IRIS InSight™ Professional Publisher User's Guide

Document Number 007-2863-001

CONTRIBUTORS

Written by Jed Hartman, Debbie Myers, and Pam Sogard
Edited by Christina Cary
Production by Gloria Ackley
Engineering contributions by David Clarke, Victor Riley, Steve Schmitt, and Lorrie Williams

IRIS InSight™ Professional Publisher User's Guide
Document Number 007-2863-001

Contents

List of Figures

List of Tables

About This Guide

This guide explains how to prepare and produce online technical documentation using the IRIS InSight™ Professional Publisher online-document-building tools. Users can read online manuals using the IRIS InSight™ document viewer, proprietary software that exploits the graphics capabilities of Silicon Graphics® workstations.

Some documentation ("online help") is intended to be viewed through SGI Help, an application derived from IRIS InSight. SGI Help presents discrete pieces of information about an application from within the application, as opposed to the book-oriented approach of IRIS InSight. Like IRIS InSight, however, SGI Help allows readers to view portions of technical manuals and to launch other applications.

To create and edit manuals and help material to be viewed with IRIS InSight or SGI Help, use the FrameMaker® document publishing application from Frame Technology® Corporation. Use the FrameMaker tags and templates provided as part of InSight Professional Publisher to structure document files so that they can be converted to Standard Generalized Markup Language (*SGML*) for online viewing. For manuals delivered in both print and online versions, both versions are derived from a single FrameMaker source file.

The InSight-viewable version of a technical manual is delivered as an installable software image, usually a subsystem within a software product. For this reason, you must put the component files of a viewable book through a special preparation process referred to as a *book-build*. The book-building process translates FrameMaker source files to SGML using the *SGIDOC DTD*, and generates the additional files that are required for the installable software image.

Audience for This Guide

The audience for this guide includes writers who are responsible for revising existing technical manuals and creating new ones, and production editors who prepare a writer's document source files for delivery to print vendors or to a software release group. (It's assumed that a *production editor* is associated with the book and knows something about your company's document-production issues. If you don't have a production editor, you may have to handle production tasks yourself.) Experienced writers and production editors who are familiar with the Silicon Graphics book development process can refer to this guide for selected procedures and points of information. New writers and production editors should rely on this guide to learn the tools and processes with which to develop online manuals.

Readers of this guide are assumed to be experienced FrameMaker users.

Organization of This Guide

This guide is organized to present the book development procedures in an unbroken sequence. These procedures are described in Chapters 1 through 4. Information on using FrameMaker features (Chapter 5) and preparing figure files (Chapter 6) is presented after the book development process; but readers are advised to refer to Chapters 5 and 6 routinely throughout the book development process.

- Chapter 1, "Getting Started," recommends ways to become familiar with the InSight viewer and to adjust your writing style for the online reader.
- Chapter 2, "Preparing Your Workstation," explains how to organize the area of your workstation that is used during book development and how to set up the software that you will use.
- Chapter 3, "Building Online Books," contains the procedures for converting FrameMaker source files into an online manual. It also explains how to view an online book created by the book-building process.
- Chapter 4, "Finding and Fixing Online Book Errors," explains how to locate and correct problems that occur in the book-building process.
- Chapter 5, "Using FrameMaker Files and Template Features," explains how to use the Silicon Graphics document templates and their associated tags in creating a FrameMaker source file. It also explains file naming conventions and the specifics of applying FrameMaker tags to ensure the successful build of an online manual.

- Chapter 6, "Working With Figures," identifies acceptable formats for figure files and explains the purpose of directories where figure files are stored. It also explains how to process supported and unsupported figure types so that they are viewable in an online manual.
- Chapter 7, "Creating Online Help" explains how to create online help for the SGI Help viewer. Help information can be either a separate document or part of an online manual.

In addition, this guide contains the following appendices:

- Appendix A, "Quick Reference: Rules for Using the Templates"
- Appendix B, "Capturing Images for FrameMaker Documents"
- Appendix C, "IRIS Insight Book-Building Architecture"

Supplementary Reading

Refer to these documents to supplement the information in this guide:

- *Designing and Writing Online Documentation: Help Files to Hypertext*, by William K. Horton. Published by John Wiley and Sons, Inc. For writers who want to know more about writing for the online medium.
- *Illustrating Computer Documentation: The Art of Presenting Information Graphically on Paper and Online*, by William Horton. Published by John Wiley and Sons, Inc. Provides guidelines for illustrating and designing effective graphics in either the paper or online medium.
- *IPTemplate*. Published by Silicon Graphics as part of the InSight Professional Publisher distribution. These are the templates used to produce manuals. The *IPchap.doc* file in the */usr/share/Insight/templates/frame/IPTemplate* directory contains information on how to use the various tags and features of the templates.
- *Using FrameMaker* and *FrameMaker Reference*. Published by Frame Technology Corporation. These books document basic and advanced FrameMaker features.

Chapter 1

Getting Started

This chapter explains what to do to prepare for the book development process. The chapter recommends ways to learn about the online medium and how to adjust your writing style for the online viewing audience. It also gives an overview of the book development process and tells you where to look in this book for an explanation of each step in the process.

This chapter contains these sections:

- "Before You Begin"
- "Viewer Features to Keep in Mind"
- "Guidelines for Organizing an Online Book"
- "Guidelines for Cross-Referencing in Online Books"
- "Overview of the Book Development Process"

Note: The writing guidelines in this chapter are very brief. For a more thorough discussion of effective online writing style, refer to the books listed as "Supplementary Reading" on page xiii.

Before You Begin

Before you write or revise a book for online presentation, take a look at an existing online book. Start by launching IRIS InSight if it's not already running; then try some of these features:

1. Select a topic from the Help menu on the InSight main window's menu bar.
2. Open a book by double-clicking its title.
3. Click some hypertext elements:
   - blue text (a link to a location in the current book)
   - red text (a link to a location in a different book, or to an executable)
   - underlined word or phrase (a glossary item)
   - rectangle containing an arrow pointing right then curving upward (an icon to launch an application)
   - rectangle containing an arrow pointing straight right (a margin note or a footnote)
4. Navigate through the book. Try using the Table of Contents, the List of Examples, the List of Figures, and the List of Tables (all accessible through the View submenu of the Book menu) and the Index (choose Index from the Book menu).
5. Click a figure to bring up a restricted view.
6. Click a table title to bring up a restricted view.
7. Use the text search feature.
8. Read through an entire chapter to get a feel for the flow of online text.

Encourage engineering reviewers to review both the soft copy and the hard copy of your book. This is particularly important if your book is not shipped in hard copy by default.

Viewer Features to Keep in Mind

When writing an online book, it might help you to be aware of the IRIS InSight features that are available to your readers. Readers can

- search for specific text within one book or across an entire library
- navigate through a book by clicking entries in the Table of Contents, List of Example, List of Figures, List of Tables, or Index, or by clicking cross-references
- change font sizes in book windows
- open several books at the same time
- retrace their steps within a session
- get help by way of the Help menu
- print sections, chapters, or entire books
- launch applications by clicking launch icons within InSight
- view images, movies, and IRIS Inventor™ models inline
- view images and tables either inline or in their own separate windows

Guidelines for Organizing an Online Book

This section includes some tips for writing good online documentation. These tips are primarily aimed at preventing users from getting lost in the online version of a book. When writing your online book, remember that users might not see the pages in order. Further, they might not know exactly where they are in the book at any given time.

Here's an overview of basic online writing tips:

- Write in chunks. Try to make each section answer one question. Provide links to supplemental information.
- At the beginning of each chapter, provide a list of links to each of the chapter's major headings. See "Provide a Linked Chapter Summary" on page 4 for details.
- Write good section headings. Make them distinct and make them descriptive enough to stand alone. See "Write Descriptive, Standalone Headings" on page 4 for details.

- Be consistent in your use of terms.
- Avoid print-oriented wording. See “Avoid Print-Oriented Wording” on page 5.

“Chunk” the Text

Usability tests indicate that readers get lost after scrolling through more than three windowfuls of text without headings. Your book will be easier to read and navigate online if you organize text in *chunks*: brief, to-the-point, stand-alone sections. Think of the topics in your book as index cards scattered on a desk. Readers may jump to a topic without seeing what comes before or after it. Make each section answer one question, then provide links to supplemental information.

Provide a Linked Chapter Summary

At the beginning of each chapter, provide a bulleted list of all first-level headings in that chapter. Usability tests indicate that, when a chapter begins with a bulleted list, users expect the items in the list to be links. Use the FrameMaker cross-reference tool to create the list, so that each section heading will be a link in the online version of the book.

Of course, in some books it may not make sense to begin each chapter with a linked summary. As always, use your best judgment of what is appropriate for your book.

Write Descriptive, Standalone Headings

A good online heading tells the reader what is contained in the section without relying on information contained in the parent heading. For instance, if a section called “Comparing Files” contains a subsection that describes how to use *diff*, name the subsection, “Using *diff* to Compare Files,” rather than simply “*diff*.”

Since your readers may enter the section from a different chapter or even a different book, using “*diff*” by itself as a subsection title doesn’t provide enough context information; it requires the reader to skim the previous section and to know what the *diff* utility is used for.

To make it easy for a reader to stay “located” in your book, stick to one topic per section. If the subject changes, write a new heading. Don’t worry about having too many headings; a large number of headings will make your document more usable and won’t overload it.

Use Terms Consistently

Online readers leap quickly from section to section. While looking for information on a particular topic, a reader might consecutively read eight different paragraphs from various chapters in your book, and others from another book as well. So, for example, if you call something a “short name” in one chapter and a “short title” in another, and another writer calls it a “shorty” in a different book, readers may get confused. This kind of inconsistency is inadvisable, at best, in a hard copy book, but it’s deadly online. If you change a term partway through the writing process, use FrameMaker’s find-and-change tool to make sure you change all the occurrences of the old term.

Avoid Print-Oriented Wording

Online readers approach information from many different directions. Don’t assume that a reader knows or has read information that isn’t currently on the screen. Avoid wording that refers to location within a book, unless you also create a link to that information. Table 1-1 gives examples of how to avoid print-oriented wording.

Table 1-1 Suggestions for Avoiding Print-Oriented Wording

Use...	**Instead of...**
“in the section named...”	“as shown above (or below)”
“in Chapter 3, you learned that...”	“by now you have learned that”
“in the previous section, <section title>,”	“in the previous section”

Guidelines for Cross-Referencing in Online Books

In the online version of your book, local cross-references appear in blue and are linked to the referenced material. This gives your readers another way to navigate the book. Well-designed cross-references get information to the people who need it, and keep it out of the way of those who don't. Poorly-designed cross-references can lose readers in annoying and confusing wild-goose chases. This section offers some guidelines for good cross-referencing.

Write Informative Cross-References

Include enough information in a cross-reference so a reader can decide whether to follow it. For example, "See the *Indigo Owner's Guide*" doesn't tell the reader what information you are referencing. A better reference might be, "See the section called 'Setting Up Your System' in the *Indigo Owner's Guide* for more information about plugging in the power cord." This reference tells the readers exactly what information they can expect to find at the other end of the link.

Cross-Reference Repeated Information

Don't include the same information in multiple places in your book. Instead, put it in the most important place, then create cross-references to it as necessary. This way, readers who need the information can just click the cross-reference, while readers who don't need it won't need to scroll through it. And when you need to update the information on that topic, you won't have to remember to update it in more than one place.

Use Cross-Book References Carefully

Linked cross-references between different books are one of the most useful features of online documentation. They do, however, involve risks. Another writer could change the organization, the content, or even the name of the referenced book. Or the book might be on a different release schedule, which means your cross-reference might be destroyed when a new version of the referenced book is released. Here are some guidelines to minimize such risks:

- Avoid gratuitous cross-book references.
- Notify the authors of books that you plan to cross-reference. The writers can give you an idea of how far along their books are in the revision cycle and whether the content and organization are stable.
- When referring to documentation for an optional product, make it very clear that the document is optional and that the reader might not have it: “If you purchased the C++ option, see the *C++ User's Guide* for more information.”
- The safest cross-references are to local books and books that ship with the same software product as your book.

Overview of the Book Development Process

The book development process comprises two general steps:

1. Prepare your workstation.

   The first step in developing a book is to set up the software tools on your workstation. See Chapter 2, “Preparing Your Workstation.”

2. Build and debug the online version of your book.

   Build the online version of the book periodically; see Chapter 3, “Building Online Books.” If you encounter errors during the build process or while viewing the completed online book, see Chapter 4, “Finding and Fixing Online Book Errors.”

Chapter 2

Preparing Your Workstation

The book development process requires access to the book-building and viewing tools.

This chapter explains how to prepare your workstation for book development. The chapter contains these sections:

- "Summary of Software/Configuration Requirements"
- "Setting Up Tools and Environment"
- "Organizing a Local Working Directory"

Summary of Software/Configuration Requirements

When you complete the procedures in the remaining sections of this chapter, your workstation will be configured with the software required to build, view, and archive online books. The following items are required:

IRIX 5.3	The base operating system. The InSight Professional Publisher tools will not run under IRIX 5.2 or earlier, or under IRIX 6.1.
FrameMaker	You must create your documents using FrameMaker 3.1.X or later in order to use the tools.
book-build tools	The translation and figure-processing tools that convert FrameMaker files and associated images to an InSight-viewable format. (This conversion process is called a *book-build*.) These tools are part of the InSight Professional Publisher distribution, in the insight_dev.sw.tools subsystem.

smake utility	A parallel version of *make*, a program that uses a specified rule set to invoke the appropriate translation and figure-processing tools under various book-building conditions. *smake* allows multiple tool invocations at once, which lets your book build faster. *smake* is automatically invoked when you use any of the *make* commands documented in this guide, so you can pretend you're just using *make*. The *make* and *smake* utilities are probably already installed on your workstation; they're part of the software subsystem dev.sw.make. To use these utilities, you also need the dev.hdr.lib subsystem.
InSight software	The InSight viewer, support files for InSight, and help files for InSight and SGI Help. This software should already be installed on your workstation; it's part of the standard IRIX distribution.
other tools	You need the following additional software subsystems installed in order to make full use of the book-building tools: eoe2.sw.imgtools (basic figure processing tools), and the PostScript-processing tools in impr_rip.sw.impr and impr_fonts.sw.adobe22.
revision control software	Silicon Graphics publications groups check files into and out of a document archive using tools based on RCS (the Revision Control System). If your company uses such a system, you'll need the appropriate revision-control software. This book assumes that you have some sort of system for checking files into and out of an archive; if you don't use such a system, just ignore any references to checking in or out.

Setting Up Tools and Environment

To make a book viewable online, your workstation must have access to the tools that process FrameMaker files into files that can be viewed with IRIS InSight or SGI Help. In addition, your workstation must be running IRIS InSight software so that you can check the online book for problems during development.

Checking the *make* Utility

The dev.hdr.lib and dev.sw.make subsystems must be installed on your workstation before you can use the book-building tools. The installed version of *make* must be from the same base operating system release as the version of IRIX that you are running; that is, if you're running IRIX 5.3, you must use the IRIX 5.3 version of *make*. To compare your version numbers, enter this command:

```
% showprods -D 3 dev.hdr dev.sw.make eoe1
```

If the IRIX version numbers listed for eoe1 and dev are different, install the correct version of *make*. See your system administrator if you have questions about software installation.

Installing the IRIS InSight Viewer and Help

The IRIS InSight viewer occupies about 2 MB of disk space without books. It's installed by default, but if it isn't already installed on your workstation you can install it using Software Manager, the standard Silicon Graphics graphical installation utility.

Note: If you need help running *swmgr*, see your system administrator or the *Personal System Administration Guide*.

Here's how to install InSight:

1. Become superuser:

   ```
   % su
   ```

2. Execute Software Manager:

   ```
   % swmgr
   ```

3. Specify the location of the InSight software distribution in the Available Software text region, then click the Lookup button.
4. Click the Customize Installation button. After a brief processing period, a list of InSight software appears.
5. Select Short Product Names from the Software menu, if it isn't already selected. (If it is, leave it selected.)

6. Click the folded arrow icon to display a list of subsystems. The folded arrow icon appears between the Install checkbooks and the product name.
7. Select these subsystems by clicking the Install checkbooks:
   - insight.books.ViewerHelp
   - insight.books.help
   - insight.sw.client
   - insight.sw.data
   - insight.sw.public
   - insight.sw.sgihelp
8. Click the Start button to install your selections.
9. Click the OK button when the successful installation message comes up.
10. Exit Software Manager, using the Quit command in the File menu.

Organizing a Local Working Directory

A working directory contains working copies of the files that compose a book, as well as the subdirectories and support files that are needed to produce hard copy and online versions of the book (see Table 2-1). Such a directory is often called a “local working directory” in environments which use a document archiving system; in such an environment you copy the archived document into a working directory on your local system, and make any changes to that local copy.

You may put extraneous files in a working directory and create any extra subdirectories that you wish; however, all files that are actually part of the book must be named according to prescribed naming conventions. These conventions ensure successful book builds and smooth handoffs to other people who need to use your files.

Note: Refer to the information in this section throughout the development process to be sure that your working directory contains the required set of files and subdirectories in the correct location.

Standard Files in a Local Working Directory

Figure 2-1 illustrates the source files that compose the hard and soft copy versions of a technical manual. These files should be archived periodically during the book development cycle. The asterisk (*) indicates files and directories that are unnecessary for books that ship only in hard-copy format.

