is specified in .ohx, that topic file is displayed. Otherwise the file referenced from the first Contents entry is displayed. Any CSH selection shown at initial load is ignored. The navigation-control and navigation panes are set to show the current Contents.

Prev and Next buttons

Prev and **Next** move to the topic immediately before or after the current topic in the full Contents sequence, just as though the user clicked the corresponding Contents entry. If the Contents control is set to other than **Full**, the topic selected might not be one that appears in the visible Contents. If **Prev** is clicked at the starting topic, or **Next** at the ending topic, nothing happens.

Back and Fwd buttons

Back and **Fwd** work much like a browser’s Back and Forward, except that they apply only to the files displayed in the topic pane; the browser’s Back button might well move out of OmniHelp altogether, returning to the previous URL. (If OmniHelp is loaded in its own window, usually there are no browser buttons). If **Back** is clicked for the first OmniHelp file displayed, or **Fwd** is clicked without previously clicking **Back**, nothing happens.

Hide/Show button

Hide/**Show** allows the user to remove the navigation-control and navigation panes entirely, usually when screen space is limited. The topic pane expands to fill the space formerly occupied by the left navigation panes. Unfortunately, in Netscape Navigator 4, doing so requires a full reload of OmniHelp, which can be slow; in later versions, and in Internet Explorer, the change is instant. When the navigation pane is displayed, the button name is **Hide**; when the navigation pane is hidden, the button name changes to **Show**.

Left navigation pane

The navigation-control and navigation panes work together. Normally they provide four navigation features: Contents, Index, Search, and Related topics (ALinks). However, if some of these are inappropriate to a particular OmniHelp project, the unwanted features can be removed with .ohx variables useNavToc, useNavIdx, useNavFts, and useNavRel. All four variables are normally set to true; if one is set to false, the corresponding navigation feature is disabled, and the name of the feature does not appear as a tab in the top row of the navigation-control table. The cells for the remaining features are widened to compensate. OmniHelp does not load the data file for the deselected feature, ohctrl.htm does not load the corresponding JavaScript code file, and the

omitted feature is ignored when subprojects are merged. If Mif2Go is used to generate OmniHelp, these settings are made in configuration file `mif2htm.ini`.

Contents

The Contents list automatically stays in sync with the topic file loaded into the main pane, unless that topic file is not listed in the Contents (that is, is not a part of the current project). If subprojects are defined, OmniHelp checks to see if the selected file belongs to a subproject, and if so merges the subproject into the main project. Otherwise, the Contents continues to display the context of the last-displayed topic file.

As with WinHelp and HTML Help, the OmniHelp Contents list expands and contracts if in the `.ohx` file `tocExpand` is set. In addition, buttons at the top of the panel permit full expansion or contraction of the Contents in one operation. The `tocGroupsOpen .ohx` setting determines whether groups are all open or closed at the start.

Closing the groups containing the current topic

It is problematic to close groups that contain the currently-displayed topic, as that would break the synchronization between topics and Contents. So OmniHelp ignores any such request; even if the **Close All** button is pressed, all of the ancestors of the current topic remain open. This may be startling to those used to the HTML Help way of dealing with this situation, which is to close the group and change the current topic to the one at the top of the group just closed. One can accomplish this in OmniHelp simply by clicking on the topic itself, to select it, before clicking on the minus box to close its group. That way, the change of topic is an explicit choice, rather than an inconsistent side-effect of closing.

Contents list for Nav 4 can be Full, Medium, or Short

If the expanding Contents is not selected, or if Nav 4 is detected; the Contents is fully expanded instead, so that every entry is an immediately clickable link. Because a fully expanded Contents can take several seconds to display for a large project, the user can choose to view a subset of Contents entries, using radio buttons in the second row of the navigation-control table. These buttons offer a choice of **Short**, **Medium**, or **Full** display; the start-up default is **Medium**; however, any user-specified setting is retained, using persistent cookies.

