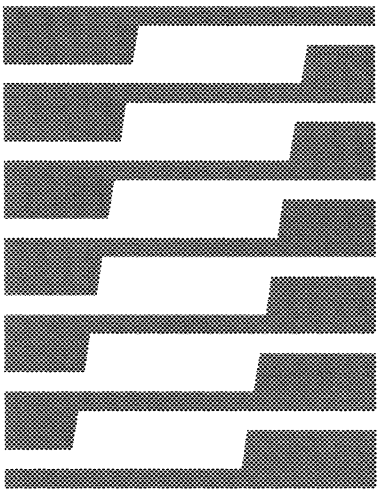
Interleaf



Technical Publishing Software

File Formats

Release 3.0

This manual was prepared using the Interleaf publishing software.

Interleaf reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Interleaf to determine whether any such changes have been made. This manual may not be reproduced and is intended for the exclusive use of Interleaf's customers.

The terms and conditions governing the sale of Interleaf hardware products and the licensing and use of Interleaf software consist solely of those set forth in the written contracts between Interleaf and its customers. No statement contained in this publication, including statements regarding capacity, suitability for use, or performance of products, shall be considered a warranty by Interleaf for any purpose or give rise to any liability of Interleaf.

In no event will Interleaf be liable for any incidental, indirect, special, or consequential damages (including lost profits) arising out of or relating to this publication or the information contained in it, even if Interleaf has been advised, knew, or should have known of the possibility of such damages.

The software programs described in this document are copyrighted and are confidential information and proprietary products of Interleaf. The copyright laws prohibit the copying of this manual or the software programs without the written consent of Interleaf, except in the normal use of the software or to make a backup copy. This exception does not allow a copy to be made for others. Copying, under the law, includes translating into another language or format.

© 1987 Interleaf, Inc.
Ten Canal Park
Cambridge, Massachusetts 02141
(617) 577-9800

All rights reserved.
Printed in U.S.A.
First Printing: February 1987

Interleaf, the Interleaf logo,  , Workstation Publishing Software, and Technical Publishing Software are trademarks of Interleaf, Inc.
Unix is a trademark of Bell Telephone Laboratories, Inc.
AEGIS is a trademark of Apollo Computer Inc.

Table of Contents

File Formats

Interleaf ASCII Format for Text **1**

Creating ASCII Format Documents 1-2
- Comparing Versions of ASCII Format 1-2
- How Interleaf Documents Are Saved 1-4
- How Interleaf Documents Are Opened 1-5

Markup in ASCII Format Documents 1-6
- Declarations and Commands 1-6

Creating a Marked-Up Document 1-16
- Printing an ASCII Document 1-16
- Limitations of Save ASCII 1-17
- Canceling an ASCII Load 1-17
- Templates for ASCII Documents 1-17
- Using Include Commands 1-18

Special Loading Features for Documents Created Outside the Interleaf Desktop 1-23

Non-Marked-Up Files 1-24

Desktop Document Icons 1-24

Hints on Writing Translation Programs 1-25

Specifications for Interleaf ASCII Format 1-26
- Quoting Rules 1-26
- Naming Conventions 1-27
- Basic Syntax Rules 1-27
- Declaration and Command Specifications 1-29

Default Values for Interleaf ASCII Format Documents (Markup Version 5.2) 1-41
- Document Defaults 1-41
- Page Defaults 1-41
- Autonumber Defaults 1-42
- Component Defaults 1-42
- Frame Defaults 1-42

Additional Information 1-43
- Special Characters 1-43

Interleaf ASCII Format Error Messages 1-45

ASCII Format for Diagramming Objects **2**

Overview 2-1
- Syntax 2-1

General Format Descriptions 2-3
- Format for Diagramming Objects 2-3

Field Specifications 2-14
- Font Representation 2-14
- Textures 2-16
- Line Patterns 2-17
- Flags 2-17

Error Messages 2-18

Appendixes

Default Document in ASCII Format ..... **A**
Document with Text in ASCII Format ..... **B**
Using an Include Command ..... **C**

Index

Chapter 1

Interleaf ASCII Format for Text

Interleaf ASCII Format is the name of the language that describes an Interleaf publishing software document. Interleaf ASCII Format documents are documents that contain special markup so that they can be read by the Interleaf publishing software and can be exchanged between Interleaf software and other software.

A marked-up document contains **declarations** and **commands** that tell the publishing software how to format the document and its components and which fonts to use and where to use them. To make sure a document is correctly handled by the publishing software, it should be marked up according to the specifications in this chapter.

ASCII files that are not marked up also can be opened in the Interleaf publishing software. However, you do not have as much control of the structure of the documents as you would have if the documents were marked up.

There are three kinds of documents discussed in this chapter.

- *Marked-up documents* contain instructions written in Interleaf ASCII Format. A marked-up document begins with the characters <!.
- *Non-marked-up documents* do not contain Interleaf ASCII Format instructions. A non-marked-up document does not begin with the characters <!.
- *External documents* refers to marked-up documents created outside of the publishing software.

The chapter also covers these topics related to ASCII Format and the publishing software:

- Interleaf ASCII Format
- Interleaf publishing software documents
- externally created documents
- Include files
- specifications for Interleaf ASCII Format
- default values for Interleaf ASCII Format documents
- hints on writing translation programs
- Interleaf keyboard mapping
- ASCII Format error messages

Creating ASCII Format Documents

This chapter describes the three ways to produce an Interleaf ASCII Format document:

- by using the **Save ASCII** command when a document is open on your desktop
- by using the Interleaf markup language to mark up documents manually
- by using a translation program, either provided by Interleaf or written by an engineer in your company

The **Save ASCII** commands on the Document pulldown menu create documents that can be exported to other machines and can be edited outside the publishing software. Saving ASCII is also the most efficient way to create templates for use with the **Include** commands discussed in this chapter.

Marking up a document manually is useful for creating documents on word processors or with any non-Interleaf editor. When these documents are transferred to and opened on your Interleaf desktop, they will need minimal attention because the markup will have supplied the Interleaf structure in advance.

Alternatively, you can markup just the body of a document and then use the **Include Declarations** command to supply the structure of the document.

Interleaf provides a number of filters that translate files from other devices to Interleaf ASCII Format. See the manual *File Transfer* for details. If your device uses a format that Interleaf does not support, you can write your own translation program if you are proficient with the UNIX or AEGIS operating systems and C or another programming language.

Comparing Versions of ASCII Format

This chapter describes Version 5.2 of the Interleaf ASCII Format, which is produced by Release 3.0 of the Interleaf publishing software. Future releases of the Interleaf publishing software will always be able to load documents created by earlier versions. On the other hand, earlier releases cannot load Interleaf ASCII Format documents produced by Release 3.0 because of new features. There is a filter program that will strip a newer document of incompatible elements; please see the chapter on *opsback* in the manual *File Transfer*.

This section presents information that may be useful for people who have used a previous version of the Interleaf ASCII Format. It outlines the differences between Versions 4.0 and 5.2 of the Interleaf ASCII Format. New users may want to skip this section.

There are a number of significant differences between Version 5.2 and Version 4.0 of the Interleaf ASCII Format.

With the new version documents are loaded and saved faster than with the previous version.

In the ASCII format, there have always been component **classes** or **masters**. With this release, the definition of a component master has changed. Previously, the properties of the component classes were identical to the properties of the first component of each name in the document. Now component classes exist apart from any particular component. See *Component Declarations* on page 1-8.

In this release, there is markup for autonumber declarations (*Autonumber Declarations* on page 1-7) and for autonumber and autoreference commands (*Autonumbers and Autoreferences* on page 1-12); for index tokens (*Index Tokens* on page 1-14); and for documents created with the **Index** and **Create TOC** commands on the Icon Selected popup menu within a book window.

There have been changes in the numbers assigned to the various fonts when you **Save ASCII**. When you mark up documents, you need never be concerned about these changes since you can declare any font numbers you like in the *<!Font Definitions, ...>* declaration and the software will automatically convert them to the numbers used by the publishing software.

When you save a document in ASCII format within the publishing software, page breaks are now indicated in the markup. See *Page Breaks* on page 1-11.

A number of changes and additions have been made to the declarations. See *Specifications for Interleaf ASCII Format* (page 1-26) for the specifications for each declaration and *Default Values for Interleaf ASCII Format Documents* (page 1-41) for the default values.

The *Hyphenation = on/off* and *Consecutive Hyphens =* settings have been moved from the *<!Document, ...* declaration to the *<!Page, ...* declaration, but old versions are still supported.

The following settings have been added to the *<!Page, ...* declaration:

- Starting Page Number = Inherit
- Revision Bar Placement
- Page # Prefix
- Balance Columns
- Vertical Justification
- Margin Stretch
- Margin Shrink
- Frame Margin Stretch
- Feathering
- Maximum Feathering
- Vertical Justification Pages
- Depth At Page Break
- Depth No Page Break

In the *<!Document, ...* declaration, *Keep Unused Classes* is no longer necessary because a master component is created for each *<!Class, ...* regardless of whether there is an actual component by that name in the document. If this entry is present in old documents, it is ignored. The following settings have been added to the *<!Document, ...* declaration:

- Double Sided
- Manual Sheet Feed
- Print Revision bars, Strikethroughs, and Underlines

There are now *<!Master Frame, ...* declarations. See *Master Frame Declarations* on page 1-10. In *<!Master Frame,...* declarations, as well as in the commands for individual frames, there are the following settings that did not exist in previous versions:

- Name
- Shared Contents
- Diagram

When a document is saved in ASCII, there is a *Page #* entry in the commands for individual frames.

Same Page and *Same Column* are now synonymous in a *<!Master Frame, ...* declarations and in *<Frame* commands.

Component *<!Class, ...* declarations now may include *New Page*, *Allow Page Break Within*, and *Allow Page Break After*.

How Interleaf Documents Are Saved

Documents are stored by the publishing software in one of two forms: as *fast* documents or as *Interleaf ASCII Format* documents. A fast document is stored in a binary format that cannot be used easily by programs other than Interleaf's. Restricted code numbers assigned by Interleaf at the beginning of a fast document distinguish it from an ASCII document or any other kind of file.

When you choose the **Save ASCII** command on the Document pulldown menu (Figure 1-1), you call up the ASCII storage mechanism or *dumper*. The ASCII dumper saves the document in the form described in this chapter. A document in this form can be transported to other computers or edited with an ordinary ASCII editor that you may have on your workstation.

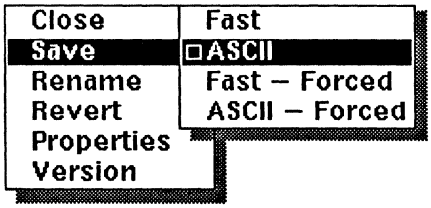


Figure 1-1. Save submenu

How Interleaf Documents Are Opened

When you use the **Open** command on the publishing software desktop to open a document, the publishing software checks to see whether the document is an ASCII document or a fast document.