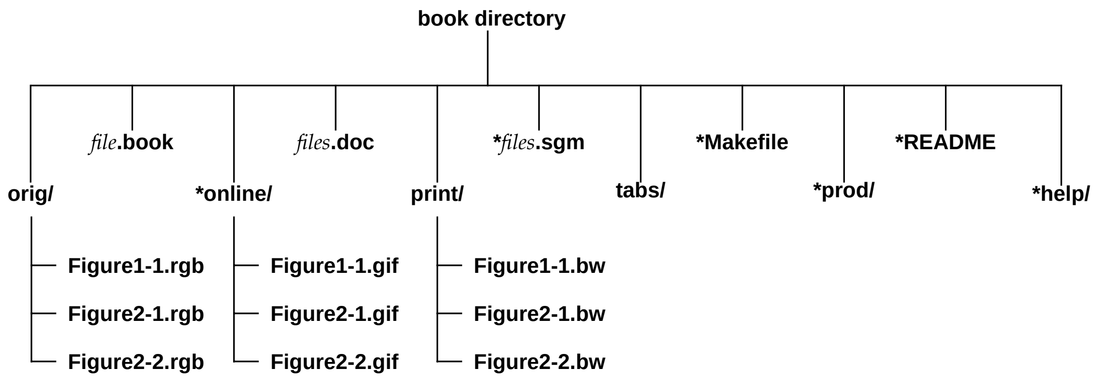


Figure 2-1 Files and Directories in a Working Directory

Table 2-1 describes the files and directories shown in Figure 2-1. The asterisk (*) indicates files and directories used only for online books.

Table 2-1 Files and Directories in a Book Directory

File	Description
print directory	Contains the images used in figures for the printed version of the book. To generate and populate the *print* directory, use the command *make _print*. All figures in your FrameMaker files should be imported by reference from this directory.
orig directory	Contains the original images for all figures in the book. Files are either complete figures or figure components that must be processed before storing in the *print* or *online* directories.
**online* directory	Contains the figures for the online version of the book, generated during the build process.
**help* directory	Contains the helpmap file for linking SGI Help to the book. See Chapter 7, “Creating Online Help,” for details.

Table 2-1 (continued) Files and Directories in a Book Directory

File	Description
tabs directory	Contains *chapter tabs* for the book. *Part tabs*, if any, must be in the same directory as the document files.
FrameMaker document files (with *.doc* suffix)	Includes all FrameMaker files for the book: front matter, introduction, chapters, appendices, glossary, and all generated files (TOC, LOF, LOT, and IX).
filename*.book*	The FrameMaker file that contains information about all component files for a book.
*SGML files (with *.sgm* suffix)	The SGML translations of the *.doc* files; *.sgm* files are the components of the online version of your book. Building a book (with the *make* command) generates these files.
**Makefile* file	Specifies information about how to build your book. The *Makefile* is used to generate the SGML files for the online version of the book, as well as to generate the various forms of the figure files.
**README* file	Contains the writer's name, the date, and the full title, part number, and short name of the book. In addition, gives any other information about the book that might be helpful to future writers, production specialists, or build engineers.
**prod* directory	Contains information that might be used by the writer or production editor for purposes such as storing print vendor specifications.

Observing Naming Conventions

Observe these conventions for naming FrameMaker files:

- All FrameMaker document files (front matter, introduction, chapters, appendices, glossary, and all generated files) must end with the suffix *.doc*.
- The front matter file should include the string "front" in its name (*PSAGfront.doc*, for example)
- The file containing the introduction to the book—called "About This Guide" in Silicon Graphics documents—should include the strings "about" or "intro" in its name (*00.about.doc*, for example)

- Chapter filenames should include the chapter number in their names. (*01.getstart.doc*, for example, might be a good name for the first chapter of a book.) Starting the filename with the chapter number means that the filenames appear in the correct order in file-open dialog boxes, when listed by *ls*, and so forth. Including a mnemonic word in the file's title can be enormously helpful if you ever subsequently add or remove chapters in the middle. Thus, filenames like *09.debug.doc* are preferable to names like *ch9.doc*.
- Appendix filenames should include the appendix letter in their names (*C.variables.doc*, for example, for the third appendix in a book). Again it's useful to start with the sequence letter and to include a key word, rather than using filenames like *AppA.doc*.
- To name your book file, first choose a title for your book, then abbreviate it to form a *short title*. See "Choosing a Book's Title" and "Choosing a Book's Short Title" for help with this choice. Name the book file *short_title.book*, where *short_title* is the short title you've chosen for the book.
- Your FrameMaker-generated TOC, LOF, LOT, and Index files should end with the suffix *.doc*. For example, if your book file is called *MyBook.book*, the generated files are *MyBookTOC.doc*, *MyBookLOF.doc*, *MyBookLOT.doc*, and *MyBookIX.doc*. FrameMaker chooses these filenames by default when you generate the files within a book.

Choosing a Book's Title

For consistency, it's a good idea to follow standard conventions in naming a book. Silicon Graphics has complex guidelines on choosing a title for any kind of document, including many unusual cases. Luckily, most titles follow a fairly simple structure:

product_name audience_type document_type

The *audience_type* indicates who the book is aimed at, or the kind of task that the book describes. Choices include (among other possibilities):

- Installation
- Language
- Programmer's
- System Administrator's
- User's

The *document_type* indicates what kind of book it is:

- Overview
- Tutorial
- Guide
- Reference Manual

By mixing and matching from the two lists, you should be able to choose a title relatively easily. Here are some examples of titles of Silicon Graphics book titles:

- *ImageVision Library® Programmer's Guide*
- *IRIS® Volume Manager System Administrator's Tutorial*
- *OpenGL® Porting Guide*
- *WebMagic™ User's Guide*

Choosing a Book's Short Title

A short title is an abbreviated way to refer to a book. A book's short title should be a condensed version of its full title, no more than twelve characters long. Short titles are used in a variety of ways: as the titles of icons representing iconified InSight books; to refer to other books when making cross-book links; and to identify books which contain an item a user has searched for in the main InSight library window.

Short titles follow conventions similar to those used in the full titles of books; for instance, the short title of a user's guide should end with "_UG", while that for a programmer's guide should end with "_PG." Also, the short title of a book is used to identify its subdirectory in the InSight book directories, so you can't use spaces or other characters that are illegal in a filename (such as * or /).

Be sure to list your book's short title on the TITLE line of your *Makefile*, and in your *README* file.

Note: Do not use the short title of your book as the basis for the name of any file other than the book file. For example, *short_title.doc* (where *short_title* is the short title of your book) is not a valid name for a chapter file.

Copying Template Files to Your Working Directory

Use the template files provided with InSight Professional Publisher to create new FrameMaker files for your book. These template files are located in the directory */usr/share/Insight/templates/frame/IPTemplate*. That directory contains a subdirectory called *samples* which contains a sample book that uses the template. If you're unfamiliar with this template, start by launching FrameMaker and opening the file *IPChap.doc* from the *samples* directory; that file contains information on all tags and formatting procedures for FrameMaker documents that will be converted to InSight.

In addition to FrameMaker template files, the book-building tools also require a *Makefile*. This file specifies the characteristics necessary to prepare a document for online viewing—particularly information about what files are part of the document to be built. Copy the file */usr/share/Insight/templates/make/Makefile_books* into your working directory, rename it to *Makefile*, and then set the permissions so you can both read and write it:

```
% cd working_directory
% cp /usr/share/Insight/templates/make/Makefile_books ./Makefile
% chmod 644 Makefile
```

Once you've set up your *Makefile*, you have to edit it to make it work with your particular book. Directions for editing the *Makefile* are given in Chapter 3, "Building Online Books."

Chapter 3

Building Online Books

The book-building process creates SGML files from FrameMaker source files. It also generates the additional files that are required to view the SGML using the IRIS InSight viewer or the SGI Help viewer. The build process also generates error reports to help you debug source files and create error-free online documents.

To invoke the build process, use the *make* command. You can give *make* a variety of arguments to perform different build-related tasks. The *make* command reads a file called *Makefile*, which describes the book and the source documents using a set of variables. Once a book is built, writers and production editors can proofread it while viewing it in IRIS InSight, and/or test the resulting online help.

This chapter explains how to build a book, generate error reports, and view the built book with IRIS InSight. The chapter includes these sections:

- "Editing the Makefile"
- "Files and Directories Generated by make"
- "Arguments to the make Command"
- "Using make Commands"
- "If make Doesn't Work"
- "make book Error Messages"
- "Viewing a Built Book"

Note: The build process goes more smoothly if you compose and import figures, and then use the Generate/Update command in the FrameMaker book file, before beginning the build.

Editing the Makefile

The *Makefile* specifies the FrameMaker files that compose the book and gives information about the figure files in the book. The *Makefile* also specifies the short title and full title of your book, the part number and date, any foreign language attributes, and the bookshelf where the book is displayed. An edited version of the *Makefile* that contains the specifications for your book must be stored in your local working directory to build the book.

Note: Be prepared to supply the short title (see "Choosing a Book's Short Title" in Chapter 2), bookshelf, and a list of your document and figures files during this procedure. (Though you can change any of these things and rebuild the book later if necessary.)

The *Makefile* is self-documenting; it contains instructions on what to add and where to add it. The procedure that follows provides supplementary information for editing *Makefile*.

1. Edit the TITLE variable:

   Specify the short title for your book as the value for the TITLE variable. For help choosing a short title, see "Choosing a Book's Short Title" in Chapter 2.

   Note: Do not use the short title of your book as the basis for the name of any files in your document other than the book file. For example, if *short_title.book* is the name of your book file, then *short_title.doc* is not a valid filename (though *short_titleCh1.doc* is a valid filename).

2. Edit the FULL_TITLE variable.

   Specify the full title for your book as it appears on the cover. For help choosing a full title, see "Choosing a Book's Title" in Chapter 2.

3. Edit the VERSION variable.

   Specify the version number, month and year for this revision. The version number for a Silicon Graphics book is the part number for the book. For example:

   ```
   007-2863-001, 9/95
   ```

4. Edit the BOOKSHELF variable.

   Specify the directory in which your company's bookshelves are located and bookshelf on which your book is to appear in the InSight library, in this form: */usr/share/Insight/library/*companyname*_bookshelves/*companyname. (You can use a standard abbreviation of your company's name as your *companyname* if you wish, but in that case all books from a given company must use the same abbreviation.) For instance, a company named YoyoDyne Systems would specify */usr/share/Insight/library/YoyoDyne_bookshelves/YoyoDyne* for the BOOKSHELF variable; the books would then appear in InSight under a bookshelf named "YoyoDyne." Note that since *companyname* is part of a directory name, it must follow all rules for UNIX filenames—for instance, *companyname* can't contain characters such as spaces, slashes, or asterisks.

   Silicon Graphics publications can appear on any of three bookshelves: SGI_EndUser, SGI_Developer, and SGI_Admin. Field Engineer manuals go on the SGI_Service bookshelf. All Silicon Graphics bookshelves are in the */usr/share/Insight/library/SGI_bookshelves* directory.

5. Edit the BOOK_ICON variable.

   Specify which icon should represent your book on the bookshelf in InSight. Valid icon choices for Silicon Graphics publications are sgi_end_user, sgi_developer, and sgi_admin. Silicon Graphics Field Engineer manuals use the "binder" icon. Books developed outside of Silicon Graphics use the "generic" icon, which is the default.

6. Edit the BOOK_FILES variable:

   Specify the source files that compose your book, in the order in which they appear in the book. Include the front matter, introduction, chapter, appendix, and glossary files in the order in which they appear in the book. Do not include any generated files (index, TOC, LOE, LOF, or LOT). Also leave out part and chapter tabs.

Table 3-1 lists the FrameMaker files that you should and should not include in the BOOK_FILES variable.

Table 3-1 FrameMaker Files to Include and Omit in Makefile

Include	Omit
Front matter file	Table of Contents
About This Guide or Introduction file	List of Figures
Chapter files	List of Tables
Appendix files	List of Examples
Glossary file	Index
	Book file

7. Specify figure information:

   Use the information in "Adding Image Files to the Makefile" on page 72, to complete this step. The *Makefile* file contains several variables that you can edit to specify how to process your figures files to display them in the online version of your book. When you assign values to figure variables, keep these facts in mind:

   - The *make* commands can only process the original figure files in the *orig* directory. This processing populates the *print* and *online* directories with correctly formatted figures.
   - The *print* directory is generated by the *make _print* command; it contains figure files that are processed specifically for the paper version of a book. All figure files that are imported into FrameMaker files must be imported from *print*—so if you create a new figure in the *orig* directory, you must use *make _print* to put a usable version of that figure in the *print* directory before you can import the figure (from the *print* directory) into your book.
   - The *online* directory is generated by the *make _online* command; it contains figure files that are processed specifically for the online version of a book.

8. Edit the INLINE_MEDIA variable, if needed.

   Specify the name of inline media objects for movie, audio, or Inventor files if your book contains any.

Files and Directories Generated by *make*

You can run *make* with a variety of different arguments. The argument that you use determines the generated files that result from the build. The *make* command produces four different types of files: files that are actually part of the online book; image files for use in figures in the printed version of the book; intermediate files; and error files. Intermediate files are artifacts of the build process. *make* must produce these files in the course of creating an online book, and the files can be used to speed up subsequent book builds when only part of the book source has changed. The intermediate files are also used to create error files, which are reports of various types of source errors detected during the build process. Error files are not necessary steps in the build process and, though it is generally not recommended, it is possible to create an online book without generating any error files.

Note: For instructions on how to find and fix errors reported in the various error files, see Chapter 4, "Finding and Fixing Online Book Errors."

Since the build process is a sequence of steps, *make* commands that take source files closer to a finished online book necessarily require that all earlier steps are completed. Likewise, generating an error file that relies on certain intermediate files will cause those intermediate files to be updated, if necessary, before an error report is generated.

Figure 3-1 shows where the various intermediate and error files are produced during the build process.

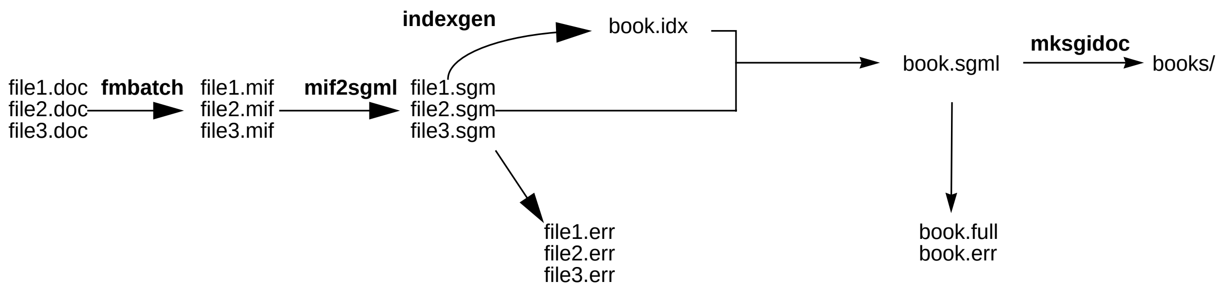


Figure 3-1 Files Produced During the make Process

Intermediate Files

The *make* command generates a sequence of intermediate files in the process of creating an online document. Some are descendants of individual source files, while others combine information generated from all the source files for a book. All intermediate files appear in your document directory. They are generated automatically as part of the online build process. By using different *make* commands, you can produce a single file or set of files instead of all the generated files.

.mif files	These are ASCII versions of the FrameMaker *.doc* files, in Frame's Maker Interchange Format. The *mif2sgml* translation software reads and converts these files to SGML.
.sgm files	These are individual SGML files, produced from the *.mif* files by the *mif2sgml* translation software. One *.sgm* file is generated for each FrameMaker file listed in the *Makefile*.
short_title.idx file	This file contains index information for the SGML version of the book.
short_title.sgml file	This file contains the concatenated SGML for all the files listed in the *Makefile*, plus all the information from the *short_title.idx* file.

Error Files

Error files list errors found in the various intermediate files. For instructions on how to find and fix errors, see Chapter 4, "Finding and Fixing Online Book Errors." For information on error files, see "Error Files" in Chapter 4.

The *books* Directory

The *make book* command produces a subdirectory called *books* in your working directory that contains the built version of the book. See "Viewing a Built Book" on page 33 for more information.

Arguments to the *make* Command

The *make* command takes many arguments, giving you flexibility to build different pieces of your book or selected error files. Since the build process is a sequence of steps, *make* commands that require more steps necessarily take longer. Figure 3-1 shows the basic sequence.

Note: The dev.sw.make and dev.hdr.lib subsystems must be installed on your workstation to use the *make* command. The installed versions of those subsystems must be from the same base operating system as the version of IRIX that you are running. See "Checking the make Utility" on page 11 for details.

Table 3-2 lists the commands that operate on all files in a book. To perform any action listed in the table, enter the command exactly as shown.

Table 3-2 *make* Commands For the Entire Book

Command	**Output**
make book	A *books* subdirectory containing a viewable book.
make book.err	*shortname.err* file, containing all ERROR and WARNING messages for the book.
make book.full	*shortname.full* file containing all error information, including link error reports and unresolved glossary terms; also a *books* subdirectory containing a viewable book.

Table 3-3 lists the commands for individual FrameMaker files.

Table 3-3 *make* Commands for Single FrameMaker Files

Command	**Output**
make *filename*.mif	FrameMaker MIF file from *filename.doc*
make *filename*.sgm	SGML file from *filename.mif*
make *filename*.err	error/warning report on *filename.sgm*

Table 3-3 lists the commands that operate on figure files. Note the underscores; *make* can get confused when its argument is both a build rule and a directory name, so you should use *make _print* and *make _online* when building figures instead of just *make print* and *make online*.

Table 3-4 *make* Commands for Figures

Command	Output
make _print	newly updated figure files in the *print* directory
make _online	newly updated figure files in the *online* directory (note that *composite figures* are not updated by this command; to update composite figures, you must rebuild the *.sgm* file that contains them)

Table 3-5 lists the commands that remove previously-built files. Note that none of these commands removes your source files (such as your FrameMaker document files and the files in the *orig* subdirectory). However, take care in using these commands if you've modified any generated files (such as image files in *print* or *online*) by hand.

Table 3-5 *make* Commands for Removing Files

Command	Effect
make clean_book	Removes the *books* subdirectory (which contains the online book, if previously built) and all generated MIF files, plus the *short_title.sgml* file
make clean_reports	Removes the error report files, including the *.full* file
make clean	Removes all of the above, as well as any temporary files that have been left in place
make clean_figures	Removes all files from the *online* subdirectory
make clean_sgm	Removes generated SGML files
make clobber	Removes all of the above
make clean_print	Removes files in *print* subdirectory
make clobber_all	Removes all of the above

Using *make* Commands

Which argument you should choose for the *make* command depends on the stage of development of your book. Use these recommendations as a general strategy for choosing a *make* argument:

- Run *make book.full* on a book fairly early in the development process to generate error reports that you can use to begin debugging. Try to fix as many of these errors as possible before creating your first viewable book.
- After fixing most of the errors from your initial *make book.full*, you can produce *.err* files for individual source files, as needed, as you continue development. You should also build the entire book periodically using *make book* or *make book.full* to monitor the progress of your online book.
- Finally, before you incorporate the final book files into your software build, you must run *make book.full* and verify that your *shortame.full* file contains no errors you should fix. Then make sure the newly generated book is viewable with InSight, that all figures and tables work, and that the book looks the way you want it to look.