Short Contents list

The short display is designed to fit in the navigation pane without requiring scrolling under almost all conditions. The short display includes the current topic; above the current topic, its direct ancestors up to the top level of the document, much like the list shown in trails (“bread crumbs”); below the current topic, one level only of children of the current topic. The short display ends with the next topic at the same level as the current topic, and the next after that at a higher level; this allows easy movement forward through the system to later major heads. This configuration displays very quickly.

Medium Contents list

The medium display gives a bit more information, although usually a lot less than the full display. The medium display includes all items in the short display, plus above the current topic the siblings of the direct ancestors (aunts and uncles, but not cousins). Below the current topic the medium display includes all levels of descendants (grandchildren), and all following siblings (not just the next sibling).

Index

For the Index, the navigation-control table shows an array of letters, possibly preceded by items **Sym** and **Num**. The array shows only those letters for which actual index entries exist. When the Index tab is first displayed, the first item in the array shows as selected; however, any subsequent user selection is stored, using persistent cookies. The navigation pane displays the index entries for the selected letter, in an expanding form if in the `.ohx` file `idxExpand` is set. While the number of levels is not limited by OmniHelp, there is a practical display limit of about nine levels, and after six very little of the text is shown.

However, since hovering over any bit of an entry displays a tooltip with the full entry title, this is not a major problem. A horizontal scrollbar is also provided.

As with the Contents, the Index presents all destinations as direct links, in each case showing the full title of the topic referenced. There are no additional dialogs to navigate when an index entry links to multiple topics, and a user can tell instantly if the link is to a topic already viewed. We consider this an improvement over older designs.

Search

The Search facility pre-indexes topic text to achieve good performance. Search terms can consist of a single word or a phrase; the phrase search is implemented by successively ANDing the search terms. The search results can be refined by using Boolean operations with additional terms.

The user enters the search term in a box on the second row of the navigation-control table; if the term begins with a forward slash, `/`, it is processed as a JavaScript regular expression (regex). The user selects a Boolean operator by clicking a radio button on the third row: **New** for a fresh search, **AND**, **OR**, or **NOT**. The user clicks **Find** to the right of the text box to populate the navigation pane with a heading that lists the terms used in the search, followed by the search results, which consist of topic titles linked directly to their topics.

When a found topic link is clicked, the topic displays in the right pane with the search terms found highlighted; by default, the highlighting is a yellow background color, but the CSS style to be used for it can be set in the `.ohx` file with `ftsHighlightStyle`.

To find out what search terms are useful, the user can click **List** to the left of the text-entry box, to populate the navigation pane with a list of all available search terms. Each term is followed by a number in parentheses that shows how many references to the term exist in the topic files. The terms listed are themselves links; clicking a term causes that term to be entered into the text box.

The last text entry, and the last Boolean setting, are both retained in persistent cookies until changed by the user. If there are no previous results to work with, the Boolean setting is automatically changed to **New** when the user clicks **Find**.

Related topics

Related topics in OmniHelp are the same as those implemented in WinHelp and HTML Help using ALinks to associate items having the same subject matter. Related topics also provide a way to manage links to subprojects—when the target topics might or might not be present—without risking a broken link if the topic is indeed missing. In OmniHelp, ALinks are treated as first-class navigational aids, just like Contents, Index, and Search. There is no need to add “Related Topics” links and buttons, and navigate through dialogs, to use this feature in OmniHelp. When the **Related** navigation tab is selected, the navigation pane shows as links the titles of all topics that share any of the ALink references in the current topic. This display stays in sync with the current topic. The upper part of the pane shows the actual ALink terms. However, since ALink terms are not shown in other systems, and might even prove embarrassing, their display may be turned off in OmniHelp by an `.ohx` setting, `relShowSubjects`.

OmniHelp also supports ALink jumps and KLink jumps, like those used in WinHelp and in HTML Help. (JavaHelp does not support ALinks or KLinks, and Oracle Help for Java supports ALink jumps only.) With Mif2Go, the link is specified with an `alink:` or `klink:` protocol; in the OmniHelp, this protocol becomes a call to `javascript:alink(...)` or `klink(...)` where the parameter is the standard set of one or more semicolon-delimited identifiers. When a user clicks an ALink or KLink jump

in a topic, with the **Related** tab selected in the navigation-control pane, the navigation pane shows a list of links to the relevant topics, sorted in Contents order.