You cannot tell by looking at an icon whether it is an ASCII document or a fast document. A document icon that looks like computer paper (▭) *usually* contains an ASCII document, but it need not; and an Interleaf document icon (🗋) *usually* contains either an ASCII or a fast document.

If a document does not appear to be "fast", the ASCII *loader* is called up to load the document onto the screen. The ASCII loader then looks at the first few characters of the document. If the first characters in the document are *<!*, the loader knows the document is an Interleaf ASCII Format document and loads it accordingly. Even if the loader does not see Interleaf markup *(<!)*, it will still load the document. See the section *Files that Have Not Been Marked Up* for a discussion of how it interprets such a document.

The only time a document will not be loaded is when it contains unusual binary codes among the first few characters.

While a document is being loaded by the ASCII loader, the *Loading ASCII* message flashes every 4096 characters.

The ASCII loader is used not only when you open an ASCII document, but also when you **Paste** an ASCII document — including documents produced when you use the **Copy** command in a virtual terminal window on a SUN workstation — into another document on your desktop.

Markup in ASCII Format Documents

There are two kinds of information in an Interleaf ASCII Format document: the markup and the ASCII text of the document. All of the information that you see inside the angle brackets <...> is markup. There are two kinds of markup: declarations and commands.

Documents coded in Interleaf ASCII Format are fairly easy to understand. Appendix A shows how the default *document* on the desktop looks in this format, and Appendix B shows how a two-column document with text looks.

Declarations and Commands

A **declaration** is preceded by the characters <!. Each declaration must end with >. In Appendix A and Appendix B, *<!OPS, Version = 5.2>*, *<!Document, ...>*, *<!Font Definitions, ...>*, *<!Page, ...>*, *<!Autonumber Stream, ...>*, *<!Class, ...>*, and *<!Master Frame, ...>* and are document-wide declarations.

The declaration for each component name contains the word *Class* (in ASCII markup, *Class* means the same thing as *Master Component* within the publishing software). In Appendix B, *<!Class, paragraph, ...>* refers to all of the components called *paragraph*, and *<!Class, title, ...>* refers to all of the components called *title*.

Commands describe the actual content of the document in a specific sequence. This includes the placement of individual components and also information within a component. A command is preceded by the character <. In a command the < is not followed by !. Each command must end with >.

For example, in Appendix B, the command *<paragraph>* on page B-6 is used to indicate the *position* of the component *paragraph* in the document. This is different from the declaration *<!Class, paragraph ...>*, which describes the *properties* of components named *paragraph*, but does not call for the creation of any particular *paragraph* component.

The command *<F33>* in the text of the first *paragraph* component indicates a change to Classic 10 italic font. The command *<F0>* always indicates a return to the component's default font, in this case Classic 10 roman.

A full list of declarations and commands and the information they contain can be found in *Specifications for Interleaf ASCII Format* (page 1-26).

Font Declarations

The basic form of the font declaration is *<!Font Definitions, ...* followed by a list of the fonts that are used in the document. One example of a simple font declaration is

```
<!Font Definitions,
 F31 = Classic 10
 F32 = Classic 10 Italic>
```

The reference *F31* is the number by which *Classic 10 roman* is currently known at Interleaf, but *F31* may not correspond to its number at any other installation or in future releases of the software. Since when a document is saved the font numbers are assigned to it on the basis of the fonts available *at a particular time at a particular installation*, it may be more convenient for you to assign your own numbers in the font declaration and maintain these consistently when you mark up documents. The software will convert your numbers to the proper fonts when a document is loaded. In other words, if you were to use the designation *F1 = Classic 10*, this would be interpreted to mean the same thing as *F31 = Classic 10* if the document is loaded with a release that uses this number. For some other release, your *F1 = Classic 10* might be equivalent to Interleaf's *F190 = Classic 10* or *F60 = Classic 10*.

If you write a program that translates Interleaf ASCII Format to some other format, the program must interpret the *<!Font Definitions, ...* declaration, so that the command *<F32>*, for example, will be properly translated as italic.

Autonumber Declarations

The basic form of the autonumber declaration is *<!Autonumber Stream, ...* followed by the name of the autonumber stream, the number of levels in the stream, and the closing >. For example, *<!Autonumber Stream, List,* 3> is the declaration for a stream named *List* with 3 levels. An autonumber stream can have 1 to 8 levels. There can be other entries between the number of levels and the closing >. These entries are comparable to the entries on the Autonumber Stream property sheet. Only those entries that are different from the default need to be specified (see *Default Values for Interleaf ASCII Format Documents* on page 1-41).

The declarations for two default autonumber streams — *List* and *Outline* — are supplied automatically when a document is opened on the desktop. You may declarations for other streams and you may change the properties of the two default streams.

Component Classes

As in the publishing software itself, text in ASCII Format is structured by *components*. Much of the information you see in an Interleaf ASCII Format document, as illustrated in Appendix A and Appendix B, is identical to the information in component property sheets used in the publishing software.

One important difference is that while the property sheets for each individual component display the full set of properties, a document in Interleaf ASCII Format does not list *every* property of each component, since this would produce a verbose description of the document. Instead, when a property has the default value, it is not listed in the component class declaration. In commands indicating

the location of a component, only those properties that differ from the component declaration or from the defaults are listed. See the section *Default Values for Interleaf ASCII Format Documents* on page 1-41.

Component Class Declarations

A component *class* declaration defines the general or default properties for the components with a particular name. In the publishing software itself, component classes are referred to as *master components.*

In each component class declaration, the component name is preceded by *Class,* for example *<!Class, paragraph, ...>*. In Appendix A, the class properties for *paragraph* are *Top Margin = 0.07 inches, Bottom Margin = 0.07 inches, Line Spacing = 1.308 lines,* and *Font = F31.*

It would be an error to have two or more *<!Class, ...>* declarations specifying the same component name.

When a document is saved, the properties assigned to each component class are taken directly from the master component for that class.

In the body of the document, the command *<paragraph>* indicates the beginning of a *paragraph* component. If the properties of a particular component, in this case, *paragraph,* are identical to the properties of the component class (*<!Class, paragraph, ...>*), only the name of the component will be indicated; for example, *<paragraph>*.

If the properties are not identical to those of the particular component class, this will be indicated within the command. For example, *<paragraph, Font = F33>* would indicate that the font of this paragraph is different from the font in the class declaration for *paragraph* components.

If there are several successive components of the same class (e.g., *paragraph*) that have exactly the same properties, then the command *<paragraph>* will not be repeated at the beginning of each new *paragraph.* All that will separate the components when they are saved in Interleaf ASCII Format is a blank line. The first component in such a series can have the standard component class properties, or it can have different properties of its own. In either case, the blank line indicates that the next component has properties identical to the one before it. See the section *Newlines* on page 1-28.

Defaults for Component Class Properties

On page 1-41, the section *Default Values for Interleaf ASCII Format Documents* lists default property values. These are the values that the publishing software automatically assigns unless others are indicated in declarations or by commands.

These assignments are made because there are certain property values that are very common. For example, many component classes will have Left Margin = *0* inches, so *0* is the default setting for Left Margin.

When marking up a document, if you want to override any of the defaults and declare a different property value to be used throughout the document, you can insert another declaration called the *<!Class Defaults, ...>* declaration. For example, if you wanted the entire document in Appendix B (or most of it) to be *Classic 10 italic* instead of *Classic 10 roman*, you would enter the following:

```
<!Class Defaults,
  Font = F33>
```

This font must have been defined in the *<!Font Definitions, ...>* declaration, which must precede the *<!Class Defaults, ...>* declaration.

Component Masters During a Paste Operation

To ensure that the components in a document maintain their properties, the software temporarily contains two sets of component masters when components from one document are pasted into another. As the pasted document is loaded, the software compares the properties of each component with the masters for the document it is being pasted into.

If the component being pasted has a name different from any of those in this document, the master definition for the component is added to the masters in the new document. For example, if a component named *para* is pasted into a document with components named *paragraph*, the master definition for *para* will become part of the new document.

If the component being pasted has the same name as one of the masters, its properties are compared to the properties of the master. If they are different, the differences are retained for the individual component; but, when the loading is completed, the set of component masters from the pasted document are discarded. For example, if a *paragraph* component in 12 point Classic bold is pasted into a document in which the master definition for *paragraph* components calls for 10 point Classic roman, the *paragraph* will retain its font, but the master definition will continue to have 10 point Classic roman as its font.

When you **Paste** an ASCII format document into another document on your desktop, use the **Paste** command on the Component Location popup menu to ensure that the components are properly delineated. If you use the **Paste** command on the Text Location popup menu and the document has more than one component in it, you will see a stickup menu asking you to confirm that you want to execute the **Paste** command. If you confirm, you will lose component information that may not be easily recoverable.

Master Frame Declarations

The concepts of master frames and named frames are new with this release. Just as the component classes (masters) exist apart from any particular component, frame masters exist apart from any particular frame in a document.

The representation of frames and frame masters in Interleaf ASCII format differs from the representation of components and component masters in one significant way: except for frames with shared contents, each *master frame* and each actual *frame* is represented by the full complement of properties, not just the properties that differ from the defaults, as is the case with components.

If *Shared Contents* is set to *yes* in the command for a specific frame, there should not be markup for the diagram in the command itself, but there should be a diagram in the declaration for the master frame with the same name.

The following is the markup for a typical master frame:

```
<!Master Frame,
        Name =                      "Following Anchor",
        Placement =                 Following Anchor,
        Horizontal Alignment =      Center,
        Height =                    6.0 inches,
        Width =                     5.0 inches,
        Diagram =                   ...>
```

See the chapter *ASCII Format for Diagramming Objects* for the markup that follows *Diagram =*.

See Appendix B for examples of *<!Master Frame, ...* declarations. The master declaration and the command for the frame name "filled space" comprise an example of the markup for a frame with *Shared Contents*.

If an externally created ASCII document contains no *<!Master Frame, ...* declarations, six default master frames are loaded with the document when it is opened on the desktop. If there are any *<!Master Frame, ...* declarations in an external document, only these masters are loaded with the document when it is opened on the desktop.

Markup within Component Text

Not all of the markup has to be associated with declarations or component commands. There can be markup within the text of a component. This markup includes:

- page break information
- font changes and attributes
- tabs
- returns

- special characters
- autonumbers and autoreferences
- index tokens
- frames

The markup within components is indicated by commands, for example, *<F33>*, *<#7f>*, and *<Tab>*. The most frequently used commands for markup within components are these: *<F ...>* (font changes), *<Tab>* (tabs), *<SP>* (a hard or unbreakable space), *<#xx>* (special characters, such as ¢), and *<HR>* (hard return). These commands are placed in the text at the specific locations where they are to take effect.

Page Breaks

When you save a document in ASCII, page breaks are indicated by <|, *n*> where *n* is the page number including the prefix if there is one. Roman numerals are preserved. Thus, <|,2> denotes page 2 in Arabic numerals without a prefix, <|,"page 2"> denotes page 2 with a prefix, and <|,ii> denotes page 2 in lowercase Roman numerals without a prefix.