The *make* commands provide some information as they run; this information appears in the window from which you issued the command. If a *make* command encounters an unresolvable error, it reports an error message. If a *make* command terminates abnormally, see "If make Doesn't Work" on page 30 for suggestions on finding and fixing the problem.

For instructions on how to find and fix errors that appear in the various error reports generated by *make*, see Chapter 4, "Finding and Fixing Online Book Errors."

Initial *make*: *make book.full*

As a first step in the debugging process, use the *make book.full* command to build the SGML files with full error reports. (This command also generates a viewable copy of the book.) These reports give you an idea of how far along your book is, and how much work it may take to get it online. Try to eliminate as many of the easy, obvious errors as you can before you try and view your book. After you've fixed most of the errors, you can rebuild the viewable book using *make book* or *make book.full*.

To run *make book.full*, follow this procedure:

1. Close all your FrameMaker files. (The files must remain closed until the MIF files have been generated from them; at any point after that you may reopen them.)
2. Change directories to your local working directory.
3. Generate the SGML files and error reports.

```
% make book.full
```

This converts your FrameMaker files into SGML and creates a complete error report file that you can use to debug your book. (Chapter 4, "Finding and Fixing Online Book Errors," explains how to do this debugging.) This complete error report is called *short_title.full*, where *short_title* is the *short title* of your book. The *make book.full* command also generates individual *.err* files, one for each chapter of the book; you can examine those files individually if you prefer, instead of the entire *.full* file.

make book.full may take anywhere from a minute to half an hour depending on the size and contents of the book.

Debugging Individual Files: *filename*.err

There's no need to rebuild the entire book every time you make a change. This takes time and CPU cycles, and in many cases the amount of noticeable change is fairly minimal. Usually, a better strategy is to clean up and rebuild files individually, rebuilding the entire book only when you feel you have accumulated significant changes.

You can produce an error report (a *.err* file) for either individual files or the complete book. The *.err* files list errors pertaining to document structure as well as information about incorrect tag use. To produce a *.err* file for a single source file, type:

```
% make filename.err
```

To produce a *.err* file for an entire book, replace *filename* in the above example with the book's short title.

See Chapter 4, "Finding and Fixing Online Book Errors," for instructions on fixing errors.

Making a Viewable Book: *make book*

If you want to view your book online and aren’t interested in the error reports, follow these steps:

1. Close all your FrameMaker files. (The files must remain closed until the MIF files have been generated from them; at any point after that you may reopen them.)
2. Change directories to your local working directory.
3. Generate the book.

   ```
   % make book
   ```

This command builds the files needed to display your book in the IRIS InSight viewer. “Viewing a Built Book” on page 33 explains how to view the book in IRIS InSight.

Verifying an Error-Free Book: *make book.full*

The last thing you should do before checking in final book files is to run *make book.full* to verify that you have cleaned out all the errors in your book and to make sure that your online book builds. Then check the newly built book with the InSight viewer to make sure it looks the way you want it to.

To build your final book:

1. Close all your FrameMaker files. (The files must remain closed until the MIF files have been generated from them; at any point after that you may reopen them.)
2. Change directories to your local working directory.
3. Type the *make* command.

   ```
   % make book.full
   ```
4. Check the error files to make sure there are no errors.
5. To view the online book, type:

   ```
   % iiv -f -b .
   ```

If *make* Doesn't Work

If a *make* command doesn't work, use these suggestions to correct the problem:

- Try to find and interpret the error message. Usually, the final cryptic "error code 2" (or similar error) will have been preceded by a more coherent message farther back in the build output. Look for phrases such as "cannot process", "cannot find", or "file not found."
- Make sure that you renamed *Makefile_books* to *Makefile*, and that the permissions are right: you need permission to read and write the *Makefile*. For information on setting up your Makefile, see "Copying Template Files to Your Working Directory" in Chapter 2.
- Make sure all your FrameMaker files are closed. Look for stray *.lck* files in your working directory. These are "lock" files that should exist only as long as the corresponding Frame file is open. (Note that *.mif* files cannot be generated if the corresponding *.doc* files are in use.)
- Make sure all the filenames in the *Makefile* are spelled correctly, and that there are no extra spaces after the shortname, or after any of the line continuation characters (\).
- Examine the *Makefile* to verify that lists of files extending over multiple lines have a backslash (\) at the end of every line but the last.
- Run *df* to make sure you have room in *TMPDIR* (type *echo $TMPDIR* to get the actual path). The *make* commands (particularly the figure-processing parts) can use quite a bit of space in the temporary directory specified by the *TMPDIR* environment variable; if there's no room, the commands won't work properly.
- Verify that your files (particularly figures) end in the appropriate extensions (suffixes).
- Remember to use *make _print* instead of *make print* (and *make _online* instead of *make online*).
- Try using *make -n book.full* (or use **-n** with any of the *make* commands). This tells you exactly what *make* is trying to do. Look for extra spaces, misspelled file names, or anything else that doesn't look quite right.
- Try copying a new *Makefile* into your working directory from */usr/share/Insight/templates/make/Makefile_books*, and moving the entries from your current *Makefile* to the new one.

make book Error Messages

make book errors occur during execution of the *make book* or *make book.full* commands. This section contains a list of such error messages, their causes, and how to fix them.

Note that these errors are different from the errors that occur during translation from MIF to SGML. *make book* errors appear in a shell window while a *make* command is running, and usually prevent the command from completing until you fix the problem. Translation errors are listed in error files (such as the *.err* and *.full* files) and don't usually prevent a *make* command from completing. For information on translation errors, see "MIF-to-SGML Translation Errors" in Chapter 4.

Problem: File Already Locked

```
FrameMaker reports error -1 "File already locked by login @ machine on date"
*** Error code 1
smake: 1 error
```

Here *login* is a login name (probably yours), *machine* is a machine name (probably yours), and *date* is a date (probably the current date). This means *make* can't open one (or more) of the FrameMaker files because it's locked. Close all your FrameMaker files (from within FrameMaker) and try again. If you don't have any open FrameMaker files, look through your book directory and remove any files that end in *.lck*. Try *make book* again. Another possible problem could be that someone else has logged in to your machine and moved or opened your files.

Problem: Can't Open File

```
========Building master SGML File========
cat: cannot open .sgml.tmp
*** Error code 2
---Short_title ---
cat: cannot open .sgml.tmp
***Error code 2
smake: 2 errors
```

Here, *Short_title* is the short title of your book. This error message isn't quite so informative. If you get something like this, open your Makefile and look at the line that lists the shortname of the book. Look for extra spaces or punctuation. In the example shown above, there is an extra space following the short title of the book. Make sure the short title listed in the Makefile is the same as the name of your book file.

Problem: *commonbookdefs* File Errors

```
make: file '/usr/share/Insight/templates/make/commonbookdefs' line 3: Must ba a
separator (: or ::) for rules (bu39)
make:file '/usr/share/Insight/templates/make/commonbookdefs' line 3: Syntax error
```

The dev.sw.make software subsystem is not properly installed. Re-install it. (Note that the pathname specified in the error message may vary depending on where the InSight Professional Publisher tools are located.)

```
"Makefile", line 16: Could not find
/usr/share/Insight/templates/make/commonbookdefs
"Makefile", line 51: Could not find include file '${COMMONBOOKRULES}'
Fatal errors encountered -- cannot continue
```

This error indicates that the build tools are not properly installed or mounted.

Problem: Can't Open *booklist.txt* File

```
mkbook warning: can't open file './booklist.txt'.
```

The *booklist.txt* isn't writable. Make it writable using *chmod*. The *booklist.txt* file is generated by the build tools; don't edit it by hand.

Problem: Must Have impr_rip.sw.impr

```
--- online/figurename.ai ---
ps2gif: ERROR - you must have impr_rip.sw.impr for this to work
```

The specified figure is a PostScript® file, but the tools you need to process PostScript files are not installed. Either install them (they're in the impr_rip.sw.impr software subsystem) or use only non-PostScript figures.

Problem: Cannot Map *libnsl.so* Library

```
29292://usr/frame/bin/sgi.irix.mips/fmbatch: rld: Fatal Error: cannot
map soname ’libnsl.so’ using any of the filenames
/usr/lib/libnsl.so:/lib/libnsl.so:/lib/cmplrs/cc/libnsl.so:/usr/lib \
/cmpls /cc/libnsl.so:/usr/lib/libns
--- font.mif ---
*** Error code 1
---localfigrules ---
*** Error code 208
```

This indicates that the networking software for the license manager isn’t installed. Install the eoe1.sw.svr4net software subsystem.

Viewing a Built Book

It is important to view a built book regularly during the development process to evaluate the online presentation of your material. It might be useful to review the information provided in “Guidelines for Organizing an Online Book” on page 3, and “Guidelines for Cross-Referencing in Online Books” on page 6, in completing an evaluation of an online book.

Viewing an Uninstalled Book

Use the following command to display a book with InSight during the book development process. This command assumes that you’re in your working directory.

```
% iiv -f -b .
```

In this command, the arguments to the *iiv* command (**-f -b .**) direct InSight to launch in the foreground (ignoring any other copy of InSight that might be running) and to look for a *books* directory in the current directory. You can also specify other pathnames (including relative pathnames) in an *iiv* command to open books in different directories. However, any directory that you specify must contain a local *books* directory that was created by a successful book build.

To view an uninstalled book on a different workstation, log in remotely to the other workstation, go to the appropriate directory, and launch InSight:

```
% rlogin guest@hostname
% cd bookbuild_directory
% iiv -f -b .
```

Viewing an Installed Book

In addition to viewing a book during the development process, you might also want to verify the installed image of your book. Viewing the installed image shows the final form of the online book as it will appear to the intended audience.

Note: Before using this procedure, be sure that your workstation contains the disk space that the book requires. If you are viewing Help material, you will also need additional disk space for the application that your material documents.

Use this procedure to view an installable book:

1. Determine the location of the installable book images.

   An installable image of a book is part of the software subsystem that the book documents. Find out where to install the subsystem image from. The software engineers for your product should be able to provide this information.

2. Use *swmgr* to install the book on your workstation.

   Use the standard software installation process to install your book.

   Note: You might be required to install prerequisite subsystems before installing a book. To install Help material, for instance, you must also install the associated application.

3. Start IRIS InSight to view the book.

   ```
   % iiv
   ```

   This command starts InSight and displays the newly installed book on a bookshelf.

   Note: If you are viewing Help material, start the associated application and use the application's help feature to view your material.

Chapter 4

Finding and Fixing Online Book Errors

This chapter describes the errors that can occur when FrameMaker files are converted to SGML format for online viewing. It compares the FrameMaker and SGML versions of a sample file and identifies the intermediate files that are created to store errors in the translation process. It also lists some common error messages and explains how to correct them.

This chapter contains these sections:

- "About Error Messages" discusses error messages and the basic process for finding and resolving them.
- "Error Files" lists and describes the various files generated by the online tools during the conversion to SGML.
- "About SGML" discusses SGML and provides a simple example of an SGML file.
- "Checking the .full File" tells you how to look at the *.err* files and *.full* file for error messages generated during conversion.
- "Finding and Fixing Errors" explains how to find error messages and warnings in the *.sgm* files, how to pinpoint the location of errors in the FrameMaker files, and how to interpret error messages. This section also lists some common *.sgm* file error messages and their causes.
- "Checking for Display Problems" explains how to review the document for dangling cross-references, dangling glossary items, and typical display problems.

About Error Messages

While your book is in development, even if your *make* command completes successfully (that is, doesn't terminate prematurely with a *make* or *smake* error), chances are that the content of your book still contains errors. These errors usually involve tags and structures that the translator considers invalid. The conversion tools report such problems as error messages in error files. Before you check in the final version of your book, you must examine the error messages, fix the problems that caused them, and build a clean version of your book with the *make book.full* command. (A "clean" book is one which generates no ERROR messages or link problems during conversion.)

Error Files

Depending on the specific *make* command you run, you can generate a number of different error files. (See Chapter 3, "Building Online Books," for more information on *make* commands.) The *.full* and *.err* files in this list are the most useful files for error tracking:

.err file	A *.err* file contains errors and warnings generated from the corresponding SGML file, as well as information on obsolete or unrecognized tags from the corresponding *.mif* file. You should fix all messages labeled ERROR, but you may ignore the messages labeled WARNING.
short_title.full file	The *make book.full* command generates a file called *short_title.full* that contains a complete error report for the entire book. This is the only error report that contains information on links and unresolved glossary terms.

About SGML

Each *.sgm* file contains the SGML version of its associated FrameMaker file. In SGML, each item is assigned a structural tag, similar to a FrameMaker tag, that describes that item's role in the overall document. So, in SGML, as in FrameMaker, an item (such as a paragraph or a phrase) might be tagged as part of a hanging list, or as a command, or as a chapter title. A word or phrase can have multiple (nested) labels, as happens with a command within a hanging list.

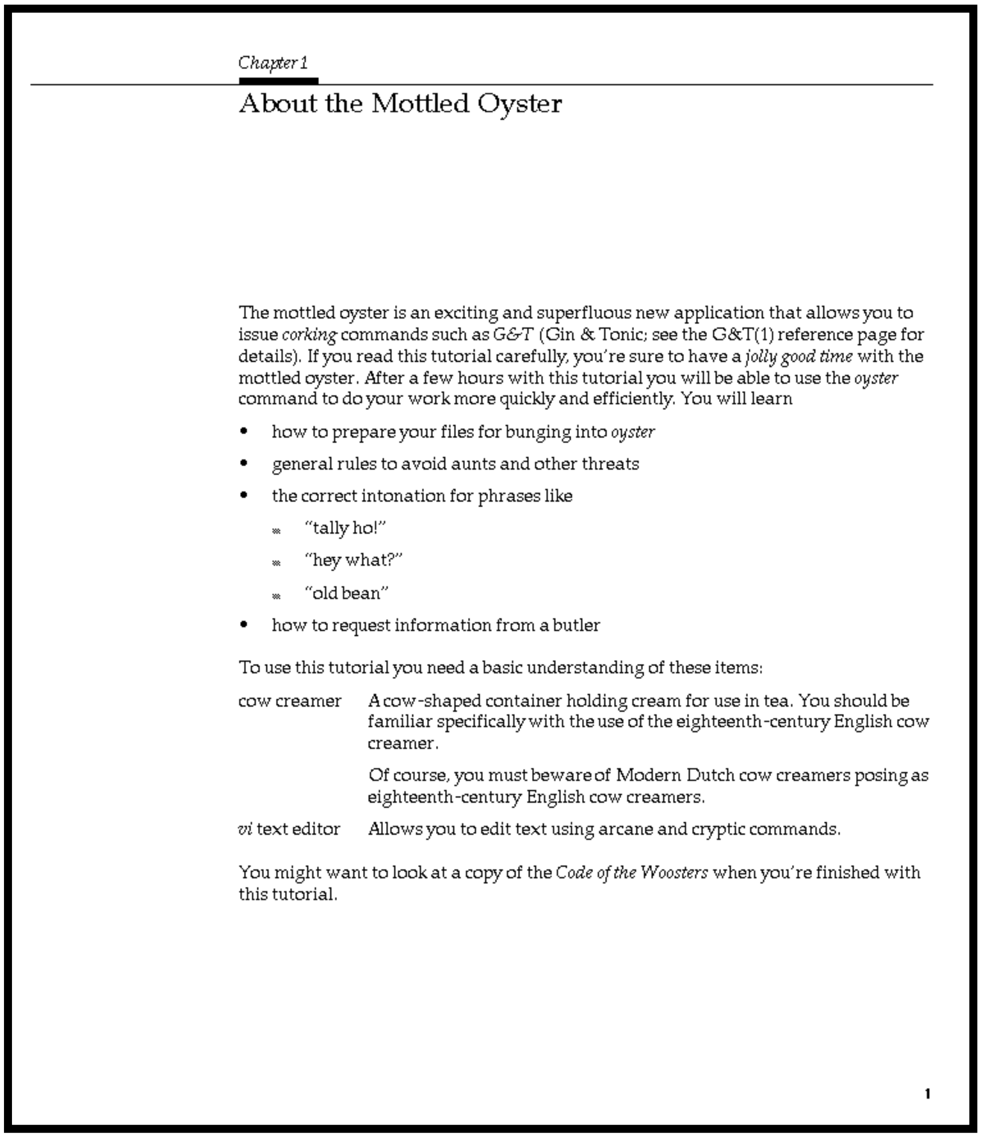
When the conversion tools find something in the FrameMaker file that they don't know how to tag, or when they find an item somewhere it's not supposed to be (such as a HangingList paragraph tag before the ChapTitle tag), they produce an error message. The example *.sgm* file in "Example SGML File" shows what error messages look like within an SGML file.

Example SGML File

This section contains an example FrameMaker document file and the corresponding example *.sgm* file generated by the *make book.full* command.

The FrameMaker File

Figure 4-1 shows an example FrameMaker file. This one-page document contains several paragraph and character tags, including an invalid structure that would generate an error message during online translation. The invalid structure is a set of square-bulleted paragraphs within a set of round-bulleted paragraphs; the BulletSquareInd tag should only be used for sub-steps within a step in a numbered list. (See the annotated chapter template file, */usr/share/Insight/templates/frame/IPTemplate/samples/IPChap.doc*, for more information on which structures are legal in a document and which aren't.)