Context-sensitive help

OmniHelp's ability to support context-sensitive Help (CSH) is somewhat limited by the fact that JavaScript cannot offer an API interface to other programs. Nor is there any way for another program to pass information to JavaScript after OmniHelp is loaded. Therefore CSH is restricted to what can be done by invoking the browser with a file name. You can invoke OmniHelp with either a file name or a Context ID:

- File name: `_project.htm#myfile.htm`
  When the OmniHelp frameset comes up, the named file is displayed instead of the usual starting topic.
- Context ID:`_project.htm#contextID`
  The file with which the ID is associated is located in `project.oha` and loaded first.

In the topic file, the Context ID is a normal anchor: `<a name="contextID"></a>`. The entry in `project.oha` looks like this:

```
["idh_chooseproject","02x998911.htm"],
```

where the first element is the ID, and the second is the file to display.

In Mif2Go, the entry is created from any FrameMaker **newlink** marker, possibly qualified by a required prefix (such as `idh_`) specified by the user in a `mif2htm.ini` configuration setting.

When CSH is invoked by calling a specific browser, by name, it works as expected. However, when invoked via `cmd start` for the filename, the hash value can prevent startup. In that case, it's necessary to call it via a "redirect file", with no hash, which contains a meta tag identifying the real URL wanted:

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
<html lang="en">
<head><title>List of Tables</title>
<meta http-equiv="refresh"
 content="1;url=file:///G:/Omnisys/UG/OH/Done/_usergd.htm#tablist">
</head>
<body></body></html>
```

Modular help

OmniHelp supports modular Help in a way very similar to that used by HTML Help and WinHelp. The main Help project must include a setting in `ohproj.js` that describes each subproject; for example:

```
var mergeProjects = [
["oh8",0,0,0,["htmlref","ag*"]],
["oh11",0,0,51,["htmllink","ah*"]]]
```

where for each subproject, the items included are:

- The subproject name, as in `"oh8"`
- A zero to indicate unmerged status
- A zero for the size of the subproject (filled in later)
- The location in the main TOC sequence of the contents entry for the subproject, the point at which it is inserted into the main project

- A list of files contained in the subproject, used when a file that is part of the main project links to a file outside that project to determine if the file about to be loaded is known to be part of the subproject

The main project must also have an entry in `.ohc` showing the subproject's position in the Contents:

```
[2,"8 HTML Reference","*oh8"],
```

where the three elements are:

- contents level number
- title to be displayed in the Contents
- name of the subproject; the asterisk identifies it as a subproject entry.

A subproject is loaded into the main project in three circumstances:

- when a user clicks a link to a file that is not part of the current project, and the name of that file appears in the file list of a subproject in the `mergeProjects` setting
- when a user clicks a Contents entry that specifies a subproject name
- at initial load, for all subprojects, if the setting `mergeFirst=true` is in the `.ohx`.

Subprojects are loaded one at a time, under control of JavaScript code in `ohmerge.js`, which is part of the `ohctrl.htm` set. The merge process does the following:

- integrates the full contents of the subproject's data files with the main project, seamlessly, as if they had always been one project
- merges the contents of the subproject's `mergeProjects` setting, so that any subprojects included in the previous subproject are now included in the main project.

Subproject merging can be nested to any degree. A newly merged subproject is checked to make sure it does not include as a subproject the original main project, or any other already loaded subproject; if either is the case, the duplicate entry is quietly removed from both `mergeProjects` and Contents.

When `mergeFirst` is set to merge all projects at load time, the process is carried out in a depth-first recursive fashion until all subprojects have been loaded. Any that cannot be found are marked as unloadable and are not attempted again; the entry is removed from the Contents. This permits including entries for subprojects that may not exist yet.

All subproject files must be in the same directory as the main project files. Files may be shared between two or more projects; only one copy of a shared file is needed. This sharing can include the `oh*.htm` and `oh*.js` files. Because the project-specific files for two projects always have different names, there are no name clashes with this system.

In Mif2Go, the location for a subproject Contents entry is marked with a FrameMaker **HelpMerge** marker that contains the subproject name. The text to display for the link, and other subproject details, are specified by the user as settings in the `mif2htm.ini` configuration file. The Mif2Go `[Automation]` facility can place the needed project files into a shared "wrap" directory, and archive them in a single `.zip`.

Data file formats

All data files used by OmniHelp are in standard JavaScript array format. Each begins with a line naming the array and providing the opening bracket, such as `var relItems = [`, and ends with a closing bracket after the bracket that ends the last entry. Each entry is on a line of its own, enclosed in brackets `[]`, and all *except the last* end with a comma.

Text items in an array are enclosed in double quotes. A double quote or backslash within text must be escaped with a backslash. If the text is destined for display, it must be

properly encoded for HTML, using entities and numeric character references as needed. If, however, the text is a link, it should *not* be URL-encoded, because OmniHelp takes care of that encoding.

We strongly recommend that file names consist of alphanumeric characters only, with one dot (period) for the file extension (which should be standard), with forward slashes (rather than backslashes) for paths, and with no spaces whatsoever. Otherwise cross-platform usage can be impaired, if not rendered impossible.

Contents: .ohc

Each Contents entry consists of three parts. The first item is always the contents level, a single digit from 1 to 9. (Zero is not used, and is reserved.) The second is the text to be displayed in the Contents navigation pane, normally the topic title. This text is used by all the other navigational files, which reference the text by its index from the start of the (zero-based) `tocItems` array.

The third item is usually the link to the file to display. This can be a plain name, or a name with a hash value to specify an anchor within the file. In the latter case, the file loads directly at that anchor point, scrolled to the top if possible. (It might not be possible if the anchor is within the last screenful of the file.) If the file is already loaded, it scrolls to the new anchor; it's perfectly fine to have several contents items reference different points in the same file:

```
[2,"10 Splitting and extracting files","splits.htm#Xaa1069764"],
```

If, however, the third item begins with an asterisk, that item is not a link. Instead, it is the name of a subproject to be merged with the current project as needed. That subproject must appear in the `mergeProjects` array in `.ohx` (otherwise, the Contents item is not displayed); and that entry specifies the `tocItems` index of the subproject's Contents entry (see § Modular help on page 9). If the subproject fails merging, the asterisk is changed to an exclamation point, and the entry is no longer displayed:

```
[2,"11 HTML Linking","!oh11"],
```

Index: .ohk

Index entries use three distinct formats: a one-part format for sections, a two-part format for the index term, and a three-part format for the index references.

For the one-item terms, the content is normally a letter A-Z for aphabetic sections, 0 (zero) for any numeric (Num) section, and ! for any symbol (Sym) section. Each is used just once at the start of its section:

```
["F"]
```

For both of the other two items, the first item is the same: the Index level, a single digit from 1 to 9. (As with Contents, zero is unused and reserved.).

For the two-item index terms, the second item is always the text of the term to display:

```
[1,"FileName"],
[2,"custom marker for split/extracted files"],
```

For the three-item references, the second item is normally the link to the specific anchor that marks the location within the topic file where the index entry was placed. The last item is the index of the Contents entry in `tocItems`, whose title is used for the link text:

```
[3,"aa1069770.htm#Xaa1069772",50],
```

If, however, the first character of the second item in a three-item reference is a hash, then the reference is for a *See* or *See Also*, and the link is to a location in the index itself, where

it matches the text of its target entry. At present, OmniHelp accepts only top-level entry names; any added levels are trimmed off. The third item is then the actual text to be shown for the link, normally the same as the second item, but without the leading hash:

```
[2,"See also HTML Help, JavaHelp, Oracle Help"],
[3,"#Help","Help"],
```

Search: .ohs

A Search entry consists of a single text item containing the search string, followed by an array of numbers, with no extra comma at the end of the last number. The numbers are indexes of the Contents entries in `tocItems` where the search term can be found; both the title texts and links of the Contents entries are used:

```
["delete",[6,7,34,36,38,42,47,49]],
```

Related topics: .ohl

Related topics entries look exactly like Search entries; the text item is the ALink identifier, which is displayed as the Subject for the related topics:

```
["custom",[2,3,8,16]],
```

Context IDs: .oha

A Context ID entry consists of two text items. The first is the ID itself, which is never displayed, and the second is the file name of the topic to be displayed:

```
["hid_split","splits.htm"],
```

Open issues

OmniHelp is very much a work in progress, and all those interested are cordially invited to participate in its development. These are just a few of the areas that need help; no doubt numerous others could also benefit from some focussed attention.

Cross-browser issues

The main source of cross-browser problems centers on CSS usage, especially for browsers other than Internet Explorer, Opera, and the Mozilla-derived ones; except for these, testing has ranged from minimal to non-existent. We very much welcome any assistance other browser vendors might offer, to make sure that OmniHelp is a truly universal Help system.

Internationalization

Closely related to cross-browser support is internationalization. So far, we have been working only with US English. It’s likely that the Index character listing, and Index and Search list sorting, will prove unacceptable when accented characters are in common use. We’d like to add full support for international character sets, including Asian characters, regardless of current OS locale, and need the help of native speakers of other languages to do so.

Since all visible text, including button labels, is defined in the three `ohlang*.js` files, localization requires only replacement of the text in those files (currently US English) with equivalent text in the preferred language. This works regardless of the OS language support, or of the language used in the content.

Infotypes

We had originally planned to support infotypes, dynamic conditional text, in the initial release of OmniHelp. Implementation is not terribly difficult, using HTML `span` and `div` tags, and CSS to show and hide them. Even Boolean selection is not too difficult, requiring combination classes where conditions overlap. But the UI is still unclear, as is the way in which control should be divided between author and user. Obviously, hiding text does not remove it from the HTML source files, so there could be security to consider. We are continuing to work on these design issues.

User-interface issues

The main outstanding UI issues have to do with performance. On older hardware, with older browsers, JavaScript-based applications are just slow, and OmniHelp is no exception. We have done everything we can think of to move the compute burden from the display end to the authoring end, but still, for large documents (such as the 500-page User's Guide for Mif2Go that we use as a test case), some operations can take long enough to annoy users. All we can see to do about this is to proceed with implementation of a C++-based viewer, per our original plans for this project. This will not, however, lead to instant results, and so we continue to look for ways to accelerate the JavaScript version of OmniHelp.

We are also studying ways to allow users to easily modify OmniHelp settings. At present, we are thinking in terms of a new button, **Set**, in the top navigation pane at the far right. Clicking **Set** would display a form in the main pane (temporarily replacing the displayed topic) that would contain checkboxes, text boxes, and lists that the user could modify, with the desired values stored in persistent cookies. (There's no way to modify the JavaScript settings file from within the browser. Fortunately.) This could provide a convenient way to choose infotypes for display, and for specifying other projects to merge on the fly.

Authoring-tool issues

Currently, only Omni Systems Mif2Go offers OmniHelp output. That could change as other vendors become interested in the possibilities, both of major cost savings for themselves, and of improved functionality for their customers.

Participation in this open-source project provides significant benefits for authoring-tool vendors. The lack of an open standard for cross-platform Help has meant that major resources had to be assigned to the task of creating a proprietary solution. For some of the smaller vendors, that cost has been prohibitive, and so they have not created such a cross-platform solution. But even for larger vendors who have done this, the costs of code maintenance and of updates to deal with new browser and OS versions are burdensome. OmniHelp offers all vendors the opportunity to work together solving common problems, while still preserving their own proprietary value-added in the way they generate the project-specific OmniHelp files.

An additional new factor is the Eolas win over Microsoft, which will impact HTML Help and all browser-based Help that works with applets. This means that only pure JavaScript applications may continue to be royalty-free. It also means that MS may well drop from IE support for the <object> tag. OmniHelp is one of the very few alternatives not affected by the Eolas case.