In the markup for all frames except header, footer and At Anchor frames, there is the field *Page # = n* where *n* is the page number including the prefix if there is one. Roman numerals are preserved.

See *Quoting Rules* on page 1-26 for information about when *n* should be enclosed in quotation marks.

These page numbers are only for the convenience of people or programs that look at documents saved in ASCII. They are ignored by the publishing software when the document is loaded.

Changing Fonts

The markup for a font change can be entered as in the example in Appendix B, *<F33>text in the new font <F0>*. The numbered fonts are defined in the *<!Font Definitions, ...>* declarations that must appear at the beginning of the document. In this example, *<F33>* indicates that the words following the command are to be Classic 10 italic. *<F0>* indicates a return to the default font of the component.

You could also use a font number in the command at the end of the font change. For example, *<F33>text in the new font <F60>* would mean that you were switching from Classic 10 italic to Modern 18 italic.

The Interleaf font code numbers used when a document is saved change from one release of the software to the next and may change if new fonts are installed. For your convenience, you may want to use different numbers to indicate the fonts you want. If, in your markup, you always use the same numbers for each font, you will get accustomed quickly to your own set of numbers. Just make sure you have defined the fonts and the numbers you are using in the

<!Font Definitions, ...> at the beginning of the document. When the document is loaded and then saved in Interleaf ASCII Format, the fonts will be assigned the code numbers that the Interleaf software uses.

See the section *Special Loading Features for Documents Created Outside the Interleaf Desktop* for shortcuts you can use when changing fonts.

Font Attributes

There are three attributes associated in Interleaf ASCII markup with fonts: *underlining, strikethrough*, and *revision bars*. The mark up for these three attributes comes directly after the font number. *Underlining* is indicated by @U, *strikethrough* by @S, and *revision bars* by @R. For example, *<F33@U@R>* indicates that the text that follows is Classic 10 italic and that it is underlined and has a revision bar.

Using Special Characters

In Appendix B in the text of the first *paragraph* on page B-6, 7f is the hexadecimal code for the opening double quotation mark. The command *<#7f>* is put in text exactly where the character belongs.

See the layouts in *The Keyboard* chapter in the *Reference Manual,* Volume 1 for the positions of the other special characters. Also see the section *Interleaf Keyboard Mapping* at the end of this chapter for their hexadecimal codes.

Autonumbers and Autoreferences

In Appendix B, on pages B-6 and B-7, there are commands for several autonumbers. The following is an example of a simple autonumber command: *<Autonum, figure, 1>*. An autonumber command always consists of *<Autonum,* followed by the name of the autonumber stream, the level of the autonumber, and the closing >.

There are two other entries that can come between the level of the autonumber and the closing >. If the autonumber is the first one in a stream, the command would be *<Autonum, figure, 1, First = Yes>*. It is not necessary to include the *First = Yes* entry because the software will insert it when the document is saved on the desktop. If the numbering of an autonumber stream is being restarted, the command would be *<Autonum, figure, 1, Restart = Yes>*. It is necessary to include the *Restart = Yes* entry. Otherwise, the numbering will not diverge. It can be used only in the command for a level 1 autonumber.

There is one other entry in the markup if the autonumber is used for autoreferencing. This is the *Tagname* entry. There are two forms that this entry can take. When you name the tag, the command will look like this *<Autonum, figure, 1, Tagname = properties>*. If you let the software name the tag, the command will resemble this: *<Autonum, figure, 1, Tagname = eIb5W3accal>*. The only part of the software-generated entry that will be recognizable and comprehensible is the very end, which will be the login of the user.

An autoreference can refer either to an autonumber or to the page on which the autonumber appears. The form is either <*Ref, Auto #, Tag = Tagname* (from *Autonum* entry)> or <*Ref, Page #, Tag = Tagname* (from *Autonum* entry)>. The *Tag* in the autoreference must the same as the *Tagname* for the autonumber that is being used as the reference point.

You can use *Tag* and *Tagname* interchangeably in markup. When you save documents, though, you will see *Tagname* = in autonumber markup and *Tag* = in autoreference markup.

If you want to use spaces in an autonumber tag, you must put the entry in quotation marks. Otherwise, the spaces will be stripped out when the document is loaded by the Interleaf software. You should also use quotation marks around a reference tag with spaces. See *Quoting Rules* on page 1-26 for more information.

In Appendix B, on pages B-6 and B-7, there are also commands for several autoreferences.

Tab Markup Commands

In the publishing software, there are four types of tab stops — left, right, center, and decimal — and four kinds of tab appearances — blanks, dots, lines, and underlines. The *tab stops* are indicated in the <*!Class, ...*> declarations. *Tab appearances* are indicated by commands in text.

The ASCII Format commands for the four appearances are

- <*Tab*> for blanks
- <*Tab.*> for dots
- <*Tab–*> for lines
- <*Tab_*> for underlines

Tabular material that was prepared outside of the publishing software may require some cleanup. This may be true of both *fixed width font* to *fixed width font* tables and *fixed width font* to *proportional font* tables. In both cases, you might need to adjust tab settings or add or subtract tab stops.

If you are using a *fixed-width font* for tables (for example, the Typewriter font) in which each character takes up the same amount of space on a line, there could be a difference between the amount of horizontal space that the characters require in the publishing software and what you expect.

The difference may be even more noticeable if your tables were originally in a *fixed width font* and are being converted by the publishing software to a *proportional font* in which each character requires a different amount of horizontal space.

Frames

In the declarations for each document created or saved with Release 3.0 (Version 5.2 of the Interleaf ASCII Format), there will always be at least one master frame definition. Where there is an actual frame in text, there will be markup for the particular frame and its contents. This markup will look like the following:

```
<Frame,
        Name =                      "Following Anchor",
        Placement =                 Following Anchor,
        Horizontal Alignment =      Center,
        Height =                    6.0 inches,
        Width =                     5.0 inches,
        Diagram =                   ...>
```

See the chapter *ASCII Format for Diagramming Objects* for markup that follows *Diagram =*.

If *Shared Contents* is set to *yes*, the markup for the diagram itself appears only in the declaration for the master frame. See Appendix B for other examples of *<Frame, ...* commands.

Index Tokens

Index tokens (or their markup) are inserted in text at the point where the index reference begins. The basic format for an index token is <Index, "*Heading*">. For example, <Index, "Looking and discovering"> on page B-6 of Appendix B would insert a Level 1 entry in the index under "Looking and discovering", and <Index, "Figures for thinking", "a large space and a small space", Typeface = Italic> would insert a Level 2 entry in italics under "a large space and a small space".

Even though there are spaces in these *Heading* entries, quotation marks are not strictly necessary as they are elsewhere in ASCII markup. However, quotation marks would be necessary if you used any of the other characters discussed in the section *Quoting Rules* on page 1-26, and we recommend that you use them in any case to make it easier to distinguish between *Headings* and other markup.

Other markup that can go inside an *<Index, ...* command includes markup for the range the reference covers — This Page/To Next/Count/To Named/See/See Also:

- If range is This Page, there is no need for an entry because this is the default.
- If range is To Next, the entry is simply *To Next*. This means that the range of the entry ends with the next component.
- If range is Count, the entry must include the number of components to be included in the reference (i.e., *Count = n*).
- If range is To Named, the entry must include the name of the component where the reference stops (i.e., *To Named = component name*).

- If range is See or See Also, the entry must include the name of the entry being referred to (i.e., *See entry name* and *See Also entry name* will work equally well).

Other entries that can be in an index token command:

- Sort or Sort String = (for a customized sort string). When you use *Sort* or *Sort String* to designate a customized sort string, the entry must follow the corresponding *Heading* entry.
- Doc = (this corresponds to *Index Document* on the Custom sheet of the Index property sheet).
- Index = if you want to name the index.
- Typeface = Bold/Italic/Italics/Normal/Roman. The default font for index entries is a 10 point Roman typeface. You can designate only a different weight or slant.

When you examine an index with an ASCII editor, you will find a very long list of fonts in the *<!Font Definitions, ...* declaration. We have included all the fonts that are likely to be necessary.

Tables of Contents

The components that are to be part of a table of contents or related documents like figure lists have a TOC entry in their *<!Class, ...* markup. For example, the markup for a component named *title1* might look like this:

```
<!Class, title1,
        Bottom Margin =              0.25 inches,
        Line Spacing =               1.146 lines,
        Alignment =                  Left,
        Font =                       F180,
        Allow Page Break Within =    no,
        Allow Page Break After =     no,
        Left Tab =                   0/1*13 inches,
        TOC Doc Name =               TOC>
```

When a TOC-style document is created, the software takes the first component in the document with the *TOC Doc Name* entry, converts it to a TOC component (for example, *title1TOC*), and makes it Classic 10 point bold. All other components with the same name are made Classic 10 point bold. The rest of the components with the *TOC Doc Name* entry are made Classic 10 point roman in the TOC document.

Creating a Marked-Up Document

A good way to begin creating marked-up documents from scratch is to follow these four steps. Note that the first two steps are performed within the publishing software and the last two are performed outside of the publishing software.

1. Create a document on the Interleaf publishing software desktop that contains the page and component characteristics you want, including the various fonts.
2. Save it as ASCII.
3. Examine the file by printing it and/or looking at it with another editor. In the next section, there are instructions for printing the document with its markup.
4. Copy this file to a place where you can easily access it. This could be somewhere on the workstation with the publishing software, on another workstation, on your network, or on a word processor.

When documents that contain Interleaf markup are prepared externally, remember to identify components consistently. If you make a mistake marking up a component class, you can correct it easily and quickly when you open the document on your desktop, provided you identified components uniformly. For example, an incorrect margin setting can be corrected in all components named *para* in your document by changing the *para* component's margin setting on the property sheet and using the **Global Apply** command. Had you created a group of components identical to *para* but named *paragraph*, you would have to repeat the **Global Apply** sequence to correct the same mistake for all *paragraph* components. See *Text Processing* and *Page Makeup* in Volume 1 of the *Reference Manual* for information on changing these properties on the desktop.

It is best to keep names short and simple, without blanks or punctuation. Avoid using names that correspond to command names, such as *tab*. Suitable names are *para, paragraph, title1, title2, section, head*, and *label*. See the section called *Specifications for Interleaf ASCII Format* on page 1-26 for rules concerning component names.

Printing an ASCII Document

Just as you need to use an editor other than the publishing software to examine the ASCII markup on the screen, you must print an Interleaf ASCII format document from outside the publishing software in order to see the markup.

To print an ASCII document with markup using UNIX:

- ✓ Either close your desktop or open a virtual terminal on your desktop.

- ✓ Change your working directory to the one that has the ASCII document in it.
- ☐ **lpr** *documentname*.**doc** <RETURN>
  Use this command to print the document on your default printer.

or ☐ **lpr** **-P***printername* *documentname*.**doc** <RETURN>
Use this command to print the document on any printer on your network.