Chapter 1

About the Mottled Oyster

The mottled oyster is an exciting and superfluous new application that allows you to issue *corking* commands such as *G&T* (Gin & Tonic; see the G&T(1) reference page for details). If you read this tutorial carefully, you're sure to have a *jolly good time* with the mottled oyster. After a few hours with this tutorial you will be able to use the *oyster* command to do your work more quickly and efficiently. You will learn

- how to prepare your files for bunging into *oyster*
- general rules to avoid aunts and other threats
- the correct intonation for phrases like
  - "tally ho!"
  - "hey what?"
  - "old bean"
- how to request information from a butler

To use this tutorial you need a basic understanding of these items:

cow creamer	A cow-shaped container holding cream for use in tea. You should be familiar specifically with the use of the eighteenth-century English cow creamer.
	Of course, you must beware of Modern Dutch cow creamers posing as eighteenth-century English cow creamers.
vi text editor	Allows you to edit text using arcane and cryptic commands.

You might want to look at a copy of the *Code of the Woosters* when you're finished with this tutorial.

1

Figure 4-1 FrameMaker Example File

The SGML File

The listing in Example 4-1 represents the *.sgm* file for the FrameMaker file shown in Figure 4-1. Look for names of character and paragraph tags to get an idea of how SGML tags work ("SGML Tag Structure" on page 41 explains the structure of SGML tags in more detail). Notice the ERROR lines in the middle of the file. This is how error messages appear within SGML files. The build tools copy all the error messages from each *.sgm* file into a corresponding *.err* file for your convenience.

Example 4-1 SGML Translation of Example FrameMaker File

```
<!-- Produced by version 3.11 (09/11/95) of SGI Frame/SGML translator
-->
<CHAPTER LBL="1"><TITLE>About the Mottled Oyster</TITLE><PARAGRAPH>The
mottled oyster is an exciting and superfluous new application that
allows you to issue
<!-- ERROR: (4) Unknown Frame tag "Slang" encountered - detected on
page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "CHAR" TAG
= "Slang" TAGCOUNT = "1" UID = "324065" TEXT = "corking"-->
<DUMMY tag="Slang">corking</DUMMY>
 commands such as <COMMAND>G&amp;T</COMMAND> (Gin &amp; Tonic; see the
G&amp;T(1) reference page for details). If you read this tutorial
carefully, you're sure to have a <GLOSSARYITEM>jolly good
time</GLOSSARYITEM> with the mottled oyster. After a few hours with
this tutorial you will be able to use the <COMMAND>oyster</COMMAND>
command to do your work more quickly and efficiently. You will
learn</PARAGRAPH>
<BULLETLIST><BULLET><PARAGRAPH>how to prepare your files for bunging
into <COMMAND>oyster</COMMAND></PARAGRAPH>
</BULLET>
<BULLET><PARAGRAPH>general rules to avoid aunts and other
threats</PARAGRAPH>
</BULLET>
<BULLET><PARAGRAPH>the correct intonation for phrases like</PARAGRAPH>
<BULLETLISTIND>
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "1" UID = "324242" TEXT = "&ldquo;tally
ho!&rdquo;"-->
<BULLETSQUAREIND><PARAGRAPH>&ldquo;tally ho!&rdquo;</PARAGRAPH>
</BULLETSQUAREIND>
```

```
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "2" UID = "324246" TEXT = "&ldquo;hey
what?&rdquo;"-->
<BULLETSQUAREIND><PARAGRAPH>&ldquo;hey what?&rdquo;</PARAGRAPH>
</BULLETSQUAREIND>

<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "3" UID = "324243" TEXT = "&ldquo;old
bean&rdquo;"-->
<BULLETSQUAREIND><PARAGRAPH>&ldquo;old bean&rdquo;</PARAGRAPH>
</BULLETSQUAREIND>
</BULLETLISTIND>
</BULLET>
<BULLET><PARAGRAPH>how to request information from a butler</PARAGRAPH>
</BULLET>
</BULLETLIST>
<PARAGRAPH>To use this tutorial you need a basic understanding of
these items:</PARAGRAPH>
<HANGLIST><HANGPAIR><HANGITEM>cow creamer</HANGITEM>
<HANGBODY><PARAGRAPH>A cow-shaped container holding cream for use in
tea. You should be familiar specifically with the use of the
eighteenth-century English cow creamer.</PARAGRAPH>
<PARAGRAPH>Of course, you must beware of Modern Dutch cow creamers
posing as eighteenth-century English cow creamers.</PARAGRAPH>
</HANGBODY>
</HANGPAIR>
<HANGPAIR><HANGITEM><COMMAND>vi</COMMAND> text editor</HANGITEM>
<HANGBODY><PARAGRAPH>Allows you to edit text using arcane and cryptic
commands.</PARAGRAPH>
</HANGBODY>
</HANGPAIR>
</HANGLIST>
<PARAGRAPH>You might want to look at a copy of the <DOCTITLE>Code of
the Woosters</DOCTITLE> when you're finished with this
tutorial.</PARAGRAPH>
</CHAPTER>
```

The usual method of finding such errors is to look at the *.err* output files after issuing a *make* command. However, you may sometimes need more information about where exactly in a chapter the error occurs. The simplest approach to finding an error in context is to use a text editor (such as *emacs*, *vi*, or *jot*) or a pager (such as *more* or *less*) to locate all occurrences within a file of the word "ERROR".

SGML Tag Structure

You can see from the example in "The SGML File" on page 39 that SGML tags come in pairs. Each tag pair contains an *opening tag* and a *closing tag*, and the opening tag applies to all of the content between it and the closing tag. Figure 4-2 shows an example of SGML opening and closing tags with tagged content, in this case simply the word "oyster."

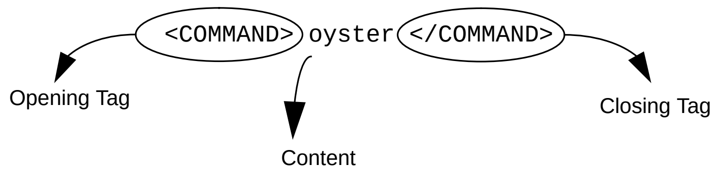


Figure 4-2 SGML Opening and Closing Tags

Checking the *.full* File

The *short_title.full* file provides a complete list of all the errors, warnings, and link problems in your book. This list consists of three separate kinds of reports: the translation error and warning report (one for each FrameMaker file), the link QA report file (a single report for the entire book), and the glossary QA report file (a single report for the entire book). Each report starts with three equals signs ("===") to set it off from the previous report.

You must fix all the errors listed in the error and warning report and the link QA report. You should fix the errors listed in the Glossary QA report, but if you don't fix them your book will still build.

This section begins with an example *.full* file, then goes on to explain how to deal with each of the three types of report files contained in a *.full* file.

An Example *.full* File

This section contains the *.full* file corresponding to the example FrameMaker file shown in Figure 4-1 (for the purposes of this example, assume that the single page of FrameMaker text is an entire book).

```
=== oyster.err Translation Error/Warning report ===
<!-- ERROR: (4) Unknown Frame tag "Slang" encountered - detected on
page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "CHAR" TAG
= "Slang" TAGCOUNT = "1" UID = "324065" TEXT = "corking"-->
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "1" UID = "324242" TEXT = "&ldquo;tally
ho!&rdquo;"-->
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "2" UID = "324246" TEXT = "&ldquo;hey
what?&rdquo;"-->
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "3" UID = "324243" TEXT = "&ldquo;old
bean&rdquo;"-->

=== Mottled_Oyster Link QA report file ===
 Unresolved References present in this book: None

 External Book References present in this book: None


=== Mottled_Oyster Glossary QA report file ===
 Unresolved Glossary terms present in this book:
------------------------------------------------
jolly good time
```

The Translation Error and Warning Report

The translation error and warning report contained in the *.full* file provides a complete list of all the translation errors and warnings in your book, grouped by chapter. Before you hand off your final book to be incorporated into a software build, you must fix all the translation errors, so that no error messages appear in the translation error and warning report. You do not need to fix warnings (marked as WARNING instead of ERROR); you can just ignore those (but you might want to take an especially close look at pages where WARNINGs occurred when you review the book in IRIS InSight to catch any display problems).

Here's the translation error and warning report for the example FrameMaker file shown in Figure 4-1.

```
=== oyster.err Translation Error/Warning report ===
<!-- ERROR: (4) Unknown Frame tag "Slang" encountered - detected on
page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "CHAR" TAG
= "Slang" TAGCOUNT = "1" UID = "324065" TEXT = "corking"-->
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "1" UID = "324242" TEXT = "&ldquo;tally
ho!&rdquo;"-->
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "2" UID = "324246" TEXT = "&ldquo;hey
what?&rdquo;"-->
<!-- ERROR: (16) The Frame tag BulletSquareInd should not occur in a
BULLET - detected on page 1 -->
<!-- ERRORLOCATION: PAGE = "1" SRC = "oyster.mif" TAGTYPE = "PARA" TAG
= "BulletSquareInd" TAGCOUNT = "3" UID = "324243" TEXT = "&ldquo;old
bean&rdquo;"-->
```

The report includes, for each error, the error message itself (the part that starts with "`<!-- ERROR:`") and the chunk of SGML that contains the error (the part that starts with "`TEXT =`"). The error message briefly describes the problem and gives a page number for the page in the FrameMaker file where the error occurred. You can see which file the error is in because you get a separate error report for each FrameMaker file.

In this example, the problem is that the writer has tried to use the BulletSquareInd paragraph tag to create a bulleted list within a bulleted list. This is an illegal structure. To fix this problem, the writer must open the FrameMaker file and re-tag the offending paragraphs with the DashInd paragraph tag. Square bullets should only be used to mark sub-steps within a numbered list; DashInd should be used for sub-bullets within a round-bulleted list. See */usr/share/Insight/templates/frame/IPTemplate/samples/IPChap.doc* for more information on which structures are legal in a document and which aren't.

The Link QA Report

The link QA report is the second report contained in the *.full* file. It lists unresolved cross references and all externally cross-referenced books (it just lists the books, it doesn't verify the links.) Here's the link QA report for the FrameMaker file shown in Figure 4-1:

```
=== Mottled_Oyster Link QA report file ===
 Unresolved References present in this book: None

 External Book References present in this book: None
```

In this example, there were no unresolved cross-references or externally cross-referenced books.

Some unresolved cross references that FrameMaker doesn't notice are trapped by the build tools. Usually such undetected cross references are caused by remnants of incomplete conversions to a new template or from third-party documentation. You can check for remnant cross references using the FrameMaker search for "Any Cross-Reference." The link QA report error indicates the approximate location of unresolved references. If you have problems finding the unresolved reference, try using *grep* to find the MIF files containing the referenced FrameMaker marker ID, then look in those files after the given marker ID. Sometimes there's an extraneous FrameMaker marker containing a cross reference or cross-reference target, and sometimes the link QA report identifies a nearby marker rather than the marker with the problem.

Glossary QA Report

Unresolved glossary terms are words that are tagged with the "GlossaryItem" character tag but have no entry in either the local or the global glossary. Here's the glossary QA report for the example FrameMaker file shown in Figure 4-1.

```
=== Mottled_Oyster Glossary QA report file ===
 Unresolved Glossary terms present in this book:
------------------------------------------------
jolly good time
```

Resolve such errors by creating an entry in the glossary or by removing the GlossaryItem tag from the word in question.

Finding and Fixing Errors

This section discusses error messages that you may encounter while building a book, and explains how to find and fix the corresponding errors in your FrameMaker files.

Types of Errors

There are three general categories of errors that can occur during the book-build process:

- *make book* errors (see “make book Error Messages” in Chapter 3 for details)
- MIF-to-SGML translation errors
- unknown errors

Error messages for *make book* errors appear in the shell window in which you enter the *make* command. Error messages for MIF-to-SGML translation errors and unknown errors appear in files generated during the build process.

Errors are listed in several different files. As explained in “Checking the .full File” on page 41, all the errors from all the files are listed in the *shortname.full* file, so that file is a good place to start looking for errors. However, once you have a general idea of what errors (and how many) are present in your book, you may prefer to work from a file with smaller scope, such as the *.err* file for a particular chapter, and to look in individual *.sgm* files to find the context a given error occurred in. “Error Files” on page 36 describes the different files generated by the conversion tools, and “Using make Commands” on page 27 explains how to use the *make* command to generate a specific type of file.

Finding the Errors in FrameMaker Files

To find a given error, open the FrameMaker file that contains the error, go to the listed page number, and look for the problem. “MIF-to-SGML Translation Errors” on page 46 lists some common error messages and their causes, so look for your error message in that list first. If your error message is not in the list, you’ll have to try and find the problem on your own.

Often the problem is easy to spot: an empty anchored frame or a mis-formatted table, for example. Sometimes, however, the problem is more difficult to find. “Unknown Errors” on page 54 lists some tips to help you find more obscure errors.

MIF-to-SGML Translation Errors

If the translator thinks it knows what the problem is, it provides an error message that looks something like this:

```
ch1.sgm: <!-- ERROR: INFORMATIVE, HELPFUL ERROR MESSAGE HERE -->
```

or

```
ch4.sgm: <!-- WARNING: STOP YOUR SLACKIN’ -->
```

This type of error is called a *known error*, because it comes with an informative error message. This section contains a sample of known error messages with explanations and suggestions for fixes. The messages are listed in numerical order, according to the number in parentheses at the beginning of each message. (The numbers don’t mean anything, and not all numbers are assigned to a unique error message; don’t worry about gaps in the numbering, or about repeated numbers.) A few error messages don’t have numbers assigned; these are listed after all the numbered messages. If your error message isn’t listed here, follow the instructions in “Unknown Errors” on page 54.

Note that error messages and warning messages generated during MIF-to-SGML translation have the same basic structure; however, error messages usually indicate that the book won’t build properly, while you can safely ignore warning messages if you prefer.

Problem: Character Tag Spans Hanging List

```
<!-- ERROR: (1) Character tag tagname spans Hanging List delimiter -->
```

Character tags shouldn't be applied to whitespace (such as tabs and hard and soft returns). This error message indicates that you've applied a character tag to both parts of a hanging list, including the tab that separates them. To fix the problem, apply Default Paragraph Font to the whitespace between the two parts of the hanging list.

Problem: Maximum Number of Table Footnotes

```
<!-- ERROR: (2) Maximum number of table footnotes exceeded -->
```

You can't have more than 26 footnotes in a single table. Remove some.

Problem: Illegal Structure

```
<!-- ERROR: (3) Illegal structure starting at 'MIFMarker' -->
```

There is a cross-reference marker in the ChapNum paragraph tag at the start of the chapter being processed. Remove that marker and change any cross-references that pointed to it so that they point to the chapter's ChapTitle paragraph instead. You can't have a cross-reference to a ChapNum paragraph.

Problem: Unknown Frame Tag

```
<!-- ERROR: (4) Unknown Frame tag tagname encountered - detected on
page pagenum -->
```

There is an unrecognized tag in a table. Here are some possible fixes:

- Search for the unrecognized tag in the document file and replace it with a recognized one.
- If the tag is not found, see if the table uses an unrecognized table format and replace it with a recognized format.
- If the above two don't work, look for straddled cells in the table, unstraddle them, and search on the revealed cells for unrecognized tags.

Problem: Hanglist Indent

```
<!-- ERROR: (10) Hanglist indent not under Hanging list -->
```

This error occurs if a FrameMaker tag (such as CodeInd5) that should occur only directly after a HangingList or HangingListInd, occurs somewhere else in a document. Don't try to use the *Ind paragraph tags to create your own indentation scheme; there are rules about what indentation can appear where. (The rules are discussed in the sample chapter file, */usr/share/Insight/templates/frame/IPTemplate/samples/IPChap.doc*)

Problem: Hanglist Not Legal

```
<!-- ERROR: (11) Hanglist not legal in this context -->
```

The HangingList or HangingListInd tag does not structurally belong in this position in the FrameMaker file. For example, you can't legally put a HangingList tag inside a Bullet list.

Problem: Illegal Help Subtopic

```
<!-- ERROR: (12) Illegal Help Subtopic content, ':' not found -->
```

Items tagged with HelpSubTopic should use this syntax:

id_in_frame_file : *title_to_appear_at_top_of_card*

This error message indicates that the colon in the middle is missing, so the item can't be parsed correctly. Add a colon to the HelpSubTopic item. (See Chapter 7, "Creating Online Help," for more information about the HelpSubTopic tag.)

Problem: No Inline Type

```
<!-- ERROR: (13) No Inline type specified -->
```

This message indicates a problem with your specification of an inline media object. The only allowed types are SGIVIDEO, SGIRGB, SGIAUDIO, and SGIINVENTOR. It's also possible that there's a problem with your syntax: an inline media object should be specified as *type*:*filename*. If you leave out the colon or introduce extraneous spaces or mistype the type or filename, you may get an error message.

Problem: Hierarchy Problem

```
<!-- ERROR: (14) Hierarchy problem for PARAGRAPH -->
```

Unable to create an SGML <PARAGRAPH> tag for a figure definition. This means the figure definition (using the Fig paragraph tag) has been applied somewhere it's not supposed to be, such as before a ChapTitle. You might also get this message if you use a Text tag in an unusual position, such as before the ChapTitle. This is a very rare error message.

Problem: Margin Figure

```
<!-- ERROR: (15) Margin figure text/graphic flows not in proper order -->
```

Margin figures are constructed from two anchored frames appearing at the beginning of a paragraph. The first anchored frame contains a graphic image, and the second (which is optional) contains a text rectangle with a FigTitleMargin paragraph tag. This error message indicates that the order of the two anchors has been reversed. Select the anchored frame for the caption (best to select the frame itself rather than the anchor; it's difficult to select a single anchor since the two appear to overlap), cut it using FrameMaker's Cut command, place the cursor after the other anchor (you can press the right-arrow key to move forward over the anchor) and paste the caption back in.

If you get a "margin figure" error about a regular (non-margin) figure, you probably made a tagging error. If you attach your anchored frame to anything other than an empty paragraph tagged with the Fig paragraph tag, the figure is treated as a margin figure.

Problem: Frame Tag

```
<!-- ERROR: (16) The Frame tag tagname2 should not occur in a tagname1 -->
```

The FrameMaker tag *tagname2* cannot legally occur within the structure denoted by FrameMaker tag *tagname1*. For example, a DashInd tag does not belong under a BulletSquareInd tag. Note that sometimes items which look fine from an aesthetic point of view nonetheless violate the SGML hierarchy and tagging rules set up by Silicon Graphics' DTD.