To print an ASCII format document with markup using AEGIS:

- ✓ Move to a shell prompt ($) in any DM process window.
- ✓ Change your working directory to the one that has the ASCII document in it.
- ☐ **prf -pr** *documentname*.**doc** <RETURN>
  Use this command to print the document on your default printer.

or ☐ **prf -pr** *printername documentname*.**doc** <RETURN>
Use this command to print the document on any printer on your network.

Limitations of Save ASCII

When a document is saved in ASCII, all hyphenation points are discarded. When the document is loaded, they are re-computed. Thus, any hyphenation changes that were made by the user (discretionary hyphens) are lost. When the document is being saved, the following message appears briefly in the status line: *Warning: Discretionary hyphens will be lost.*

When a document is saved in ASCII, some fonts may be mentioned in the *<!Font Definitions, ...>* that are never used in the document. These font declarations are ignored when the document is loaded.

The position of the text caret is lost when you save a document in ASCII.

Canceling an ASCII Load

To cancel the loading of an ASCII document, you can use **Cancel** on the Interrupt stickup menu. You cannot cancel a **Save ASCII** command.

Templates for ASCII Documents

You can create any kind of document on your desktop, **Save ASCII**, and then use the document as a template for other Interleaf ASCII documents. Such a document is called a **template** because it is used as a pattern for other documents.

You can create templates for several kinds of typical documents — such as letters and memos — by using the Create submenu of the Desktop Nothing Selected

popup menu. Figure 1-2 shows the default Create submenu on the Sun (the submenu is the same on the Apollo except that there is no **Terminal** entry).

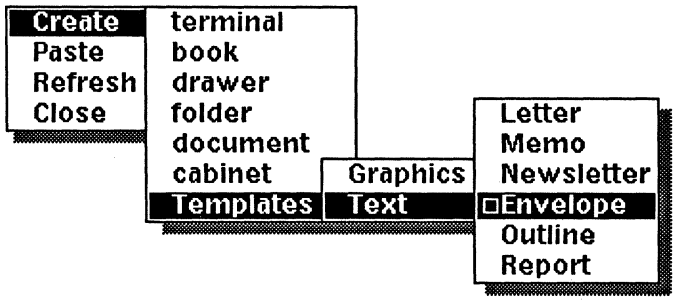


Figure 1-2. Desktop Nothing Selected popup menu

There are two ways to use a template to create other Interleaf ASCII documents:

- You can leave the template on the workstation running the publishing software, but use it as the basis for inserting commands in the body of a document you are creating on another workstation, a word processor, or a personal computer. After you have marked up the external document, you can transmit it to the workstation running the publishing software and use the template as an Include declarations file.
- You can copy the template to another workstation, word processor, or personal computer, add text and markup to it, and transmit the resulting document directly to your desktop. If you choose to use this method, you can skip the next section, *Using Include Files*, but be sure to read the section *Special Loading Features for Documents Created Outside the Interleaf Desktop* (page 1-23). Also be sure to refer to the section *Specifications for Interleaf ASCII Format* (page 1-26).

Using Include Commands

Once you have a file with marked-up text (but no declarations) on your workstation, you can use an **Include** command to incorporate into it either the declarations alone or both the declarations and the contents of an ASCII template document. This process will save you a great deal of time since you will not have to type in all the markup with the attendant opportunities to make typing mistakes.

An **Include** command can be inserted anywhere in a document. When it is inserted at the beginning, it often provides all the property declarations that will be used when the document is opened on the desktop.

There are two Include commands: **<!Include,** *pathname*> and **<!Include Declarations,** *pathname*>.

<!Include Declarations, ...> and **<!Include, ...>** begin with the characters *<!* and, therefore, look like declarations. However, they are not declarations in the same way that *<!Document, ...>* and *<!Font Definitions, ...>* are, so they are referred to as commands.

Use **<!Include,** *pathname*> to include an entire ASCII Format template — the declarations, headers and footers, and the contents of components — in your externally-prepared document.

However, it is not necessary to incorporate the entire text of an ASCII Format template document. **<!Include Declarations,** *pathname*> reads an ASCII Format template document until it reaches non-declaration markup or text. This declaration information becomes part of the document. The non-declaration markup — including header and footer information — and text are omitted.

To use an Include command, you must know the full pathname of the ASCII Format document you want to include. Any pathname that is valid for your operating system can be specified. Pathnames may be up to 80 characters long. The pathname must be supplied completely: ~ and * are not allowed. In addition, we recommend that you not begin an element in the pathname with a digit, that there not be more than one . in an element, and that you avoid using the following characters: ?, @, and $. If the full path is not specified, the current directory will be that of the user's desktop. See the section *Quoting Rules* on page NO TAG for a list of the characters that require that a pathname be enclosed in quotation marks.

To use <!Include Declarations, *pathname*>:

- Using an editor on your workstation, call up the externally-created document that you want to mark up.
- At the top of the file, before the first line of the document (where markup normally begins), type: **<!Include Declarations,** *pathname of the file you want to include*> <RETURN>
- On the line that precedes each paragraph, section head, or other component, type the component name from the <!Include file in command form (for example, *<title>* and *<paragraph>*). When components with identical properties occur in a row, the command must appear before only the first component. Then, a blank line can be used to separate the components with identical properties.

  At this point, the document might look like NO TAGa on a SUN workstation and NO TAGb on an Apollo workstation.

```
<!Include Declarations, /u/cal/desktop/recipe_template.doc>
<title>
Garlic Buttered Popcorn
<author>
by Mrs. J. H. Smith, Jr.
<paragraph>
Place butter and garlic in a small saucepan. Heat until butter is
melted. Remove garlic and stir in parsley. Pour butter mixture over
popcorn and toss lightly.
```

a) A document prepared for a SUN workstation

```
<!Include Declarations, //garbo/cal/desktop/recipe_template.doc>
<title>
Garlic Buttered Popcorn
<author>
by Mrs. J. H. Smith, Jr.
<paragraph>
Place butter and garlic in a small saucepan. Heat until butter is
melted. Remove garlic and stir in parsley. Pour butter mixture over
popcorn and toss lightly.
```

b) A document prepared for an Apollo workstation

Figure 1-3. Document with Include command

Appendix C shows what the file that is being included looks like. Because we used the <!Include Declarations, ...> *command, the property declarations in this file are now considered part of the externally-created document, but the text of the file is omitted.*

- Save the document and exit from the editor.
- If you did not create the document in your desktop directory, copy it to your desktop.
- Open your desktop. The document icon is ready to be opened.

The opened document will look like the one in Figure 1-4. The ASCII markup for the newly formed document is in Appendix C.

Garlic Buttered Popcorn

by Mrs. J. H. Smith, Jr.

Place butter and garlic in a small saucepan. Heat until butter is melted. Remove garlic and stir in parsley. Pour butter mixture over popcorn and toss lightly.

Figure 1-4. Opened document

- When you close the document you will be asked if you want to save it.

  When you select **Save** *on the Close stickup, the document will be saved as a regular (fast) publishing software document and will appear on your desktop as a document icon. If you want to be able to edit the document outside the publishing software,* **Save ASCII** *before you close the document. The ASCII format version of the document will also appear on your desktop as a document icon, but the document will be readable by other software.*

Include Command Shortcut

If you create a folder on your desktop for ASCII templates, you will be able to use the templates easily and efficiently, because you will not need to type the entire pathname of a template file, or its declarations, when you want to include it in another document.

For the previous example, if *recipe_template.doc* were in *ascii_templates.fdr* on your desktop, you could use a shortcut to include the declarations from *recipe_template.doc* in the Garlic Buttered Popcorn document. Compare the first line of Figure 1-5 to the first line of Figure 1-3.

```
<!Include Declarations, ascii_templates.fdr/recipe_template.doc>
<title>
Garlic Buttered Popcorn
<author>
by Mrs. J. H. Smith, Jr.
<paragraph>
Place butter and garlic in a small saucepan. Heat until butter is
melted. Remove garlic and stir in parsley. Pour butter mixture over
popcorn and toss lightly.
```

Figure 1-5. Include command with abbreviated pathname

Details about the Include Commands

Files being included should have the same ASCII Format version number as the file in which they are being included. To ensure that this is true, open each template on your desktop and save it in ASCII format with the latest version.

Even though the file that you open on your desktop may not itself start with the line *<!OPS, ...>*, the combination of files is treated as being marked up, since the *<!Include, ...>* line contained the necessary <! characters to signal a marked-up file.

If you use the declaration *<!Class Defaults, Fill = yes/no/on/off/blank>*, you may put it either in the document to be included or in the external document. We recommend putting it in the document to be included, as in Appendix C. This declaration must go immediately after *<!OPS ...>* and before all other declarations. See *Special Loading Features for Documents Created Outside the Interleaf Desktop* for more about *Fill*.

The parameter *<!Class Defaults, Fill = ...>* is not restricted to Include files and can be used in any ASCII Format document.

When a document with an Include command in it is opened on the desktop and then saved, the information from the two files is combined into the same file. This means that changes you make to the included file will not be reflected in the document.

You may want to include other files within the file you are inserting. If you want to do this, the Include commands can be nested up to four levels deep.

Remember that all declarations resulting from the use of an **Include** command (markup starting with <!) must precede all commands. Remember also that the order of document, page, and font declarations must be correct. See *Specifications for Interleaf ASCII Format*.

Special Loading Features for Documents Created Outside the Interleaf Desktop

To facilitate opening an external document for the first time, the ASCII loader has some special features for documents created outside the Interleaf publishing software desktop. These features never appear in documents saved from the desktop.

- *Comments* may be inserted in the markup that will not appear when the document is displayed using the publishing software. These comments are eliminated completely during loading (opening) of the document. See the *Specifications for Interleaf ASCII Format* for information on inserting comments.
- An alternative way to change fonts is to use the command </F>. When you use this command the font reverts to the one in use previously. Note that you use this form only on input; when you save a document, </F> is converted to a numbered font. This command is never output when a document is saved. Sixteen levels are allowed. The stack of fonts is reset at the beginning of each component; it is not possible to revert to a font used in an earlier component.
- Another shortcut: The commands <*FI*> and <*FB*> will switch to the italic or bold versions of the current font, if bold or italic versions exist. These commands are generally terminated with </F>. They are never output when a document is saved. Note that when these commands are used it is *not* necessary to declare the resulting bold or italic fonts in the <*!Font Definitions, ...*> declaration.
- The component property <...*Fill = off* ...> will cause a hard return to appear at the end of each line in the component. (If a line already ends <*HR*>, then you will get two hard returns.) This is useful for components containing tabular material.
- The component property <...*Fill = blank* ...> will cause a blank to be inserted at the end of each input line if no blank was there and if it is not the end of a component. This is useful when documents are prepared on certain word processors or with editors that do not ensure blanks at the ends of lines. When line breaks on the word processor differ from the ones in the publishing software, <...*Fill = blank* ...> ensures that words do not run together. Usually, it is easiest to put this in a <*!Class Defaults, ...*> declaration so that it will apply to all component classes.

Non-Marked-Up Files

Interleaf publishing software can also handle non-marked-up (ordinary ASCII) files that are the output of other programs or editors. If a file is read in and it does not start with the characters <!, the following rules apply:

- Default settings are used. See *Specifications for Interleaf ASCII Format.*
- *Paragraph* components are generated. Whenever there are one or more blank lines in the incoming file, a new *paragraph* component is generated.
- Access to a set of six master frames and two autonumber streams is provided.
- A hard return is placed at the end of each line.
- The character "<" has no special meaning; it should not be doubled.

Desktop Document Icons

The icons on the desktop are representations of directory and file extensions in the operating system. To avoid confusion, you should not give external documents extensions that are used for directories and non-documents by the publishing software. For a list of these extensions, see the chapter *Desktop Manager,* in the *Reference Manual,* Volume 1.

However, when you mark up a document, you may want to give it a name with the extension *.doc*, so that it will appear on your desktop as an Interleaf document icon. A document that has the *.doc* extension looks like a blank sheet of paper with the upper right-hand corner turned down on your desktop.

A document icon on your desktop that looks like a piece of computer paper indicates that the extension of the document name in the operating system (if it has one) is not *.doc*. When you open such a document on your desktop and then save it with the publishing software, the *.doc* extension is added to its the operating system name, and the document is represented by an Interleaf document icon.

The icon that looks like a piece of computer paper is still on your desktop as well. In many cases, once you have saved the document with the publishing software, you will want to cut the icon that looks like computer paper since it is no longer necessary to keep it.

When deciding what type of icon to display, the desktop does not look at the contents of a file to determine if it was saved as a fast or ASCII document: this is checked only when the icon is opened. Therefore, if you want to know which of your documents are non-marked-up files and which have been marked up or saved with the publishing software, do not use the *.doc* extension when you name non-marked-up files.

Hints on Writing Translation Programs

If you are an engineer who is going to write a translation program to convert output from a word processor to Interleaf ASCII Format, for example, there are some considerations that might not be obvious at first.

Consider the following paragraph of text, as produced by a hypothetical word processor. It is indented approximately 3 characters from the left margin, and the first line is indented an additional 3 characters:

```
   Once upon a time there were three bears, a papa bear,
a mother and a baby bear, and they lived in the forest.
One day, papa bear said, "I think we should go out and buy
a VCR." Mama bear said, "I hear that VHS is the most...
```

Assume that the actual stream of characters produced by the hypothetical word processor looks like the following paragraph in which {*NL*} means linefeed and {*Tab*} means a tab character.

```
{Tab}{Tab}Once upon a time there were three bears, a papa bear,{NL}{Tab}a
mother bear, and a baby bear, and they lived in the forest.{NL}{Tab}One day,
papa bear said, "I think we should go out and buy{NL}{Tab}a VCR." Mama bear
said, "I hear that VHS is the most...
```

Notice that there is no blank at the end of each line (before {NL}) and that each line is indented by the use of a tab. (The typical word processor may or may not have previously specified where the tab stops should be.) You would like the translation program to convert this example to an Interleaf ASCII Format file something like this (it is assumed that declarations have been included at the beginning and that the component name is *paragraph*):

```
<!declarations, ...>
<paragraph, Left Margin = 0.3 inches, First Indent = 0.3 inches>{NL}Once
upon a time there were three bears, a papa bear, {NL}a mother bear, and a
baby bear, and they lived in the forest. {NL}One day, papa bear said, "I
think we should go out and buy {NL}a VCR." Mama bear said, "I hear that VHS
is the most...
```

In this example, the tabs have been removed and blanks have been added at the ends of lines (the *Fill = blank* option could have been used instead of adding blanks. See the section *Special Loading for Documents Created Outside the Interleaf Desktop)*. The *newlines* are not necessary; they have been left for convenience.

The translation program looks at the second and succeeding lines of the paragraph and derives the *Left Margin* value. Then it compares the indentation of the first line to that of the following lines and creates the *First Indent* value.

There are many other considerations when you write a translation program. For example, a different hypothetical word processor might have used multiple blanks to generate the indentation, rather than tabs. Hanging indents would add additional complexity to the analysis and translation. The program *opsfilt*, available from Interleaf, resolves these and other aspects of converting documents. See the chapter on *opsfilt* in the manual *File Transfer*.

Specifications for Interleaf ASCII Format

This section provides details of the specifications needed to provide the markup for an ASCII document.

Quoting Rules

When you use certain characters a) in the names of components, frames, autonumber streams, or printers, b) in page number entries with prefixes, c) in autonumber entries with prefixes or suffixes d) in index headings, e) in file names, f) in **Include** commands, or g) in the text of old-style headers and footers, you must enclose the entire entry in quotation marks.

The characters that require that you use quotation marks around an entry are: spaces (including thin, em, and en spaces), commas, minus signs, en dashes, em dashes, small bullets, =, <, >, @, ", ´, !, #, |, {, }, ~, \, ^, `, £, §, “, ¢, and any other character that has a hex value of less than 0x20 or more than 0x7e.

Even when you quote entries, the results may not be exactly what you expect when you open a document on your desktop, so we recommend that you use discretion when you choose names. For example, if you named one component class " " and another " ", you would not be able to distinguish between them on popup menus.

Embedded quotations (quotation marks within quotation marks) must be doubled. For example, if you want a component name to be:

Part "A"

you must enter:

"Part ""A"""

If you were using old-style headers and footers for a single-sided document, and you wanted this header:

Just “Hello”

you would type this:

```
<Page Header,
      Left Text = "Just <#7f>Hello""",
      Font = F66>
```

If a name includes such strings as *tab*, *F0*, *HR* — strings that are the same as commands or font names — the name must also be surrounded by quotation marks.

Since there is no harm in using quotation marks when they are not necessary, the safest thing to do is use them whenever you are in doubt.

Naming Conventions

We recommend that you use only alphanumeric characters when you name components, frames, autonumber streams, tags, TOC documents, etc. If you do use other characters, the results when you open the document on your desktop may be confusing. See *Quoting Rules* for guidelines on using non-alphanumeric characters.

Basic Syntax Rules

Order of Declarations and Commands

All declarations (markup starting with <!) must precede all content (commands and text). The order of the declarations and commands should generally be as follows because of the likely dependence of one declaration on another:

1. *<!OPS, Version = ...>* is mandatory when there are no other declarations and optional when there are other declarations. If it is used, it must appear first.
2. *<!Document, ...>* and *<!Page, ...>* must follow *<!OPS, Version = ...>*, if it exists, and precede any other declarations or commands.
3. *<!Font Definitions, ...>* must appear after *<!Document, ...>* and *<!Page, ...>* and before other declarations. The legality of fonts depends of the printer type, which is declared in the *<!Document, ...>* declaration. The declarations that follow depend on the font definitions.
4. *<!Autonumber Stream, ...>* declarations should appear before *<!Class Defaults, ...>* and *<!Class, ...>* declarations in which they may be mentioned.
5. *<!Class Defaults, ...>* (if it is used) must appear after *<!Font Definitions, ...>* and before *<!Class, ...>* declarations.
6. *<!Class, ...>* must appear after *<!Font Definitions, ...>* and *<!Class Defaults, ...>* (if it is used).
7. *<!Master Frame, ...>* declarations should appear after *<!Class Defaults, ...>* and *<!Class, ...>* declarations since a master frame may contain components.
8. Commands such as *<Page Header, Frame = ...>*, *<Page Footer, Frame = ...>*, and *<component name>* appear after the declarations.

Within the Markup Brackets

Within the markup brackets <...>, tabs, blanks, and newlines may be used freely, as long as you stay within the quoting rules (see page 1-26). Note, however, that these are not saved when you save the document in ASCII on the desktop.

Within the markup brackets, a comma must be used after each command title that is followed by pertinent information. In addition, if one property value is followed by another, they must be separated by commas. For example:

```
<!Class, ack,
            Bottom Margin =          0.15 inches,
            Line Spacing =           1 lines,
            Font =                   F68>
```

Wherever you are allowed to have white space, comments may be inserted by surrounding them with two minus signs on either side --*thusly*--. For example:

```
<!Class, ack, --this name is short for acknowledgment--
            Bottom Margin =          0.15 inches,
            Line Spacing =           1 lines,
            Font =                   F68>
```

There must be no white space between a pair of minus signs: --this will work--, and -- so will this--, but - -this won't--.

Such comments cannot be used within diagramming or chart data.

Newlines

When you create a *newline* on your computer or word processor, it can be a linefeed, a carriage return, or both. Within marked-up text, these single newlines are ignored totally. The ASCII dumper creates its own newlines since it always creates lines that are a set number of characters long. A line of text from the ASCII dumper will usually end with a blank (space) before the newline.

Two newlines in a row (one blank line) will cause a new component to start. Multiple blank lines in a row are treated the same as one blank line.

When you save a file in Interleaf ASCII Format, it is possible that a newline will be inserted occasionally when a very long word falls at the end of a line. Thus, the word may be broken onto two lines.

In general, the line breaks you see when you open a document on your desktop will not match the line breaks you see when you examine the document with another editor.

When a file that is not marked up is opened, a single newline is converted to a hard return. Multiple newlines cause a new component to start, with no hard return at the end of the previous component.

Lines that *appear* blank, but in fact contain spaces or tabs, are not considered blank lines.

Markup within the Text

Commands within text are placed at the specific location where they are to take effect. For example, a component might look like this in ASCII Format:

```
<ack>
 (<#7f>In Search of the Elusive Pingo" (Ideas and Trends), <F70>The
New York Times, <F0>5 May 1974. <F5>E <F0>1974 by The New York Times
Company. Used by permission.)
```

When the document is opened on the desktop, the component looks like this:

("In Search of the Elusive Pingo" (Ideas and Trends), *The New York Times,* 5 May 1974. © 1974 by The New York Times Company. Used by permission.)

Within text that is marked up, to insert the character "<", you must double the character (<<).

A tab (Hex 09) in the input is treated as *<Tab>*.

Units of Measurement

You can use any of the following units of measurement when you enter markup. However, when a document is opened on your desktop, the measurements will be converted to inches on all property sheets. When the document is saved, the measurements will always be saved in inches.

cm	mm	
in	inch	inches
pica	picas	
point	points	
didot	didots	
cicero	ciceros	

Declaration and Command Specifications

The following list shows the declarations and commands you may use and the values you may enter after each declaration or command. The following may be used interchangeably in markup: *true*, *yes*, and *on* and *false*, *no*, and *off*.

In the list, possible values are indicated in the following way:

- *x* (as in *Top Margin = x inches*) indicates that numbers can contain decimals.
- *n* (as in *Fn = font name*) indicates that only whole numbers are allowed.