Problem: Hierarchy Problem

`<!-- ERROR: (17) Hierarchy problem for` *tagname* `-->`

The FrameMaker tag *tagname* cannot produce the corresponding SGML tag in the appropriate hierarchical position. The most common cause of this problem is the placement of the wrong tag at the beginning of a file. For example, a FrameMaker chapter file must begin with the tag ChapNum, followed by the tag ChapTitle.

Problem: Possible Unused Target

`<!-- WARNING: (18) Possible unused Frame Target (`*target_ID*`) -->`

The indicated FrameMaker target probably doesn't have anything pointing to it. Although this is not an error as such, it may indicate an obsolete marker or linking error. Since this is only a WARNING and not an ERROR, you can safely ignore it if you wish.

Problem: Footnote XREF Targets

```
<!-- ERROR: (20) Footnote XREF targets (avoid) -->
```

You can't make a footnote the target of a cross-reference. Change the cross-reference so that it doesn't point to the footnote.

Problem: Footnote INDEX Targets

```
<!-- ERROR: (21) Footnote INDEX targets (avoid) -->
```

You can't index a footnote. Remove the Index marker in the footnote.

Problem: Figure Logic Error

```
<!-- ERROR: (22) Figure Logic Error -->
```

This message indicates that you've used two anchored frames containing figures before the figure's title or caption. This could happen, for instance, if you place two anchored frames in the same Fig paragraph, before the FigureTitle paragraph. (Note that if one of the frames is tagged as PrintOnly conditional text, this error shouldn't occur.)

Problem: Invalid Marker 18

```
<!-- ERROR: (23) Invalid Marker 18 format (EXTREF) -->
```

This message indicates a problem in the syntax for a cross-book link. Make sure that your link specifies the book you're linking to, followed by a space, followed by the target within the book.

Problem: Illegal Tag

`<!-- ERROR: (24) Illegal tag in variable` *tagname* `-->`

You can use character tags in variables, but you can't use any character tags that you couldn't use in ordinary text.

Problem: Tag is Obsolete

`<!-- ERROR: (25) Tag '`*tagname*`' is obsolete or print-only -->`

If your document uses an outdated template, it may contain obsolete tags. Find and replace all occurrences of such tags using FrameMaker's Find/Change tool.

Problem: Marker 17

```
<!-- ERROR: (26) Marker 17 data required for Launchword -->
```

You can allow InSight readers to launch an executable by clicking on hot text in your document. To do this, label your hot text with the Launchword character tag, then place a FrameMaker Marker of type 17 just before the text. The marker itself must contain text indicating what to launch, in this format:

Launchword:*application_name*:*command-line_options*

This error message indicates that the marker is empty or the syntax is incorrect.

Problem: Index Format Syntax

```
<!-- ERROR: (27) Index Format Syntax
```

This message indicates simply that something is wrong with the syntax of an index entry. For information on legal index entries, see "Creating Index Entries" in Chapter 5.

Problem: Msg* Without Preceding Msg*1st

```
<!-- ERROR: (28) Msg without preceding Msg1st -->
```

You must use the Msg1st tag before you can use the Msg tag. The same is true for MsgMargin1st and MsgMargin, and for MsgMarginInd1st and MsgMarginInd.

Problem: MsgExpl Used Without Preceding Msg

```
<!-- ERROR: (29) MsgExpl used without preceding Msg tag -->
```

The Msg paragraph tag is part of a construct for displaying error messages that consists of one or more Msg* paragraph tags followed by one or more MsgExpl paragraph tags. This error indicates that the MsgExpl tag was used without a preceding Msg tag.

Problem: Index Format Syntax

```
<!-- ERROR: (30) Index Format Syntax: ':' before end tag 'tagname' -->
```

This message indicates that you've attempted to cross a colon or semicolon boundary, within an index entry, with a formatting tag. For information on legal index entries, see "Creating Index Entries" in Chapter 5.

Problem: Footnote Paragraph Tag

```
<!-- ERROR: (31) Footnote paragraph tag usage incorrect -->
```

You can't legally tag a paragraph in your normal text flow as Footnote or TableFootnote. If you want a footnote or table footnote in your book, use FrameMaker's Footnote tool to create it.

Problem: Glossary Tag Usage

```
<!-- ERROR: (32) Glossary tag usage in this context probably incorrect -->
```

The GlossaryEntry and GlossaryDef paragraph tags can only be used in a glossary file, not in any other file in your book. Re-tag the offending paragraphs.

Problem: Cannot Process Frame

```
<!-- ERROR: (33) Cannot process Frame on page pagenum -->
```

This probably means you have an empty anchored frame on page number *pagenum*. Import a figure into the anchored frame. See Chapter 6, “Working With Figures,” for instructions. If a figure isn’t ready yet, you can ignore this error message, view your book without figures, and put the figures in later. Another possible cause of this error message is an improperly formatted margin figure.

Problem: Cannot Process Indirect Frame

```
<!-- ERROR: (34) Cannot process Indirect Frame on page pagenum -->
```

This indicates that the tools are having a hard time processing a margin figure. When you add a new margin figure and then build your book, FrameMaker sometimes produces MIF for the new anchored frames that the translator can’t process. To fix the problem, save your document file as MIF (using FrameMaker’s Save As command), open the new MIF file, and resave it as the original *.doc* file. Note that this procedure resets the autonumbering for that file; you’ll have to use the Generate/Update command from the book file to make autonumbering work properly again.

Note: This error does not necessarily indicate that the anchored frames that form the margin figure are in the wrong order. Margin frames that are properly ordered can still have this problem.

Problem: Illegal Tag

```
<!-- ERROR: (34) Illegal tag in index marker: tagname -->
```

You can use character tags in index markers, but you can’t use any character tags that you couldn’t use in ordinary text.

Problem: Unclosed Tags

```
<!-- ERROR: INTERNAL: Unclosed Tags <TextFlow> -->
```

This error is usually caused by tagging something that should not be tagged or by not completing a sequence of more than one tag. For example, your file might contain a hanging list that includes a list item without a body or that is not followed by descriptive text.

Problem: Untranslated Function Character

`<!--ERROR: Untranslated Function Character (`*character_name*`) -->`

This message indicates that you've used a FrameMaker character that the translation tools don't recognize. Search the FrameMaker document for that character—it may be easiest to find its location by first using *grep* to search for *character_name* in the MIF file—and replace it with a standard character.

Unknown Errors

If the translator gets completely confused, it prints an error message that looks something like this:

```
ch2.sgm: <UNKNOWN.ERROR>
```

This type of error is called an *unknown error* and seldom provides a helpful error message. The report lists the page number (for the FrameMaker document) on which the error occurred.

The translator generates an "unknown error" message when the file conforms to the DTD (and is therefore structurally legal) but a problem exists anyway. Several different (but rare) conditions can cause this error. The problem is usually located in the object immediately proceeding the <UNKNOWN.ERROR> tag; examine the *.sgm* file to see exactly where the <UNKNOWN.ERROR> message occurred.

To fix an unknown error, open the FrameMaker file, go to the page number indicated in the error report, and look for anything that you think might be causing a problem. Here are some things to look for (in order of likelihood):

- Review the list in Appendix A, "Quick Reference: Rules for Using the Templates." Make sure you haven't violated any of these rules.
- Examine any unusual order of elements. For example, if you put a Figure directly after an Example—or a TextInd, Example, or Code directly after a Note—you'll get an error message.
- Read and follow the guidelines in Chapter 5, "Using FrameMaker Files and Template Features." Problems often arise from errors in figures and in tables.

Checking for Display Problems

After your book builds with no errors, bring it up in InSight and scroll through it carefully. Here are a few things to review:

- Do the links work? Do they take you to the right places? If you find a broken cross-reference, check to make sure it's correctly formatted in the FrameMaker file (see "Creating Online Links" on page 61 for instructions).
- Are the Table of Contents, List of Figures, List of Tables, and Index correctly formatted? If not, try deleting those files and replacing them with the appropriate template files from the InSight Professional Publisher template.
- Do all figures appear in the online text? If not, make sure that you set them up correctly (according to the instructions in Chapter 6, "Working With Figures"). If figures do appear, do they display correctly?

Chapter 5

Using FrameMaker Files and Template Features

This chapter provides information on how best to use FrameMaker features to produce online books, as well as pointers to related information elsewhere in the book. This chapter contains these sections:

- “Tagging FrameMaker Files” provides some important rules for using paragraph and character tags in your FrameMaker files.
- “Creating Online Links” explains the different types of links and how they appear and behave in the IRIS InSight viewer. This section also provides guidelines and instructions for creating links in your book.
- “Creating Tables” explains how to create a table correctly using the FrameMaker tools. If you don’t follow these instructions when creating a table, your table might not display properly online.

For information on creating and importing figures, see the instructions in Chapter 6, “Working With Figures.”

For instructions on creating online help and integrating it into a GUI application, see Chapter 7, “Creating Online Help.”

Tagging FrameMaker Files

Before you begin, be sure you are using the standard Silicon Graphics template, so that you can be sure that you have the most current tags. The example files that come with the templates contain information on basic tagging sequences and standard uses of the various tags.

This section offers some general tips for tagging.

Use Character Tags, Not Character Formats

If you want individual characters to look a certain way, be sure to use the supplied character tags to change their look. Do not use FrameMaker's character designer tool to simply change the format of a character. The build tools look only at tags, not at any modifications you may have made to a character's format. If you don't understand the distinction between tags and formats, refer to the FrameMaker documentation.

Use Valid Tag Sequences

The example files that come with the templates contain descriptions of the various tags and where/how they may be used. For example, don't put the ChapTitle tag before the ChapNum tag, or the FigTitle tag before the ChapTitle tag, or the Heading2 tag directly after the ChapTitle tag. The online translator expects to see only certain sequences of paragraph and character tags and produces errors for what it considers invalid sequences.

Note: Don't create your own tags, because the build tools can't translate any tags they don't know about.

Use the Correct Character Tag

Use the correct character tag for each tagged word or phrase. Prefer content-based tags (based on the kind of information being tagged) to format-based tags (based on the physical appearance of characters). For example, if you're tagging a command, use the Command character tag rather than the Italics character tag. Use the format-based character tags, such as Italics and Bold, only when the template doesn't contain a specific tag that fits your word or phrase.

Create Equations as Figures

The current build tools do not support FrameMaker equations. If you want an equation in your online book, you must use *snapshot* or other screen-capture tools to snap a picture of the equation. *snapshot* saves the equation as a *.rgb* file and you can import this *.rgb* file as if it were a figure. Don't give the equation a figure title, though, or it will show up in the list of figures as a figure. For more information about screen captures, see Appendix B, "Capturing Images for FrameMaker Documents."

Don’t Use Character Tags on Tabs, Soft Returns, or Hard Returns

In general, try to keep character tags on the specific word or phrase you’re tagging, without tagging surrounding whitespace. In particular, never apply a character tag to a tab or a soft return[1]. It’s easy to apply character tags to whitespace characters by mistake, particularly when you’re working with a hanging list (note that a tab character that has a character tag generates a warning message), so be careful.

Here are a couple of things to watch out for when tagging:

- If possible, select the word you’re tagging by double-clicking on it. This keeps you from accidentally selecting extra spaces or tabs. (Unfortunately, sometimes double-clicking does not select the entire “word” you’re tagging, because the word is a C expression or something else containing unusual characters.)
- It’s sometimes difficult to see your selection clearly in FrameMaker, so it’s hard to notice if you’ve selected spaces or tabs in addition to words. If you’ve accidentally tagged a hard return or a soft return along with the preceding word, insert a space between the return and the word before it. Tag this space with the Default Paragraph Format character tag. This procedure is necessary because hard and soft returns in FrameMaker take the character tag of the character preceding them; you can’t just tag the return itself to fix the problem.

Don’t Put Tabs in Code

Format code examples with spaces, rather than tabs. Tabs in code don’t display properly online. Conversion scripts exist to convert tabs to spaces before importing code examples into FrameMaker files.

Note: Be sure to turn FrameMaker’s Smart Spaces feature off before you import any code examples into your FrameMaker file. Otherwise, multiple spaces will be squashed down to one space and you’ll lose your formatting. (Also turn off Smart Quotes so that code samples use straight quotation marks instead of curved ones.) Be sure to turn Smart Spaces (and Smart Quotes) back on after you’re done working with code samples.

[1] You can create a soft return by holding down the **`<Alt>`** key and pressing **`<Enter>`**.

Create Valid Copyright and Trademark Symbols

Create copyright, registration, and trademark symbols according to the instructions listed here. If you use any other method for creating these symbols, you'll run into problems when you convert your files to SGML. (All of these key sequences can also be found in the "Character Sets" appendix to the FrameMaker documentation.)

- To create the copyright symbol (©), hold down <**`Ctrl`**> and press **`q`**, then release both keys, hold down <**`Shift`**>, and type a right parenthesis (**`)`**).
- To create a registered trademark symbol (®), hold down <**`Ctrl`**> and press **`q`**, then release both keys, hold down <**`Shift`**>, and type a left parenthesis (**`(`**). Apply the Superscript character tag to the registered trademark symbol after it is inserted. If you don't know whether to use a trademark symbol or a registered-trademark symbol, consult your company's legal department.
- To create a trademark symbol (™), hold down <**`Ctrl`**> and press **`q`**, then release both keys, hold down <**`Shift`**>, and press asterisk (**`*`**). This symbol is automatically superscripted; don't apply the Superscript character tag to it.

Create Valid Dashes

Create en dashes and em dashes according to the instructions listed here. (These key sequences can also be found in the "Character Sets" appendix to the FrameMaker documentation.) If you use any other method for creating these symbols, you might create problems when you convert your files to SGML.

- To create an en dash (–), hold down <**`Ctrl`**> and press **`q`**, then release both keys, hold down <**`Shift`**>, and press **`p`**.
- To create an em dash (—), hold down <**`Ctrl`**> and press **`q`**, then release both keys, hold down <**`Shift`**>, and press **`q`**.

Footnotes

Footnotes and table footnotes are supported in InSight. To create them, use the FrameMaker Footnote feature and the Footnote paragraph tag (for footnotes within text) or the TableFootnote paragraph tag (for footnotes in tables).

Creating Online Links

Links are connections between items of information, whether within the same online book or between separate online documents. When the reader clicks a linked item (whether text or an icon), either the InSight viewer scrolls to the referenced material, or the material appears in a popup window (depending on what kind of material is referenced).

Links fall into two groups based on whether they are automatically generated from the structure of the document (structural links) and require no writer effort, or explicitly created by the writer during document development (writer-generated links).

Automatic Links

A structural link is automatically created from the structure of the document during the conversion process. Lists of such links appear in the various Views in InSight. All the headings listed in the Table of Contents (TOC) view in InSight, for example, are automatically linked to the corresponding headings within the book itself. The same goes for all figures, tables and media shown in their respective View lists. The result is that a reader can click any item in a list view and the text area scrolls to the appropriate place in the online book.

Writer-Generated Links

The writer can use various FrameMaker tools to create links to text, tables, or figures in the same book (or in another book), between the text and the glossary, or between the index and the text. Each link has its own appearance and behavior.

Table 5-1 summarizes the writer-generated link types:

Table 5-1 Cross-Reference Link Types

Link Type	**How to Create**	**Appearance in InSight**	**Link Behavior**
Cross-reference to text or table within a book	Use the FrameMaker cross-reference format	Cross-reference appears in bold blue in the text	Viewer scrolls to referenced information
Cross-reference to a figure within a book	Use the FrameMaker cross-reference format	Cross-reference appears in bold blue in the text	Figure comes up in a restricted view
Cross-reference to another book	Use the FrameMaker cross-reference format	Cross-reference appears in red in text	Referenced information appears in new viewer
Glossary link	Use the FrameMaker character tag, GlossaryItem; then create a glossary entry in your book's glossary file	Item is underlined	Definition appears in a pop-up window
Index link	Use the FrameMaker index marker	Index entry appears in the Index View. Instead of a page number, location in the text appears as bold blue cross-reference to the associated heading/title	Viewer scrolls to referenced information
Reference page link	Tag an item with the RefPage character tag	Item appears as red text	Text window appears containing indicated reference page

Table 5-1 (continued) Cross-Reference Link Types

Link Type	How to Create	Appearance in InSight	Link Behavior
Inline media	Use the InlineObj and InlineTitle paragraph tags. See "InSight Inline Object Tags" in the sample chapter file (in the *samples* subdirectory of the template directory) for details of syntax	Object appears in media viewer inside InSight viewer window	Media viewer controls appropriate to media type allow user to manipulate or view media
Launch an application (including a World Wide Web browser)	Use a FrameMaker marker of type 17, as indicated in "Markers" in the sample chapter file (in the samples subdirectory of the template directory)	Either icon in margin or hot text, depending on how you set up the link	Application comes up in a new window

A Word on Cross Referencing Headings

It is recommended that for each chapter and appendix of your book, you begin with a list of the top-level sections in that part, making part of each item a cross-reference to the specified heading. This approach serves two purposes:

- It allows the online reader to click on any item in the list and have the viewer scroll to that point in the text.
- It creates cross-reference targets on all top-level headings. (Whenever you create a cross-reference, FrameMaker automatically creates a target marker at the referenced location.) These targets can be used by other writers to link to your book (see "Creating Cross-Reference Links to Other Books" on page 64).

Creating Cross-Reference Links Within a Book

Cross-reference links within a book are created using the FrameMaker Cross-Reference tool. They appear as bold blue text within the online document. When the reader clicks the link, InSight either scrolls to the text if the link is to a table or text, or opens the figure in a restricted view if the link is to a figure.

Be sure to use an appropriate cross-reference format. Note that the conversion tool automatically removes any page numbers that appear in your cross-reference text; if your book will appear in print as well as online, it's a good idea to use cross-reference formats that include page numbers, to help hard copy readers navigate through the book.

Creating Cross-Reference Links to Other Books

To create a cross-reference link to another book, either use the FrameMaker Cross-Reference tool, or use a combination of markers recognized by the translator. The former method is generally preferred. The latter is most often used when referencing a document whose source is an SGML document, rather than a FrameMaker document.