- *m* (as in *Fm = font name*) indicates that this number must be different from the one preceding it.
- *name* (as in *Final Output Device = "name"*) indicates that the user should enter a name. See the *Specifications* section for some naming guidelines.
- *string* (as in *Page # Prefix = "string"*) indicates that the user can insert a string or leave the entry blank. *Strings*, unlike *names*, will often contain special characters. See the section *Quoting Rules* on page NO TAG to determine when it is necessary to enclose a *string* entry in quotation marks.
- Characters separated by / (as in *New Page = yes/no/on/off*) are the alternatives you can enter. An example is, *New Page = yes*. The exception to this is *tabs*. For tabs, a / following a number (*Left Tab = 0/1/2/3/4/5/6/7 inches*) indicates that the next number will be the next tab stop.

Declarations

<!OPS, ...>

```
<!OPS, Version = n.n>
```

This declaration should always start the file. If *Version* is omitted, then the latest version is assumed. For example, the current ASCII version is 5.2. The version number could eventually be 10.9 or 11.0, but not 10.10.

<!Document, ...>

```
<!Document,
      Header Page =            on/off,
      Double Sided =           yes/no,
      Manual Sheet Feed =      yes/no,
      Print Rev Bars =         yes/no,
      Print Strikes =          yes/no,
      Print Underlines =       yes/no,
      Final Output Device =    "name",
      Default Printer =        "name">
```

<!Page, ...>

```
<!Page,
        Orientation =              Portrait/Landscape,
        Columns =                  n,
        Gutter =                   x inches,
        Height =                   x inches,
        Width =                    x inches
        Top Margin =               x inches,
        Bottom Margin =            x inches,
        Left Margin =              x inches,
        Right Margin =             x inches,
        First Page =               Left/Right,
        Bleed =                    yes/no,
        Starting Page # =          Inherit/n,
        Page # Prefix =            "string",
        Page # Style =             Arabic/Lowercase Roman/
                                   Uppercase Roman,
        Hyphenation =              on/off,
        Consecutive Hyphens =      Any/1/2/3/4,
        Revision Bar Placement =Left/Automatic,
        Vert. Just. =              on/off,
        Balance Columns =          on/off,
        Margin Stretch =           n,
        Margin Shrink =            n,
        Frame Margin Shrink =      n,
        Feathering =               on/off,
        Maximum Feathering =       n,
        Vert. Just. Pages =        on/off,
        Depth At Page Break =      n,
        Depth No Page Break =      n,
        Revision Bar Placement =left>
```

Together, page and component margins must allow at least one-half inch of space horizontally and vertically for text per column, except in microdocuments.

If *First Page* is specified, then alternating page headers and footers (double-sided document) are assumed.

The values for *Margin Stretch*, *Margin Shrink*, *Frame Margin Shrink*, *Maximum Feathering*, *Depth At Page Break*, and *Depth No Page Break* are percentages.

<!Autonumber Stream, ...>

```
<!Autonumber Stream, "name", n,
        Level n Symbol Type =       ARABIC/UPPER ROMAN/
                                    LOWER ROMAN/UPPER ALPHA/
                                    LOWER ALPHA,
        Level n Prefix =            "string",
        Level n Suffix =            "string",
        Level n Starting Value =    n,
        Level n Last Only =         yes/no,
        Level n Show =              yes/no>
```

The *n* in the first line refers to the number of levels in the autonumber stream (1 to 8). If this value is not defined, the software assumes 8 levels. The other items, for each number through *n*, need to be declared only if they are different from the default values.

<!Font Definitions, ...> or <!Fonts, ...>

<!Font Definitions and *<!Fonts* are synonymous.

```
<!Font Definitions,
        Fn =                        font name,
        Fm =                        font name,
                                    ...>
```

Font names may be specified flexibly. For example, *Classic 10 Bold* and *bold classic 10* are equivalent. Note that *F0* cannot be specified here, since it refers to the default font of a component. If you ask for a font that does not exist, an attempt is made to select the closest match.

<!Class Defaults, ...>

The *<!Class Defaults, ...>* declaration has the same fields as the *<!Class, ...>* declaration, except that tabs cannot be specified.

<!Class, ...>

```
<!Class, component name,
        Top Margin =                x inches,
        Bottom Margin =             x inches,
        Left Margin =               x inches,
        Right Margin =              x inches,
        First Indent =              x inches,
        Line Spacing =              x lines,
        Alignment =                 Left/Right/Center/Both,
        Font =                      Fn,
        Widow Control =             1-15,
        Orphan Control =            1-15,
        Allow Page Break After =    yes/no/on/off,
        Allow Page Break Within =   yes/no/on/off,
        Fill =                      yes/no/on/off/blank,
        Straddle =                  yes/no,
        New Page =                  yes/no/on/off,
        Hyphenation =               on/off/normal/0-10,
        Autonumber Name =           "name",
        Autonumber Level =          n,
        TOC Doc Name                "name",
        Left Tab =                  see Tabs below>
```

Component name can be up to 9 characters long. We recommend that you use only alphanumeric characters. See *Quoting Rules* for guidelines on using non-alphanumeric characters.

When a document is opened on the desktop, if a name indicated in a command does not match a class exactly, a loose match will be tried allowing capital and small letters to match and allowing for extra or missing blanks. For example, *Paragraph* and *paragraph* will be matched. This allows for some inconsistencies and typos.

Together, page and component margins must allow at least one-half inch of space for text per column, except in microdocuments. *Margin* settings and *indents* must be consistent. For example, negative indents must not overlap page edge nor positive indents exceed right margin.

Inches are used in this example. See the section *Units Of Measurement* for a list of other acceptable units.

Line Spacing must be no smaller than *1.0* lines and no larger than *5.0* lines.

Font is usually declared for every class.

Tabs look like this (all examples set tabs at 1 inch increments):

```
Left Tab =                    0/1/2/3/4/5/6/7 inches,
Right Tab =                   0/1*7 inches,
Center Tab =                  1*7 inches,
Decimal Tab =                 1*6/8 inches,
```

- The /'s are necessary to delineate one setting from the next.
- A * after a number in a tab setting indicates evenly spaced increments. The number immediately preceding the * is the amount of the increment and the number after the * is the number of increments. For example, if you set Left Tab = *1/.5*3* inches, tabs start at *1* and then *.5* is added to the previous setting three times. This gives you left tabs at 1/1.5/2/2.5 inches.
- If there is nothing in front of the number preceding the *, the first tab is set at *0*. For example, if you set Left Tab = *.5*3* inches, the first tab is *0*. Then, *.5* is added to the previous setting three times. The settings are at 0/.5/1/1.5 inches.
- If there are are no tabs in the markup, the 30 default settings are used.
- If you do not want any tabs at all, use the entry *Left Tab* = , (the comma is a necessary part of the entry) either in a component class declaration or in the command for a particular component.

<!Master Frame, ...>

```
<!Master Frame,
      Name =                  "name",
      Placement =             At Anchor/Bottom/
                              Following Anchor/
                              Following Text/Footnote/Top,
      Vertical Alignment =    x inches,
      Horizontal Alignment =  x inches/Left/Right/Center,
      Height =                x inches,
      Width =                 x inches,
      Shared Contents =       yes/no,
      Same Page =             yes/no,
      Diagram =               ...>
```

See the chapter *ASCII Format for Diagramming Objects* for the markup that follows *Diagram* =.

If "*name*" has any spaces in it, it must have quotation marks around it (see *Quoting Rules* on page 1-26). *Same Page* is synonymous with *Same Column*.

Vertical Alignment applies only when *Placement = At Anchor*. *Horizontal Alignment* applies only when *Placement* is not *At Anchor*.

<!Comment, ...>

With *<!Comment, ...* everything up to the next > is completely ignored. If you have embedded matching <...> (a set of brackets within *<!Comment, ...>*), they will be treated as part of the comment. Note that << is not the same as < <.

Commands

<Autonum, ...>

<Autonum, "name", n> is the basic form of the autonumber command. *"Name"* is the name of the autonumber stream, and *n* is a number between 1 and 8 indicating the level of the autonumber in the command.

Between *n* and >, there may be other entries: *First = Yes* is an optional entry if the autonumber is the first one in the stream; *Restart = Yes* is a required entry if a stream is to be restarted (only admissible in the command for a level 1 autonumber); and *Tagname = "tagname"* is a required entry if the autonumber is being used for autoreferencing. See the section *Autonumbers and Autoreferences* on page 1-12 for details.

<component, ...>

The *<component, ...>* properties are the same as the *<!Class, ...>* properties.

<Frame, ...>

The *<Frame, ...>* properties are the same as the *<!Master Frame, ...>*properties. Note that the defaults are not taken from the master; see *Master Frame Declarations* on page 1-10.

<Fn>

<Fn> represents a font change. See *Markup Within the Text of the Component* on page 1-10.

</F>

</F> represents a font change to the previous font. See *Special Loading Features for Documents Created Outside the Interleaf Desktop* on page 1-23.

<FI>

<FI> represents a font change to italic. See *Special Loading Features for Documents Created Outside the Interleaf Desktop* on page 1-23.

<FB>

<FB> represents a font change to bold. See *Special Loading Features for Documents Created Outside the Interleaf Desktop* on page 1-23.

<FJ>

<FJ> represents a force justify return.

<HR>

<HR> represents a hard return.

<Index, ...>

<Index, "*Heading*"> is the basic format for an index token. Other markup can be used between "*Heading*" and >, including the range of text the index token covers.
Possible range entries:

This page =	default
To Next =	next component
Count =	*n*, (number of components included in range)
To Named =	*component name* (where reference ends)
See, See Also =	*entry name*
Sort, Sort String =	(customized sort string instructions)
Index =	*index name*
Typeface =	bold/italic

See the section *Index Tokens* on page 1-14 for details about this command.

<|, string>

<|, *string*> indicates a page break. *String* is the page number including the prefix if there is one. *String* will often be just a number. Roman numerals are preserved.

These page numbers appear only when you save a document in ASCII with the publishing software. They are ignored by the publishing software when the document is loaded.

<#xx>

<*#xx*> represents the hexadecimal value of a special character. The *xx* represents 2 or more hexadecimal digits. See *Interleaf Keyboard Mapping* on page 1-44 for the hexadecimal codes that can be used with the publishing software.

<Ref, ...>

<Ref, Auto #, Tag = tagname> or *<Ref, Page #, Tag = tagname>*, where *tagname* is from the *Autonum* entry, are the basic forms of autoreference commands. The first command makes the reference refer to the autonumber itself and the second makes it refer to the number of the page on which the autonumber appears. See the section *Autonumbers and Autoreferences* on page 1-12 for details.

<SP>

<SP> represents a "non-breakable" or "hard" space. In order to specify an em, en, or thin space, you must use the hex code (*<#xx>*) for the character.

<Tab ...>

The four tab commands cause the text to tab to the next tab stop. Depending on which tab command is used, the tab space will be left blank or filled with dashes (-), dots (.), or underscores (_).

<Tab> (-------->) (You may see an arrow on the screen to indicate a blank tab, but it will not appear in a printed document.)

<Tab-> (---------)

<Tab.> (.....)