Cross-book links appear as bold red text within the online document. When the reader clicks on the link, InSight opens the appropriate book (if it's available) and either scrolls to the text if the link is to a table or text, or opens the figure in a restricted view if the link is to a figure.

Although cross-references between documents are much like cross-references between files within a book, there are two problems that complicate the situation tremendously:

- There is no way to guarantee that any two books (even those documenting the same product) will be revised simultaneously, so it's possible that another book you link to may change after your book releases, breaking your link to the other book.
- You usually don't have the same access to another book that you have to your own; in particular, you can't create a new link target in the referenced book.

The first problem is a coordination and document management issue. At present, there is no agreed-upon policy for when to create cross-book links or how to track them. If you want to create cross-book links, please take the time to evaluate the risk of having it break. If the book you want to link to is in development, be sure to talk with the author. If the book has already been released, find out when it will be revised and how extensive the changes will be.

Once you have decided to create a cross-book link, you must make sure that there’s a cross-reference target in the other book at the point you want to link to. If such a target doesn’t exist already, you need to ask the writer of the other book to create a target for you. It’s probably not a good idea to make changes to someone else’s book without asking first, especially if you use a revision control system to check documents into and out of a central archive.

Note that you cannot currently create a cross-book link to a target book for which you do not have access to any of the source files (whether FrameMaker or SGML source).

Using the Cross-Reference Utility

The following procedure explains how to create a link to someone else’s book using the FrameMaker cross-reference utility. Note that in order to create a cross-book link using this method, you need to have access to the FrameMaker document files that were used to create the book you’re linking to. You also need to know the target book’s short title—you can get that information from the target book’s *Makefile* if you have access to that.

The general idea is that you’re going to create a normal FrameMaker cross reference to the target file, but the procedure is a little bit tricky because you have to specify both the target filename and the short title of the target book. To make a cross-book link, follow these steps:

1. Follow your local procedures for obtaining a copy of the target book’s document files and Makefile. Place the copy in a directory named *short_title* (the short title of the target book), or else create a symbolic link named *short_title* that links to the directory containing the copy of the target book. The directory or symbolic link can be anywhere in your filesystem, as long as it’s called *short_title*.
2. Open up the target file in FrameMaker. When opening it, be certain to navigate to the file by way of the directory or symbolic link named *short_title*; if you don’t do this, your link won’t work.
3. Open your file that will contain the cross-book cross reference and place the cursor where you want the cross reference to go.
4. From the “Special” Menu in your document window, choose “Cross-Reference... “
5. From the Source Document menu in the resulting dialog, select the name of the file that contains the section you want to reference.
6. Choose the Source Type called X-Ref Markers.

7. Select the appropriate Reference Source.

   Note: The X-Ref Markers list is a list of FrameMaker cross-reference targets in the target file. If the target you want to reference does not appear in the list of X-Ref Markers, then it does not have an associated cross-reference target and you can't reference it unless you can persuade the owner of the document to put the target in for you. Since you don't (or shouldn't) have permission to write to someone else's book, you can't create it yourself.

8. Select a cross-reference format.
9. Click the *Insert* button.
10. Remove the target document from your system using your usual procedures.

Using Cross-Book Link Markers

Use the method described in this section to create a cross-book link to another book using the FrameMaker Marker tool. Note that you can only use this procedure with target books for which you have SGML source files. It's a somewhat complicated procedure, so if you have access to the target book's FrameMaker document files, you may prefer to use the procedure in "Using the Cross-Reference Utility" earlier in this chapter.

To create a cross-book link using markers:

1. Open the FrameMaker chapter in your book and insert the cursor at the beginning of the text where the cross-book link marker will be placed.
2. Go to the FrameMaker menu bar and select Marker from the Special menu.
3. In the Marker dialog, select Type 18 from the Marker Type pop-up menu.
4. In the Marker Text box, enter information about the cross-book link using this syntax:

   target_short_title`<Space>`*unique_id*

   where *target_short_title* is the short title of the target book, and *unique_id* is the identifying number associated with the target element in the target book. To identify the correct *unique_id*, look in the SGML file for the target book; find the target element and look for an <XREFTARGET> tag nearby. For instance, in the following example the *unique_id* for the TITLE element is 57332:

   ```
   <SECTION2 LBL="" HELPID = ""><TITLE>Supplementary
   Reading<XREFTARGET ID="57332"></TITLE><PARAGRAPH>Refer to these
   documents to supplement the information in this guide:</PARAGRAPH>
   ```

5. Press the *New Marker* button to create a new marker containing the information you've specified.
6. Move the cursor from the beginning of the text where the cross-book link will be placed to the end of the text.
7. From the Marker dialog box, select Type 19 from Marker Type. Leave the Marker Text box empty.
8. Press the *New Marker* button to create a new empty marker of type 19.

The result is a piece of your document with this structure:

`<type-18_marker>`*text*`<type-19_marker>`

where "*text*" is the text that will be marked as a link, the type-18 marker contains the name of the target book and the ID of the target element, and the type-19 marker is empty (a placeholder to indicate the end of the cross-reference text).

Note: The text between the markers cannot contain character tags.

Creating Glossary Links

Glossary links appear as underlined words in the text. When the reader clicks on the word, the definition appears in a restricted view. The link itself is either to the Silicon Graphics global glossary common to all books, or to a local glossary specific to the book. InSight searches the local glossary first (if it exists), then searches the global glossary.

To make a glossary item, tag the word using the GlossaryItem character tag. Do this only where you think it is important, not every time the word appears; otherwise your book will be peppered with underlined glossary links. In order to match, the text that you tag GlossaryItem must be spelled and punctuated identically to the text of a GlossaryTerm in your glossary.

Note: This restriction (requiring the linked reference to be identical to the entry) can cause some problems. For instance, if you have a glossary entry for the term "window," you can't link to it from the words "windows" or "windowing" in the text; and it can be difficult to re-word the text to fit the format of the glossary entry. One workaround for this problem is to create additional cross-referencing glossary entries for those words—have an entry for "windows," for instance, which simply says "see 'window,'" with the word "window" linked to the glossary entry. (Yes, you can put glossary links inside the glossary itself.)

To create a local glossary tailored specifically to your book, use the glossary template provided with the book template files. Once you've created your glossary, add its filename (usually simply *glossary.doc*) to your *Makefile* at the end of the list of chapter files. The build tools automatically create links between words tagged GlossaryItem and the corresponding entries in the glossary file.

Creating Index Entries

Index entries appear in the Index View in much the same way as they would in a hard copy index. The major difference is that instead of a page number, the location in the text appears as a bold blue cross-reference to the associated heading or title.

Use FrameMaker index markers to create index entries for the back-of-the-book index (so called to distinguish it from the full-text index that the build tools create).

Online books require certain limitations in index entries:

- Index entries can be no more than three levels deep.
- You can't put an index marker in a table footnote.
- A character format in an index entry can't cross a semicolon or colon boundary; for instance, `<Italics>entry:subentry<Default Para Font>` won't work the way you might expect it to. Instead, you'd have to use `<Italics>entry<Default Para Font>:<Italics>subentry<Default Para Font>`.
- Special characters that result in *entity* references (such as special characters for foreign languages) may not sort correctly in the online index.
- Sorting ignores these characters if they appear as the first character in an index entry: ", <, $, ., `/`, and `<space>`.

Creating Tables

Use the FrameMaker table tools to create tables. Use one of the FrameMaker table formats labelled TableXcol. If you try to create a table some other way—by using tabs, for example—it won't display properly online.

You can include footnotes in a table, if you use the FrameMaker paragraph tag TableFootnote rather than the tag Footnote.

Chapter 6

Working With Figures

The online build tools currently support six image formats for use in figures. An extension (suffix) on the file name of the imported image designates the format of the image. You can have figures in your book that don’t use the supported image formats, including composite figures composed of a combination of image formats, by means of a workaround. Except for FrameMaker line art, all images must be imported by reference into a FrameMaker document, from the *print* subdirectory of your book directory.

Original image files and the components of composite figures are stored in a directory called *orig*. During the build process, the images in *orig* are processed and placed into two additional figure-related directories that the build process generates: *print*, which contains processed images for the hard copy book; and *online*, which contains processed images for the online book. All image files in *orig* must be listed in the *Makefile* to specify how the build tools should process them.

This chapter explains how to prepare figures during the book development process. The chapter contains these sections:

- “Supported Image Formats”
- “Image File Naming Conventions”
- “The Figure Subdirectories”
- “Adding Image Files to the Makefile”
- “Processing and Importing Image Files”
- “Workaround for Unsupported Image Types”
- “Custom print Files”
- “Things to Remember About Figures”

Supported Image Formats

The book-building tools currently support the image formats listed in Table 6-1. Note in Table 6-1 that composite figures (figures containing multiple imported images in a single anchored frame) are supported for some, but not all, image formats.

Table 6-1 Image Formats Supported by the Build Tools

Image Format	Number of Files per Anchored Frame
RGB	One or more
FrameMaker line art	One or more
RGB and FrameMaker line art	One or more of either
PostScript, including Adobe Illustrator	One
GIF	One
XWD	One

Unsupported composite formats, such as figures composed of multiple Adobe Illustrator™ files or figures composed of multiple file formats (a combination of RGB and PostScript, for example), can be included in a book. However, these composites require a workaround (see “Workaround for Unsupported Image Types” on page 76).

All image files, including the elements of composites, must be stored in the *orig* directory. As with supported figures, the elements of unsupported composites must be listed in the *Makefile* and imported from the *print* directory.

Image File Naming Conventions

The *orig* directory normally contains at least one image file for each figure in your book (composite figures may use more than one file). The filename for each image file in the *orig* directory must end with a standard extension (or suffix) that designates the format of the image.

Table 6-2 shows the standard extension for each image format that is currently supported. Notice that Table 6-2 does not include an extension for FrameMaker line art, since line art figures are not imported files:

Table 6-2 Image File Naming Conventions

File Format	Extension
Adobe Illustrator PostScript	.ai
color RGB	.rgb
black and white RGB	.bw[a]
Non-Illustrator PostScript	.ps or .PS
GIF	.gif
XWD	.xwd

a. These images are screen snapshots (or other RGB files) that have been converted to black and white with the *tobw* utility.

The Figure Subdirectories

The local working directory for a book contains three subdirectories for use with figures:

orig	Place all original image files and the components of figure composites in this directory. All RGB snaps, Adobe Illustrator files, and other original pieces of art belong in this directory. **Hint:** For books that are printed in color, store the color versions for the printed copy in *orig*; import a black and white FPO (For Position Only) version into the FrameMaker document from the *print* directory.
print	This directory, once you've populated it by using the *make _print* command, contains all image files that will be used in the printed version of your book. Any image files that are imported into anchored frames must be imported from the *print* directory. Normally there's no reason to alter *print* or its contents through any means other than the *make _print* command.
online	This directory, once you've populated it by using the *make _online* command, contains all image files that will be used in the online version of your book. Normally there's no reason to alter *online* or its contents through any means other than the *make _online* command.

Adding Image Files to the Makefile

The *Makefile* controls the conversion of images into the appropriate format for the online and printed versions of your book. During the build process, the build tools automatically convert imported file references into the appropriate references to the *online* directory.

The *Makefile* contains variables that specify image formats and conversion processes. Use the following instructions when listing image files in *Makefile*. List each image file in only one place in the *Makefile*. You might also find it useful to look ahead to Table 6-3 when deciding where to list image files in the *Makefile*.

PRINT_BW =	Use this variable to specify original RGB color images that will be imported as black and white images for the printed book. The files that you list on the PRINT_BW line appear as eight-bit color GIF images in the online book. All image files listed on the PRINT_BW line must be stored in the *orig* directory with a *.rgb* extension. **Note:** If you want images to appear in 24-bit color online, do not list them on this line; list 24-bit color images on the RGB variable line instead. Also note that files listed in the PRINT_BW variable can't be part of a composite figure.
PRINT_COLOR =	Use this variable to specify RGB color images that will appear in color in both the printed version and the online version of your book. The files that you list on the PRINT_COLOR line appear as color GIF images in the online book. All image files listed on the PRINT_COLOR line must be stored in the *orig* directory with a *.rgb* extension. **Note:** If you want images to appear in 24-bit color online, do not list them on this line; list 24-bit color images on the RGB variable line instead.
PostScript =	Use this variable to specify all PostScript images (including Adobe Illustrator files) in your book. These images appear in color in FrameMaker and in the online version of your book; they appear in color in the printed version if printed on a color printer. Image files listed on the PostScript line must be stored in the *orig* directory with either a *.ps* or a *.ai* extension.
GIF=	Use this variable to specify all GIF images in your book. All image files listed on the GIF line must be stored in the *orig* directory with a *.gif* extension. Color GIF files appear in color in the online version of your book; black and white GIF files appear in black and white in the online version of your book. Be sure to use GIF 87a format instead of GIF 89a; you can use the *file* command to find out what format a given file is in. Some version of the *togif* utility translate to GIF 89a; be sure to use the version of *togif* that comes with InSight Professional Publisher. **Note:** FrameMaker will not display or print GIF images.

RGB=	Use this variable to specify images that should be processed for 24-bit color in the online version of a book. All image files on the RGB line must be stored in *orig* with an *.rgb* extension. Note that 24-bit images may appear dithered on some monitors, and InSight doesn't display in 24 bits by default; many or most customers won't see your images in 24 bits even if you include 24-bit images with your book. **Note:** RGB files appear in color in the FrameMaker application; this contributes to printing problems due to file size. Also note that using this variable greatly increases the size of an online book, because RGB-format images are much larger than GIF-format images. Unless you're certain that you need 24-bit images in your book, it's best not to use this variable.
XWD=	Use this variable to specify all XWD images in your book. All image files listed on the XWD line must be stored in the *orig* directory with a *.xwd* extension. Color XWD files will appear in color in both the printed and the online versions of your book; black and white XWD files will appear in black and white in both the printed and the online versions of your book.
BW=	Specifies RGB-format images that have already been converted to black-and-white format. This type of image is sometimes present in documentation created outside of Silicon Graphics, but the BW line is almost never needed for images generated with Silicon Graphic tools

Processing and Importing Image Files

All image files used in a book must be imported into a FrameMaker document from the *print* directory for a book to build successfully.

This procedure explains how to generate and populate the *print* and *online* directories and how to import images into FrameMaker files from *print*:

1. If you've already copied the *Makefile* template to your local working directory, skip to step 2. If you haven't already done this, do it now:

   ```
   % cp /usr/share/Insight/templates/make/Makefile_books Makefile
   ```

   The *Makefile* file specifies information about your book to the build tools. It contains variables that you must set and instructions for how to set them. See "Editing the Makefile" on page 20 for additional information.

2. Change write permissions on your copy of the *Makefile* file.

   ```
   % chmod 664 Makefile
   ```

3. Edit the *Makefile* template.

   ```
   % vi Makefile
   ```

   (You can use a different text editor, such as *emacs* or *jot*, if you prefer. If you use FrameMaker to edit this file, be sure you save as text when you're done editing.) Set the image variables in the *Makefile* that apply to your imported image files, such as PostScript and PRINT_BW (see "Processing and Importing Image Files" on page 74). At this point you can also set the other variables, such as TITLE and BOOK_FILES, if you wish, but only the image variables are required to generate the *print* directory.

4. Execute the *make _print* command:

   ```
   % make _print
   ```

   The *make _print* command creates a *print* directory and copies the files from *orig* to *print*, converting any formats as needed.

5. Import your image files. See Table 6-3 for filename extensions.

   Open each FrameMaker file that should contain imported images. The method that you use to import the image files depends on the current importing settings in the document files:

   - If your FrameMaker files do not specify *orig* or *print* as the location of imported image files, the FrameMaker application warns that it can't import the files. To continue, specify the *print* directory as the new location of your image files.
   - If your FrameMaker files specify *orig* as the location of the imported image files, the FrameMaker application does not issue a warning; it opens the files, importing images from the *orig* directory. In this case, you must re-import all of the image files, specifying the *print* directory as the image file location. To re-import image files, temporarily rename the *orig* directory, then reopen the document file to generate a FrameMaker error. Specify the *print* directory version of the image files in the error dialog box. When the re-importing is complete, rename the temporary directory to *orig*.

Workaround for Unsupported Image Types

If your book contains images with unsupported formats, use this workaround to prepare the images for processing by the book-building tools. This procedure assumes that you already have an anchored frame in your FrameMaker file that contains an unsupported image format:

1. Snap the composite picture as an RGB file.
2. Copy the new RGB file to the *orig* directory.
3. Add the new image file to the PRINT_BW line in *Makefile*.
4. Execute the *make _print* command to add the file to the *print* directory.
5. Create a second frame in your FrameMaker document:

   Using the same Fig paragraph tag, insert a second anchored frame below the location of the original frame containing the unsupported image.

   Note: Be sure that you have a single figure title for the anchored frame pair. The title should appear below the second frame.
6. Import the *.bw* image into the new frame from the *print* directory.
7. Tag the frame containing the unsupported version of the figure with the PrintOnly conditional text tag.

   The PrintOnly tag appears on the Conditional Text menu.
8. Tag the frame containing the *.bw* version of the figure with the OnlineOnly conditional text tag.

   The OnlineOnly tag appears on the Conditional Text menu.

Note: When you print the document, be sure to turn off the OnlineOnly condition using the Show/Hide selection on the Conditional Text menu. During the book-building process, the PrintOnly condition (among others) is turned off automatically.

Custom *print* Files

Occasionally, you may find that the quality of a printed version of an image created by automatic conversion is unacceptable. This may happen, for instance, if your images have a black background that shows up fine online, but overwhelms any reversed-out line art in print. To produce a custom version for print:

1. Perform a screen snap on the original version and put it in the *orig* directory.
2. Edit the *Makefile* as usual and execute *make _print*.
3. Modify a copy of the original file and put it in the *print* directory.

   This modification might involve either of these tasks:

   ■ Running an image processing tool on the file.

   ■ Changing the X resource values for an application to increase contrast, and re-snapping a window with contrasts acceptable for a black and white book.
4. Import the *print* version of the figure as usual.

Note: This process works because *make* does not reprocess an image file for the *print* directory if the file in *print* is more recent than its counterpart in the *orig* directory. For this reason, you *must* recreate a custom print version whenever you update the original. If you do not recreate the custom version, *make* will replace the custom version with the newer, automatically generated version that does not contain your changes.

Things to Remember About Figures

For your book to build properly (without errors) you must follow all the rules in this list:

- Import all images *by reference* from the *print* directory. When you build your book, the online tools automatically grab the corresponding images from the *online* directory for display in InSight.
- Import each image into an anchored frame. For figures that appear within normal page boundaries, the frame must be anchored to an empty paragraph (blank line) that is tagged with the Fig' paragraph tag.
- In order for a figure title to appear in the List of Figures in InSight, the Fig paragraph tag has to be followed by FigTitle paragraph tag.

- Margin figures are composed of two anchored frames, one for the image and one for the caption. The anchored frame for the caption cannot appear before the anchored frame for the figure. If it does, the *.err* file for the chapter containing the margin figure will contain an error message (see Chapter 4, “Finding and Fixing Online Book Errors”).

  Margin figures occasionally cause translator errors, producing the error message “Cannot process indirect frame on page...” To correct this error, save the FrameMaker file containing the margin figure in MIF format, then open the *.mif* file and save it as a *.doc* file.

Table 6-3 lists the various file formats that can occur in the *orig* directory. For each format, it shows the format of the file in a FrameMaker document, which produces the hard copy version of the figure, and the format of the file in IRIS InSight, which produces the figure in online books. Table 6-3 also shows the file extension in the *orig* and *print* directories and the *Makefile* variable where the file should be listed.

Table 6-3 Image Formats for Hard and Soft Copy

File Format in *orig*	**FrameMaker File Format**	**IRIS InSight File Format**	**Suffix in *orig***	***Makefile* Variable**	**Suffix in *print***
RGB	black/white RGB	GIF	.rgb	PRINT_BW	.bw
black/white RGB	black/white RGB	GIF	.bw	BW	.bw
RGB	RGB	GIF	.rgb	PRINT_COLOR	.clr
RGB	RGB	24-bit RGB	.rgb	RGB	.rgb
AI PostScript	AI PostScript	GIF	.ai	PostScript	.ai
PostScript (non-AI)	PostScript (non-AI)	GIF	.ps	PostScript	.ps
GIF	GIF (not viewable)	GIF	.gif	GIF	*.gif*
XWD	XWD	GIF	.xwd	XWD	.xwd

For all image formats except black/white RGB, color palette information remains unchanged during image file processing. This means that a black and white original produces a black and white figure, and a color original produces a color figure.

For the black/white RGB format, color palette information is reduced to black and white during processing. This means that both black and white and color originals produce black and white figures.

Chapter 7

Creating Online Help

This chapter explains how to use FrameMaker and the online book building process to create online help for an application. The same source file is used for online books and SGI Help; when you're writing a book, you can mark sections of it to be used by the help system.

The SGI Help system relies upon several main components:

- A FrameMaker file containing special labels that mark the sections you want to use as help topics. When you build the online book, these labels become a part of the SGML file.
- A helpmap file that provides a mapping between the help topics in the online book and the Help menu in the application.
- A help server that handles communication between the application and the helpmap file.

Tagging FrameMaker Files

Any full section of a book—from one heading to the next heading of the same or higher level—can be used as a help topic. To mark it as such, place an empty paragraph above the chosen section's heading (any paragraph tagged Heading1, Heading2, or Heading3); then, in this new paragraph, enter a unique one-word label to identify the section. Tag the label with the HelpTopic paragraph tag; then apply the OnlineOnly conditional text tag to it. Here's a more detailed set of instructions:

1. Place the cursor on a blank line above the heading of the section you want to use as a help topic.

   If you place it above a Heading1, the help topic will include any Heading2 and Heading3 subsections that the marked section contains. If you place it above a Heading2, the help topic will include any Heading3 subsections that the marked section contains.
2. Type a unique label for that topic. Later, you'll be entering this label into the helpmap file, thereby tying the help topic to the application.

   Do not put spaces or hyphens in the label. And make sure that you choose a name that another writer isn't likely to use. For example, you might want to include the application name, or an abbreviation of the name, followed by some other text.
3. Tag the paragraph with the HelpTopic paragraph tag.
4. Choose Conditional Text from the Special menu.
5. Select the entire paragraph containing the new label; then choose OnlineOnly from the Conditional Text dialog. Click the left-pointing arrow button in the dialog to move "OnlineOnly" into the "In:" column; then click the Apply button. Close the Conditional Text dialog.

Parts of sections of a book can also be used as help topics. Note that such pieces must be complete SGML structural units, such as paragraphs or tables or figures; you can't use part of a paragraph, or part of a table, or part of a figure, as a help topic. In fact, if you use a table you must also use the entire paragraph that the table anchor appears in.

To create a help topic from a piece of a section, follow these steps:

1. Place the cursor on a blank line above the first paragraph of the subsection you want to use as a help topic.
2. Type a unique label for that topic, then a colon, then a title for the topic. The label follows the same rules as for HelpTopic, above; the title is necessary since there's no section title to use for the help card title. The label and title are eventually put into the helpmap file by the book-building tools, thereby tying the help topic to the application.

   Remember not to put spaces or hyphens in the label, and to choose a name that another writer isn't likely to use.

3. Tag the paragraph with the HelpSubTopic paragraph tag.
4. Choose Conditional Text from the Special menu.
5. Select the entire paragraph containing the new label; then choose OnlineOnly from the Conditional Text dialog. Click the left-pointing arrow button in the dialog to move "OnlineOnly" into the "In:" column; then click the Apply button. Leave the Conditional Text dialog open.
6. Now select all of the text that you wish to appear in the help topic. Choose HelpSubTopic from the Conditional Text dialog. Click the left-pointing arrow button in the dialog to move "HelpSubTopic" into the "In:" column; then click the Apply button. Close the Conditional Text dialog.

Help Buttons Versus Help Menus and Context Sensitive Help

The number of help labels you need to include depends on the complexity of the application and the method by which people will access help. There are two common approaches to online help: a *Help* button or a Help menu.

- If the application has a *Help* button, add a label to your book above the heading of the section that you want to appear when the user clicks the *Help* button. If the application allows you to open several windows, each with its own *Help* button and help card, put a different label above each topic that should come up in response to a *Help* button. You have to tell the application developer what these labels are. The application passes this info to the help server when the user clicks a *Help* button.
- If the application has a Help menu, add a label to your book above each section that should appear on the Help menu or should appear when a user requests context sensitive help for a particular area of the window. If the application is based on the ViewKit interface toolkit, you do not have to pass this information on to the developer. If it is not ViewKit-based, you need to provide the developer with a list of the topics for the menu and all of the associated labels.

Note that you tag your FrameMaker files in the same way regardless of whether you're creating context-sensitive help or menu-based help. The difference between the two is indicated in the helpmap file.

The Helpmap File

The helpmap file acts as the glue that binds the help topics to the application. In it, you list the topics that you've designated as "help" in your FrameMaker files, and you specify when the topic should appear—when a user chooses a particular item from the Help menu, for example, or when the user clicks in a certain place in the window.

The name and location of the helpmap file are important. If the application is based on ViewKit, name the file *AppName.helpmap*. The AppName should be the same as the name of the application's defaults file in */usr/lib/X11/app-defaults* (usually a capitalized form of the name of the executable). If the application isn't ViewKit-based, the writer and the developer need to agree on a name to use for the helpmap file.

Place the file in your working directory, in a subdirectory called *help*. For example, suppose your book directory is */usr/people/debbie/books/IRISEssentials*. The helpmap file should be placed in */usr/people/debbie/books/IRISEssentials/help*.

An Explanation of the Helpmap File

Each line of the helpmap file contains six fields. The fields are separated by semicolons; all fields should be on the same line of text (so don't press Return when you reach the end of the line, even if that makes the line wrap in the middle of a word). Here's the format to use, followed by explanations of the fields:

topic_num ; *book_name* ; *title_str* ; *level_id* ; *help_key* ; *widget_string*

topic_num	A number that indicates what type of help topic it is:
	0: Not on a help menu—either context sensitive or tied to a Help button
	1: Overview topic on the Help menu
	2: Task-oriented entry that shows up on the Help menu
	3: Keys and Shortcuts entry on the Help menu
book_name	The short name of the book that contains the help topic.
title_str	The name of the topic. This determines what appears on the Help menu and in the index. This name can be different from the label that appears in the FrameMaker file. (Be sure to fill in this field even if the topic is one that doesn't appear on a Help menu.)

level_id	A number indicating whether the item is a top-level menu entry or on a rollover menu. Use 0 if the item is a top-level entry, or isn't on a menu (as with button or context-sensitive help).
help_key	The label in your FrameMaker file that is associated with this help topic.
widget_string	Specific information that identifies the application, or portion of the application to which the help topic belongs.

For the Overview menu entry, use **`AppName.windowName.overview`**

If the topic is tied to a help button, use **`AppName.windowname`**

For a list of Keys & Shortcuts, use **`AppName.windowName.keys`**

If the topic is tied to a portion of the window (as is true for context sensitive help), you need the fully qualified widget string. You can include several different widget strings if you want to use the same topic for different regions of the window. See “Creating Context Sensitive Help” and “Setting Up the Fallback Mechanism for Context Sensitive Help.”

A Sample Helpmap File

This section contains a sample helpmap file.

```
1;IRISEssentials;The Desks Overview Window;0;ov_deskoverview;Overview.DesksOvervi
ew.overview
2;IRISEssentials;Creating a Desk;0;ov_createdesk;Overview.DesksOverview
2;IRISEssentials;Switching Between Desks;0;ov_switchdesk;Overview.DesksOverview
2;IRISEssentials;Moving Windows Between Desks;0;ov_moveitems;Overview.DesksOvervi
ew
2;IRISEssentials;Copying Windows Between Desks;0;ov_copyitem;Overview.DesksOvervi
ew
2;IRISEssentials;Listing All the Windows;0;ov_listall;Overview.DesksOverview
2;IRISEssentials;Renaming a Desk;0;ov_renamedesk;Overview.DesksOverview
2;IRISEssentials;Deleting a Desk;0;ov_deletedesk;Overview.DesksOverview
2;IRISEssentials;Placing a Window in All Desks;0;ov_globaldesk;Overview.DesksOver
view
2;IRISEssentials;Manipulating Windows;0;ov_manipulatewin;Overview.DesksOverview

2;IRISEssentials;Changing the Overview Display;0;ov_changeviews;Overview.DesksOve
rview
2;IRISEssentials;Changing the Order of Desks;1;ov2_arrangedesks;Overview.DesksOve
rview
```

```
2;IRISEssentials;Displaying the Names of Items in a Desk;1;ov2_displaynames;Overv
iew.DesksOverview
2;IRISEssentials;Hints for Resizing;1;ov2_resizinghints;Overview.DesksOverview
2;IRISEssentials;Viewing Desks as Snapshots or Buttons;1;ov2_displaybutton;Overvi
ew.DesksOverview
2;IRISEssentials;Resizing Snapshots or Buttons;1;ov2_resizebutton;Overview.DesksO
verview

0;IRISEssentials;The Desk Display Area;0;ovcon_display;Overview.DesksOverview.Mai
nView.Frame.viewport.Bboard
0;IRISEssentials;The Global Desk;0;ovcon_global;Overview.DesksOverview.MainView.g
lobalDesk
0;IRISEssentials;About Desks Overview Menus;0;ov_aboutmenus;Overview.DesksOvervie
w.menuBar
0;IRISEssentials;The Overview Menu;0;ovcon_overmenu;Overview.DesksOverview.menuBa
r.Overview
0;IRISEssentials;The Desks Menu;0;ovcon_deskmenu;Overview.DesksOverview.menuBar.D
esks
0;IRISEssentials;The Window Menu;0;ovcon_winmenu;Overview.DesksOverview.menuBar.W
indows
0;IRISEssentials;The List All Window;0;ovcon_listwindow;Overview.WindowList_popup
3;IRISEssentials;Keys and Shortcuts;0;ov_keyshortcuts;Overview.DesksOverview.keys
```

Creating A Rollover Menu

You can create a rollover menu for a help topic. For example, suppose a Help menu includes the topic Changing the Overview Display and you want to include information on various ways in which you can change the display. You set this up through the helpmap file by using a different *level_id* for the rollover menu items. Note the *level_id* of 0 in the first entry and the *level_id* of 1 in subsequent entries:

```
2;IRISEssentials;Changing the Overview Display;0;ov_changeviews;Overview.DesksOve
rview
2;IRISEssentials;Changing the Order of Desks;1;ov2_arrangedesks;Overview.DesksOve
rview
2;IRISEssentials;Displaying the Names of Items in a Desk;1;ov2_displaynames;Overv
iew.DesksOverview
2;IRISEssentials;Hints for Resizing;1;ov2_resizinghints;Overview.DesksOverview
2;IRISEssentials;Viewing Desks as Snapshots or Buttons;1;ov2_displaybutton;Overvi
ew.DesksOverview
2;IRISEssentials;Resizing Snapshots or Buttons;1;ov2_resizebutton;Overview.DesksO
verview
```

Creating Context Sensitive Help

If an application has a Help menu, you can create context sensitive help. Context sensitive help allows you to provide information about specific areas of controls on a window. For example, the InPerson™ desktop conferencing software has an area into which you can drop icons that represent the people you want to call. This is called a drop pocket. Context sensitive help might explain what the drop pocket is and how to use it.

To create context sensitive help, you must identify the area of the window to which you want to link the help topic. You do so by giving the full name of the widget in that area of the window. (A widget, in X parlance, is an element of a user interface—a button, scroll bar, text area, or other interface item.) The widget's full name includes the name of each widget in the hierarchy that contains it, from the entire window down to the specific area in question. For example, here is the widget name for the drop pocket on the InPerson window:

```
InPerson.callWindow.Form.GroupPanel.scroll.iconForm.Xdraw
```

If the application is ViewKit-based, and has been linked to the help server, you can get the widget name relatively easily:

1. Open a shell window.
2. Kill the help server and restart it in debugging mode by typing:

   ```
   % /etc/killall sgihelp
   % sgihelp -debug
   ```

3. Choose "Click for Help" from the Help menu, then click on the region for which you want to provide context sensitive help. The widget string appears in the shell window. Below is an example of what you might see in the shell window:

```
% sgihelp -debug
REQUEST =client="InPerson" command="view" book="" keyvalue="callWindow.Form.Group
Panel.scroll.iconForm.Xdraw" separator="." user_data="" title="InPerson" s
ubsys="InPerson.books.InPerson_AG and InPerson.books.InPerson_UG"
INFO FOR HELPMAP =InPerson.callWindow.Form.GroupPanel.scroll.iconForm.X
draw
```

The INFO FOR HELPMAP field contains the widget name to use in your helpmap.

Setting Up the Fallback Mechanism for Context Sensitive Help

You might not provide context sensitive help for every button or set of controls on a window. However, you need to make sure that something reasonable appears, no matter where a user clicks. For example, if a user clicks on an area for which there isn't any specific context sensitive help, you might display the Overview help card.

Here's how it works. When a user requests context sensitive help, the help server looks to see if that widget name appears in the helpmap file. If not, it drops the last item in the widget name and tries again to find a match. It continues doing this until it finds a match. For example, suppose a user is reading a book in InSight and requests context sensitive help on the *Go Back* button. The widget string that gets reported is:

```
Insight.InsightBook.IvForm.ivPane2.commandArea.DocCmdAreaRC.goBack
```

Suppose help is not available for the *Go Back* button. The help server continues searching and finds a match with this portion of the widget string:

```
Insight.InsightBook.InsightBook.IvForm.ivPane2
```

Here's a helpmap entry which causes the Overview card to display under those circumstances:

```
1;InsightHelp;Overview;0;inshelp_overview;Insight.overview;Insight.Insi
ghtBook.InsightBook.IvForm.ivPane2
```

Using the World Wide Web from Your Help Menu

You can include a link to one or more World Wide Web pages in your help menu if you so desire. This feature can be useful, for instance, if you want to provide release notes, frequently asked questions lists, and other such time-sensitive information by way of the Web.

Including a URL in a help menu is much like placing any other item in that menu; it merely requires a line in the helpmap. (Note, though, that not all customers have access to the Web, so you shouldn't put any critical information on a Web page.) The format is identical to other lines in the helpmap file. For *topic_num* and *level_id*, use 0 even though the item will appear on the Help menu. For *book_name*, give the complete URL. For *title_str*, give the entry you want to appear in the Help menu; you should also indicate somehow to the user that this item launches a Web browser. (You can, for instance, use the word "(Web)" in parentheses at the end of the *title_str*.) Leave the *help_key* field blank;

there is no help topic in the book that this item corresponds to. And finally, the *widget_string* should be the name of the application's top-level widget. For example:

```
0;HREF=http://dyne.yoyo.com/dougom/Tutorial.html;examples (Web):;0; ;Cvpav.cvpav
```

That's all there is to it; when the user chooses this item from the Help menu, a Web browser is launched and the URL is opened.

Building and Testing the Online Help

This section outlines the process of building and testing online help. Once you've checked in the files and the application has been built, you can install the product and test the help. This section explains how to test the help before the product is rebuilt. It does not provide details on the *make book* command or the setup that's required to build an online book. See Chapter 3, "Building Online Books," for that information.

To test an online book:

1. Build the book:

   ```
   % make book
   ```

2. Change directories into the subdirectory where the book has been created:

   ```
   % cd books
   ```

   You should be in a directory named something like */usr/people/debbie/books/IRISEssentials/books*

3. List the contents of the directory:

   ```
   % ls
   ```

   You should see a directory named after the book. For instance, in this example, you see a directory named *IRISEssentials*. This is the directory that contains the online book and help.

4. Copy this online book directory (*IRISEssentials* in the example) into */usr/share/Insight/library/*companyname*_bookshelves/*companyname*/books*, where *companyname* is the name or abbreviation for your company (as determined in "Editing the Makefile" in Chapter 3).