<Tab_> (_____)

<Page Header, ...> or <Page Footer, ...>

If you do not put any header or footer information in your markup, the ASCII loader will use the defaults, and you can quickly add the text you want with the Interleaf software. This is by far the easiest way to handle headers and footers.

Since Release 2.5 of the publishing software, headers and footers have been in frames. As you can continue to use the old markup for headers and footers and let the software convert your headers and footers to frames, we include both ways.

Do not combine old style and new style headers and footers in a document. If you do, the old style headers and footers will be discarded when the document is loaded by the publishing software.

The markup for Release 3.0 (Version 5.2 of ASCII) :

<Page Header (Footer), Frame = V4, ...> indicates that the document is to be printed in single-sided format and that the first page has the same header or footer as the rest of the document.

<First Page Header (Footer), Frame = V4, ...> indicates that the first page has a different header or footer from the rest of the document.

<Right Page Header (Footer), Frame = V4, ...> indicates that the document is to be printed in one of the double-sided formats and that this is the header or footer for the right-hand pages.

<Left Page Header (Footer), Frame = V4, ...> indicates that the document is to be printed in one of the double-sided formats and that this is the header or footer for the left-hand pages.

For details about the markup in frames, see the next chapter *Interleaf ASCII Format for Diagramming Objects.*

The markup for Release 2.0 (Version 3.0 of ASCII):

<Page Designation,

Left Text =	"*text*",
Center Text =	"*text*",
Right Text =	"*text*",
First Page =	yes/no,
Font =	F*n* (*n* must be less than 100)>

If the layout of the document is single-sided, the *Page Designation* is either *Page Header* or *Page Footer.* If the layout is double-sided, the *Page Designation* is either *Right Page Header*, *Right Page Footer*, *Left Page Header*, or *Left Page Footer.*

The text of the footer or header must be enclosed in quotation marks. Any additional quotation marks within these must be doubled. See *Quoting Rules* on page 1-26 for details.

The text of the header or footer cannot contain any commands such as *<Fn>*, *<HR>*, *<SP>*, or *<Tab>*, etc. It can contain markup for special characters *<#xx>*. When a header or footer contains a hexadecimal code, the line with the code on it looks like this:

```
Right Text = "5 and 10<#1d> Store",
```

The tilde character within any header or footer will be replaced with the actual page number. For example, for a header or footer with *Page* followed by the page number, type:

```
<...Left Text = "Page ~", ...>
```

Default Values for Interleaf ASCII Format Documents (Markup Version 5.2)

If a value is not specified in an ASCII Format document, these defaults will be used. A few of the defaults listed here are not the same as the defaults you get when creating a document in the publishing software. For example, ASCII Format component defaults are *Top Margin = 0 inches* and *Bottom Margin = 0.14 inches*. The setting for *Top* and *Bottom Margins* is *0.07 inches* in the default document on your desktop. These differences occur because the ASCII defaults cannot always reflect changes to defaults without substantially changing aspects of documents created or edited with previous versions of the ASCII markup.

Document Defaults

Component Name =	paragraph
Header Page =	yes
Double-Sided =	no
Manual Sheet Feed =	no
Print Strikes =	yes
Print Rev Bars =	yes
Print Underlining =	yes

Page Defaults

Orientation =	Portrait
Columns =	1
Vert. Just. =	on
Height =	11 inches
Width =	8.5 inches
Top Margin =	1 inch
Bottom Margin =	1.1 inches
Left Margin =	1.4 inches
Right Margin =	1.4 inches
First Page =	*not set (indicates Single Sided printing)*
Bleed =	no
Starting Page # =	1
Page # Style =	Arabic
Hyphenation =	off
Consecutive Hyphens =	any
Balance Columns =	on
Margin Stretch =	200%
Margin Shrink =	50%
Frame Margin Shrink =	10%
Feathering =	on
Maximum Feathering =	8%

Vert. Just. Pages =	on
Depth At Page Break =	95%
Depth No Page Break =	90%
Starting Page # =	Inherit
Revision Bar Placement =	Left
Page # Prefix =	none

Autonumber Defaults

Symbol Type =	Arabic
Prefix =	none
Suffix =	.
Starting Value =	1
Last Only =	no
Show =	yes

Component Defaults

Top Margin =	0 inches
Bottom Margin =	0.14 inches
Left Margin =	0 inches
Right Margin =	0 inches
First Indent =	0 inches
Line Spacing =	1.31 lines
Alignment =	Both
Font =	Classic 10
Hyphenation =	5 *(normal)*
New Page =	no
Straddle =	no
Orphan Control =	2
Widow Control =	2
Allow Page Break Within =	yes
Allow Page Break After=	yes
Left Tab =	0, .75, 1.5, ... inches

Frame Defaults

Name =	"Following Text"
Placement =	Following Text
Horizontal Alignment =	Center
Vertical Alignment =	0 inches (for *At Anchor* placement)
Same Page (Column) =	no
Width =	2 inches
Height =	1 inch
Shared Contents =	no

Additional Information

Special Characters

To facilitate reading ASCII documents from various sources that are not in Interleaf ASCII Format, certain special characters are treated as follows in both marked-up and non-marked-up documents:

This section discusses characters represented by particular hex values; e.g., *08* is a **BACKSPACE**, rather than *<#08>*, which is character number 8 (**ESC**-h on the keyboard). For example, *<#08>* in the Symbols font is ⌧.

- Nulls (hex 00) are ignored. There is no message to indicate this.
- Backspaces (hex 08) followed by underscores are discarded, and a warning message is given, telling how many times the pair occurred. This eliminates getting hundreds of error messages when the loader encounters the underlining produced by some word processors.
- Other backspaces are discarded, and a warning message is given, telling how many occurred.
- Underscore/backspace and character/backspace/character (overprinting) are *not* treated specially.
- Tabs (hex 09) are treated the same as the *<Tab>* markup.
- Other special characters are ignored, and a warning message is given for each occurrence. Characters greater than hex 7f first have their high bit cleared. Then characters less than hex 20 or greater than hex 7e are changed to blanks.

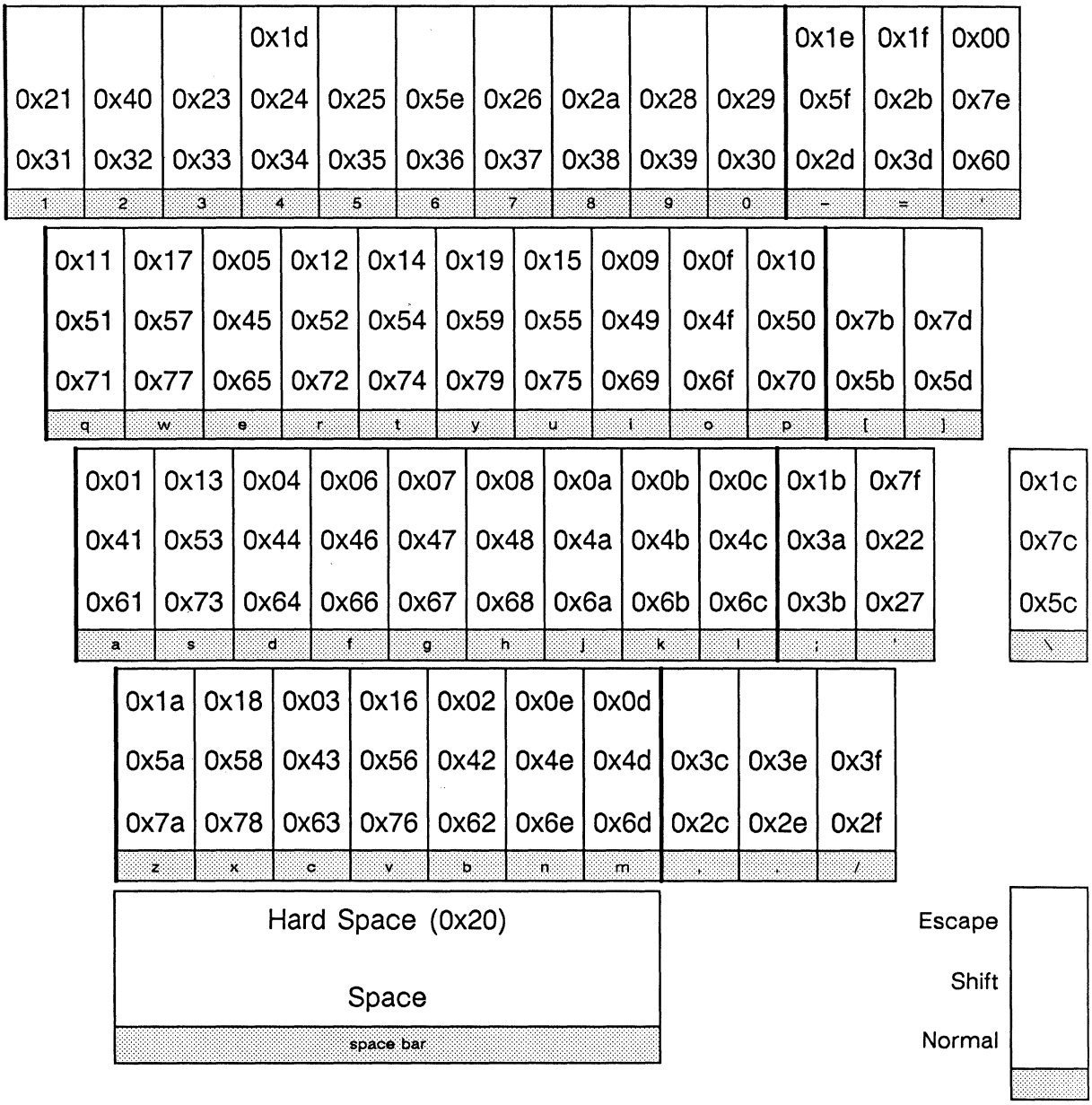


Figure 1-6. Hexadecimal codes for characters on publishing software keyboard

Interleaf Keyboard Mapping

Figure 1-6 shows hexadecimal codes used in markup that correspond to characters in all of the Interleaf publishing software fonts. Refer to the keyboard layouts in *The Keyboard* in the *Reference Manual,* Volume 1 to match up characters in the publishing software with their hexadecimal codes. For example, to enter the specifications for a ¢ sign, you need to find the ¢ sign on the keyboard layouts. Then type the code between brackets and preceded by a pound sign. The markup will look like this: `<#1d>`.

Interleaf ASCII Format Error Messages

The following list shows error messages that you may see as your document is being loaded and tells what the loader will do when these errors are encountered. The list is in alphabetical order.

If the loader encounters many errors, you will see a stickup menu asking if you really want to load the document. This stickup first appears after 25 error messages. If you ask the loader to continue loading the document, the stickup will appear after every 100 error messages.