5. Edit the associated *booklist.txt* file. For example, if you copied the book files into the directory */usr/share/Insight/library/Yoyo_bookshelves/Yoyo/books*, edit the *booklist.txt* file in */usr/share/Insight/library/Yoyo_bookshelves/Yoyo*.

   Make sure that this file includes an entry for the book whose files you copied. If it doesn't, copy the information from the *booklist.txt* file in your working directory.

6. Copy the helpmap file into the directory */usr/share/help*.
7. If you're already running the help system, kill it by typing:

   ```
   % /etc/killall sgihelp
   ```

   It will automatically restart when you next request help.

8. If you're running the application whose online help you want to test, quit out of the application and re-launch it.

   Now you're ready to try using the online help that you've created.

Telling Users Which Subsystems to Install

If a user hasn't installed the correct online book, an error message will be seen when online help is accessed. To make this error message as useful as possible, you need to ask the application engineers to include two extra lines in the application's defaults file. Here's an example of what appears in the *app-defaults* file for the Desks Overview tool:

```
*helpSubSys: desktop_eoe.books.IRISEssentials
*helpTitle: Desks Overview
```

If a user doesn't have the book installed, a message comes up saying to install the desktop_eoe.books.IRISEssentials subsystem.

Troubleshooting

When you begin testing the online help, you may see one of several problems:

- The help server says that it's unable to find help on a particular topic. See "No Help Available."
- The wrong section appears when you choose a help topic. See "The Wrong Help Topic Appears."

No Help Available

If you get a message saying something like “Sorry, no help available on that topic”:

- Check to make sure the book is installed. Look in all the relevant subdirectories of your company’s bookshelves directory under */usr/share/Insight/library*, including any Help subdirectories.
- Check the generated file *booklist.txt* in the relevant subdirectory of your company’s bookshelves directory under */usr/share/Insight/library*. Make sure it has an entry for the book, and that the book’s title is spelled correctly.
- Make sure the helpmap file is in */usr/share/help*.

The Wrong Help Topic Appears

If the wrong section of the book appears when you click the *Help* button or choose a topic, you probably have a problem in the FrameMaker file, or a mismatch between the application, the FrameMaker file, and the helpmap file.

In the FrameMaker file:

- Make sure that the paragraph with the help label is tagged HelpTopic.
- Make sure that the heading below the HelpTopic is not accidentally tagged with the HelpTopic paragraph tag.
- Make sure the text in the HelpTopic paragraph is OnlineOnly conditional text.

To check for a mismatch between the FrameMaker file, the helpmap file, and the application:

1. Stop the help server and restart it by typing the following:

```
% /etc/killall sgihelp
% sgihelp -debug
```

2. Click on a *Help* button or choose a topic from a Help menu. A bunch of text appears in the shell window. Here are two examples:

```
REQUEST =client="Overview" command="view" book="IRISEssentials"
keyvalue="ov_createdesk" separator="." user_data="" title="Desks Overview"
subsys="desktop_eoe.books.IRISEssentials"
```

```
REQUEST =client="Overview" command="view" book=""
keyvalue="DesksOverview.menuBar.Desks" separator="." user_data="" title="Desks
Overview" subsys="desktop_eoe.books.IRISEssentials"
INFO FOR HELPMAP =Overview.DesksOverview.menuBar.Desks
```

client	The name of the application. Make sure the name of the helpmap file and the last entry in the file match this.
book	The name of the book that contains the help content. Make sure the spelling in the helpmap file and *booklist.txt* file match this. This field will be empty if you have chosen “Click for Help.”
keyvalue	One of two pieces of information: either the HelpTopic tag in your FrameMaker file and helpmap file or a portion of the widget string for the context sensitive help. If it’s the label or tag you’ve used to mark the help topic, check the helpmap file to make sure the spelling, capitalization, and punctuation match.
INFO FOR HELPMAP	The widget string to be used for context sensitive help. Make sure this matches the widget string you’ve placed in the helpmap file.

Missing Items on the Help Menu

Check the helpmap file if items are missing from the Help menu. Check the following:

- Make sure the line in the helpmap file begins with the number 2. This specifies that it is a topic that should appear on the menu.
- Make sure you haven’t left out a semicolon.

Quick Reference: Rules for Using the Templates

This list was compiled by writers with online experience at Silicon Graphics. If you break any rule on this list, you'll find errors when you try to put your book online. Even if your book isn't going online immediately, follow these instructions wherever possible:

- Name all your FrameMaker files so that they end in the suffix *.doc* (*01.begin.doc*, *02.explore.doc*, *A.reference.doc*, and so on).
- Use the FrameMaker Table menu to create tables.
- Use only paragraph and character tags from the current templates; don't make up your own, and don't redefine existing tags.
- Don't use any of the TextInd paragraph tags to create multi-paragraph Notes, Cautions, and Warnings. Use two soft returns (<**`Alt`**> <**`Enter`**>) to get a multi-paragraph look when necessary.
- Tag words that appear in the glossary with the character tag GlossaryItem. Do this once per section, not every time the word appears. If you aren't creating a glossary for your book, don't use the GlossaryItem character tag.
- Don't begin character tags in the middle of a word.
- Don't include whitespace in character tags (except for spaces between words in some tags that can be used for multi-word phrases, such as GlossaryItem).
- Create copyright, trademark, and registered trademark symbols according to the instructions in "Create Valid Copyright and Trademark Symbols" on page 60.
- Begin frontmatter files with the Title tag. Begin chapters with the ChapNum and ChapTitle tags (in that order). Begin appendices with the AppNum and ChapTitle tags (in that order). Begin introductions or prefaces with the ChapTitle tag (do not include ChapNum tag). Begin glossaries with the GlossaryTitle tag. (All of these tags are set up for you in the templates; be sure not to change the order in which they're used at the beginnings of files.)

- Use FrameMaker's cross-reference formats. These automatically become links online.
- Don't create cross-references to AppNum or ChapNum paragraph tags. Cross-reference the associated ChapTitle instead.
- Import all figures by reference, from a directory named *print*. The *print* directory must be in your book directory (the directory containing the FrameMaker files for the book).
- Make sure you use supported formats and processes when creating figures (see Chapter 6, "Working With Figures").

Appendix B

Capturing Images for FrameMaker Documents

It's often necessary to use screen captures as illustrations in books. This appendix contains some instructions on how to snap a portion of your screen. There are a variety of tools available to aid you in producing screen captures; use whichever seems most appropriate to your needs. Feel free to use tools not mentioned in this appendix, as long as they output files in standard RGB format.

To use the *bsnap*, *ssnap*, and *winsnap* tools, you must append */usr/share/Insight/bin* to your *path* or PATH environment variable. This variable is initialized in your *~/.cshrc* file if you use the C shell or a shell derived from the C shell, and in your *~/.profile* file if you use the Bourne shell or a shell derived from it. Be sure to use the *rehash* command after editing your *.cshrc* file.

Screen Capture File Names

Name your screen capture figure files descriptively, so that someone else can tell which figures go where. The filenames should end in the *.rgb* suffix. For example, if your book is called *MyBook*, and you have three figures in chapter one, you might name the figures *mybook_fig1.1.rgb*, *mybook_fig1.2.rgb*, and *mybook_fig1.3.rgb*. Or you might use descriptive names such as *01.1.sample.rgb*, *01.2.menubar.rgb*, and *01.3.dialog.rgb*.

Screen Capture Procedure

The screen capture procedure is slightly different depending on what you want to capture: full windows, portions of windows, or pull-down menus. Note that for any of these approaches, if you want to show your screen capture against a white background you must first place the background, then place the window you're capturing on top of that background. One way of doing this is to open a shell window with a white background:

```
% xwsh -bg white
```

Screen Captures of Windows

If you want to capture an entire window, with or without its borders, use *bsnap*. (You can use *winsnap* if you prefer; it does essentially the same thing, though it provides fewer options and can't include window borders.) See "Tools and Where to Get Them" on page 96 for more information about these capture utilities.

1. In a shell window, change to the directory where you are going to store the screen captures (this is usually the *orig* subdirectory of your working directory for a given book).
2. Decide what you're going to name the screen capture. (See "Screen Capture File Names" earlier in this chapter for some guidelines.)
3. Bring the window you want to capture to the front, leaving part of your shell window visible.
4. If you want to include the window's border (including its title bar), launch *bsnap* like this from the shell window:

   `% bsnap` *filename*
5. If you don't want to include the window's border, launch *bsnap* like this:

   `% bsnap -n` *filename*
6. Click anywhere in the window you wish to capture.
7. Wait a few seconds for the capture to complete.
8. Verify that the capture worked as desired:

   `% ipaste` *filename*
9. If the image isn't as you expected, repeat steps 3 through 8.

Screen Captures of Portions of Windows

If you only want to snap part of a window, it's best to use *snapshot*. See "Tools and Where to Get Them" on page 96 for more information about this capture utility.

1. Change to the directory where you are going to store the screen captures (this is usually the *orig* subdirectory of your working directory for a given book).
2. Decide what you're going to name the screen capture. (See "Screen Capture File Names" earlier in this chapter for some guidelines.)
3. Bring the window you want to capture to the front, leaving part of your shell window visible.
4. Start *snapshot* from the shell window:

   ```
   % snapshot
   ```
5. Use the right mouse button in the *snapshot* window to bring up the *snapshot* menu. Select "New file name" from that menu. Enter the name you want your screen capture to have. Press **`<Return>`**.
6. Move the *snapshot* window out of the way:
   - Move the cursor over the *snapshot* window.
   - Press **`<Alt>`**+**`F7`**.
   - Move the cursor; the *snapshot* window's outline moves with it.
   - Click the left mouse button when the window is where you want it.
7. With the cursor still in the *snapshot* window, hold down the **`<Shift>`** key to let *snapshot* retain keyboard and mouse focus.
8. Still holding down the **`<Shift>`** key, move the camera cursor into the window you want to capture.
9. Drag out a rectangle, using the left mouse button, around the portion you want to capture.
10. Move the cursor off to one side and, still keeping the **`<Shift>`** key depressed, use the right mouse button to bring up the *snapshot* menu. Choose "Save as *filename*" from that menu.
11. To make additional screen captures, repeat steps 7 through 10.

Screen Captures of Pull-Down Menus

Use the same procedure for pull-down menus as for complete windows, except:

- Use *bsnap -s 5* to make *bsnap* delay five seconds before beginning the capture.
- After you click in the window you're capturing from, quickly pull down the menu you want to capture. Keep it pulled down for several seconds, until *bsnap* finishes.

Tools and Where to Get Them

The *snapshot* utility (*/usr/sbin/snapshot*) is part of the IRIX 5.3 and later system software, in the eoe2.sw.gltools software subsystem.

InSight Professional Publisher includes three shell-script capture utilities, all of which can be found in */usr/share/Insight/bin*:

- *bsnap*: allows variable time delay with the **-s** command-line option. Includes window borders by default; to avoid window borders, use the **-n** option.
- *ssnap*: identical to *bsnap*, except that after capturing a window it shrinks the window to 65% of its size in both horizontal and vertical directions, then brightens and sharpens the result.
- *winsnap*: has no command-line options. Doesn't capture window borders.

All three scripts require */usr/sbin/scrsave* (which is part of the eoe2.sw.gltools subsystem in the IRIX 5.3 and later distributions). *ssnap* also requires */usr/sbin/izoom* and */usr/sbin/hipass3*, both of which are part of the eoe2.sw.imagetools subsystem.

In addition to these tools, if you have an Indy™ system you probably have the */usr/sbin/capture* utility installed on it. *capture* is part of the dmedia_tools.sw.movietools subsystem. It can capture images from an IndyCam™, a video input, or the screen. See the capture(1) reference page for more information.

Appendix C

IRIS Insight Book-Building Architecture

This appendix describes the current IRIS InSight book-building architecture and process.

Book-Building Architecture

Figure C-1 shows the general process flow for building the textual content of an InSight book. Following the figure is a more detailed text description of the process.

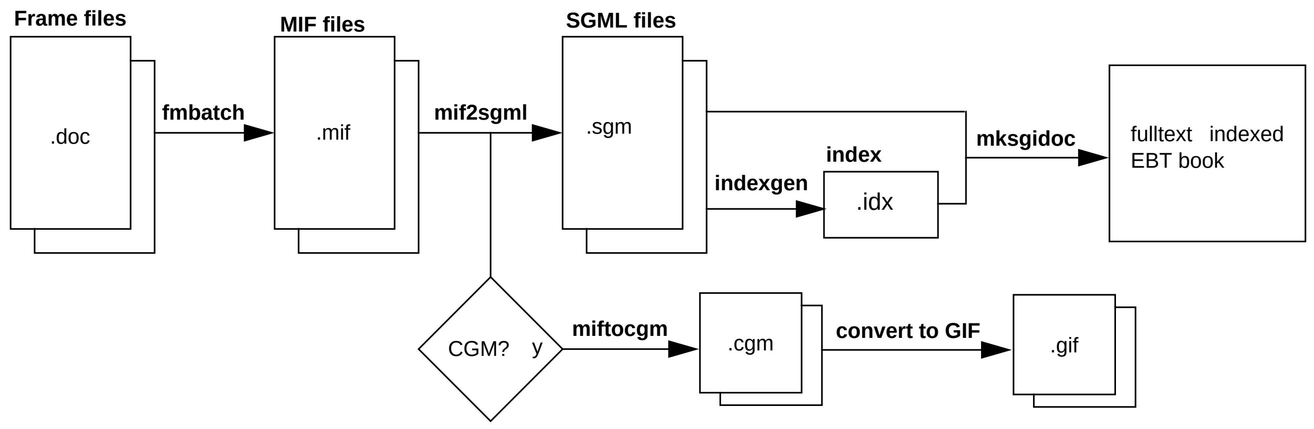


Figure C-1 Architecture for Book-Building

Authors compose a book using FrameMaker, then “compile” the book into an InSight viewable book:

- The FrameMaker files (*.doc*) are converted to Maker Interchange Format (*.mif*) using FrameMaker’s *fmbatch* utility. A special file, *conditional.doc*, is used during this process to force the proper FrameMaker conditionals to be applied.
- The next phase is to translate the MIF files (*.mif*) into SGML files (*.sgm*) using the *mif2sgml* translator from Silicon Graphics. After each file is translated, the *.sgm* file is scanned to see whether it contains any *CGM* figure references. CGM figures are generated from FrameMaker line art. If the *.sgm* file contains CGM figure references then it is run through Carberry’s *miftocgm* utility and every anchored frame is converted to a CGM image. The file is scanned again to convert the CGM images to GIF images for use by InSight.
- Once all the SGML files have been generated for the book, they’re concatenated and the *indexgen* utility is run over them to create a back-of-the-book index. This index is then concatenated to the rest of the SGML files, and an <SGIDOC> </SGIDOC> tag pair is wrapped around the result to create the master SGML file that is fed into EBT’s *mksgidoc* command to create an InSight viewable book.

Glossary

book-build

The process of converting a book from FrameMaker files to *SGML* files using IRIS InSight Professional Publisher tools. Sometimes called a “build.”

CGM

Computer Graphics Metafile image file format. An “infinite-resolution” image format, relying on shape descriptions rather than bitmaps.

chapter tabs

Pages used in a hard-copy book to mark the beginnings of chapters. Tabs often have shaded edges so that a reader can easily find them by looking at the edge of the book.

chunks

Small units of text, written so as to make sense without surrounding context. Text intended for online viewing should be chunked as much as possible, since there’s no guarantee that a reader will have read any surrounding sections.

closing tag

The second tag in a pair of *SGML* tags. The corresponding *opening tag* stops applying to subsequent text after the closing tag.

composite figures

Figures in a FrameMaker file which consist of more than one image—either multiple imported image files in your book directory, or a combination of FrameMaker line art and one or more imported image files. Note that it’s often best to create a composite image outside of FrameMaker—using ShowCase, for instance—and snap the result, in order to import only one file into a given frame in FrameMaker and thereby make color separations easier. However, it’s legal to import multiple files into a single frame should you wish to do so.

DTD

Document Type Definition: a formal description of the allowable structure of a document. Specifies, for instance, what tags can follow what other tags.

entity

In *SGML* jargon, an entity is a named definition of a character, usually named with a cryptic-looking string like "&ldquo;" (which indicates a left double quotation mark).

known error

An error during the *book-build* process with an obvious cause, recognized by the build tools.

MIF

Maker Interchange Format: an ASCII format for FrameMaker files which makes it easier to convert between FrameMaker's normal binary format and the formats of other systems and word processors.

opening tag

The first tag in a pair of *SGML* tags. An opening tag applies to all subsequent text up to the corresponding *closing tag*.

part tabs

Pages used in a hard-copy book to mark the beginnings of multi-chapter sections of the book. Tabs often have shaded edges so that a reader can easily find them by looking at the edge of the book.

production editor

The person who handles production for a book—can include anything from incorporating an online book into a software product, to checking a hard-copy book's pagination, to doing color separations for figures.

SGIDOC

The name of the *DTD* used by Silicon Graphics.

SGML

Standard Generalized Markup Language: a standardized format for describing structured documents in terms of their nested structures.

short name

Another name for the *short title* of a book.

short title

An abbreviated form of the title of an InSight book. Appears as an icon label when the book window is iconified. Also called a “short name.”

unknown error

An error during the *book-build* process with a non-obvious cause, not recognized by the build tools.

Index

We'd Like to Hear From You

As a user of Silicon Graphics documentation, your comments are important to us. They help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics to comment on:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Important Note

Please include the title and part number of the document you are commenting on. The part number for this document is
007-2863-001.

Thank you!

Three Ways to Reach Us

The **postcard** opposite this page has space for your comments. Write your comments on the postage-paid card for your country, then detach and mail it. If your country is not listed, either use the international card and apply the necessary postage or use electronic mail or FAX for your reply.

If **electronic mail** is available to you, write your comments in an e-mail message and mail it to either of these addresses:

- If you are on the Internet, use this address: techpubs@sgi.com
- For UUCP mail, use this address through any backbone site:
  [your_site]!sgi!techpubs

You can forward your comments (or annotated copies of pages from the manual) to Technical Publications at this **FAX** number:

415 965-0964