Message	*Explanation*
A ">" was expected but not found.	Loader skips to next >.
A class of that name has already been defined.	New one ignored.
A header or footer is re-defined.	Old one used.
An "=" is missing.	Ignores everything until comma.
An illegal character was found. Hex value: *xx*.	High bit is cleared. If it then becomes a legal character, it is used. If it is still an illegal character, it is converted to a space.
An undefined font is referenced.	Undefined font ignored.
"Attach previous" is not allowed.	Nothing happens. Loading continues.
Autonumber Stream not defined.	Default autonumber stream is used.
Cannot open include file.	Nothing happens. Loading continues.
Class "..." has not been defined yet.	A default valued one is created. For example, this error is caused if the command *<foo>* occurs, and *foo* is neither a defined class nor a command. This warning will appear only once for each undefined component name.
Component name longer than 9 characters.	Name is truncated.
Discretionary hyphen was ignored (n times).	Document was saved with newer release.
Distance is too big.	Truncated to 36 inches. No value can be greater than 36 inches.

Error in Autonumber level.	If number is too large, it is truncated to 8 and Arabic is used. If the final > is is omitted, whatever has been defined up to the last level stands, and the last level takes its definition from the preceding level.
Error in loading a diagram.	The empty frame is loaded.
Expected "Left," "Right," "Center," or "Both".	Default value will be used.
Expected "Portrait" or "Landscape".	Portrait will be used.
Expected "Yes," "No," "On," or "Off".	Default value will be used.
First Indent is not consistent.	Truncated to fit. Negative indent overlapped page edge, or positive indent exceeded right margin. This error could be caused by inconsistent margin settings.
"First page" is incorrect.	Ignored.
Font number is too high.	Default used (maximum is currently 4095).
Frame has no diagram.	Empty frame is created.
Frame not allowed in microdocument.	Frame discarded. The rest of the document may be garbled as a result.
Horizontal alignment is wrong.	Horizontal Alignment may not be specified on "At Anchor" frames. Otherwise, "center" used.
"Hyphenation" is incorrect.	"Normal" assumed.
Incorrect placement of frame.	Placement ignored.
"Include Declarations" follows some non-declaratives.	Treated like *"Include"*.
Include file is not an ASCII file.	The file appears to be a binary file. File ignored.
Left/right margins are not consistent.	The component margins are set to zero. Page margins are too big if there is not at least one-half inch of room for text.

(Left/Right) page header/footer incorrect.	Right assumed.
Line Spacing is out of range.	Forced up to 1.0 or down to 10.0.
Line Spacing must be, e.g., 1.5 Lines.	Lines assumed. Word "lines" must be used for line spacing.
"Lines" is used incorrectly.	1/6 inch per line used. "Lines" was used incorrectly as unit of measurement.
Measurement must be > 0.0.	Default used.
Measurement must be >= 0.0.	Default used.
More than one diagram in frame.	All except the first diagram are discarded.
More than one Master Frame with the same name.	The properties of the first master frame definition are used as the master for frames with this name.
Multiple tab stops at same position.	First one used.
Negative margin goes off page.	Margin truncated to page margin.
No master frame.	Given only for shared content frames when the master frame is expected to have contents.
Old and new style headers and footers were mixed.	Old style headers and footers discarded.
"Page Number Type" is incorrect.	Arabic used.
Page size and margins are not consistent.	Page margin values are reset to the defaults. This error occurs if the page margins make the active part of the page less than one-half inch wide (or one inch high).
"Placement" must precede "Alignment", "Baseline" and "Same Page".	Default used.
Printer type is incorrect.	Default used.
"Shared Contents" used incorrectly.	Only a master frame can contain a description of Shared Contents.

Some text was seen before any component was named.	A default component generated.
Tab stops may not be set in Class Defaults.	These tabs ignored.
The closest font has been substituted.	For example: Classic 9 will be rounded down to Classic 8.
The document is not English language.	This message appears when you open or paste an ASCII document with "Language = ...". The loading will continue anyway.
The font number has been redefined.	Old definition used.
The format is, e.g., "FONT = F1".	Default used.
The format of a hex number is incorrect.	Value ignored.
The hex value is not correct.	Value ignored.
The name of the font is not recognized.	*<!Font Definitions,* ... default used.
The number specifying the measurement is missing.	Default value will be used.
The orphan value is incorrect (1 through 15).	Default used.
The point size is way out of range.	10 point assumed.
Starting Value incorrect.	Value is set to 1.
The string of characters seen is too long to be legal.	Generally, the truncated string is used and causes further errors.
The tab measurement is incorrect.	Whole line is ignored.
The unit of measurement is unknown (e.g., 1.5 inches).	Default value will be used.
The widow value is incorrect (1 through 15).	Default used.
The word "..." is incorrect.	Loader skips until comma or >. An unrecognizable word has been used.

There is no declaration named "...".	The string is passed into the document.
This declaration must precede all components.	Results are unpredictable.
Too many </F>.	Nothing happens. Font remains the same.
Too many tokens in document.	The frame (or whatever is associated with the token) is discarded.
Too many tab stops.	Extras ignored (maximum is 30).
Top/bottom margins are too big.	Both are set to zero. Component margins are too big if there is not at least one-half inch of room for text.
Unbalanced parentheses in diagram.	Diagram loaded but in error.
Unknown Autonumber Symbol Type.	Arabic used.
Use "Fn" with n = 1 or more.	Default used.
"Vertical Alignment" not allowed unless "At Anchor" frame.	The setting is ignored.

Chapter 2

ASCII Format for Diagramming Objects

This chapter specifies the markup language needed to introduce diagramming objects from external sources into the Interleaf 3.0 publishing software. A file written to these specifications can be loaded directly by the publishing software. Thereafter, the graphic objects within a file can be modified individually with the editing tools in the diagramming system. Consult this chapter when writing software interfaces to the Interleaf software.

Diagrams created for Release 2.5 documents are supported by the Release 3.0 software.

Overview

Diagrams are contained in frames. Diagrams usually consist of a collection of diagramming objects. From the user's point of view, an object is something that can be moved or modified without affecting other objects (except perhaps their visibility — one object may obscure another). Some examples of objects are ellipses, lines, and boxes. A box may be represented as a group of lines since a group is also an object.

Syntax

Diagramming objects must appear within a frame. Except for header/footer frames, the ASCII Format markup of most frames looks like this:

```
<Frame,
      Name =                    "Following Anchor",
      Placement =               Following Anchor,
      Horizontal Alignment =    Center,
      Width =                   6.0 inches,
      Height =                  5.0 inches,
      Page # =                  1,
      Diagram =
V4,
 (g9,0,0
 (E10,0,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
    6,6,0,0.066667,6))>
```

The line at the end of this example that begins with the letter **E** contains edit state information such as the grid spacing and the fill default. This information is generated automatically by the publishing software.

For more information on frame properties markup and other Interleaf ASCII Format markup, refer to the chapter *Interleaf ASCII Format for Text*.

On the screen, all diagramming objects in a frame are contained within a rectangular region. Coordinates of objects are given in inches relative to the upper left corner of the frame (+x to right, +y down).

Coordinates of diagramming objects are given in inches. Use no more than 9 digits to the left of the decimal point and 6 to the right. Because the internal floating point precision is roughly 9 digits, we recommend that you use no more than a total of 9 significant digits (left *and* right) to represent each coordinate.

There are two kinds of text objects in frames, text strings and microdocuments. Text strings and the diagramming markup for microdocuments are discussed in this chapter in the section *Text*. The text of microdocuments is handled like the text of any other document. See the previous chapter *Interleaf ASCII Format for Text* for instructions on how to mark up text.

Since the ASCII loader ignores white space (spaces, tabs) and newlines within a diagram except when they occur in microdocuments, you should format your file in such a way that you can read and edit it easily. To see an example of a readable format, save in ASCII any file on your desktop that contains diagramming information, and then examine this file with a non-Interleaf editor.

The following characters have special meanings in Interleaf ASCII Format and must be quoted if they are used:

```
<  >  (  )  ,  \
```

If you wish to use one of these characters in a text string, precede it with a backslash character (\). Also use a backslash to quote any spaces you wish to enter in a text string. For example, if you want to use the less-than symbol followed by a space and a comma, type:

```
\<\ \,
```

Pay special attention to version numbers. In addition to a per frame version number, there is one for each object type. If you use incorrect version numbers, the results are unpredictable. Also pay attention to upper and lower case when indicating the type of object. For instance, a spline is indicated by a capital **S**, not a lowercase **s**.

General Format Descriptions

Following the frame property specifications, information for each diagram begins with the letter **V**, a diagram version number, **4**, and a comma: **V4,**. This is followed by the diagramming objects.

All of the objects in a frame are considered part of a group. Therefore, every frame *must* begin with a group object. All of the objects in the diagram are contained within this top-level group object. These objects may be other groups. The usual ASCII Format description for a frame is:

```
<Frame, ...
V4,
  (g9,1,0
    (... object1 ...)
    (... object2 ...)
     ...
    (... objectN ...))>
```

Format for Diagramming Objects

A diagramming object such as **(g9,1,0 ...)** from the previous example, consists of a string of characters generally described this way in ASCII Format:

- left parenthesis
- letter indicating object type (e.g., **g** for group)
- object version number
- comma
- z coordinate (an integer)
- comma
- flags field (16-bit — unsigned, written in decimal)
- a series of other fields (depending on the object type)
- right parenthesis

Valid object types are:

line	group	text	polygon	ellipse
image	spline	plotter	chart	arc
tracing				

The object type letters, version numbers, and other fields will be discussed with each type.

The z coordinate is used only for layering and should start with one (1) and be increased for each new graphics object in the diagram. Filled objects with higher z values obscure those with lower ones when they share space on the screen and on the printed page.

The flags field consists mostly of attribute lock bits. A zero (**0**) will suffice in most cases, but the bit assignments are given in the *Flags* section of this chapter for those who require more information.

Values and markup for other fields such as line and fill textures, line patterns, and fonts are found in the section *Field Specifications*, beginning on page 2-14.

Lines

The letter **v** (for vector) represents a line segment; the version number is **4**. (Note: this is line version 4 and should not be confused with the global diagram version, V4, mentioned earlier.) The z coordinate and flags fields follow **v4**. Next come the coordinates of the line segment end-points and integers representing line texture, line width and a line pattern. Here is a sample line segment:

```
(v4,1,0,0.7,1.3,2.1,1.3,17,1,0)
```

where

Value	Represents
v4	object type: line (version 4)
1	z coordinate
0	flags
0.7, 1.3	x, y coordinates of the beginning (in inches)
2.1, 1.3	x, y coordinates of the end (in inches)
17	line texture (black)
1	line width (1 pixel)
0	line pattern (solid)

Groups

A group is a collection of objects that is treated as a single unit. Within the publishing software, selecting one edge of the group selects all of them and subsequent changes apply to the whole group.

The letter **g** represents a group and the version number is **9**. The z coordinate and flags follow **g9**. The rest of the group consists of a list of objects. For example, a group of four line segments might look like this:

```
(g9,6,0
  (v4,6,0,1.0,2.0,1.5,2.0,17,1,0)
  (v4,7,0,0.6,1.7,0.3,2.1,17,1,0)
  (v4,8,0,1.4,1.0,1.9,0.5,17,1,0)
  (v4,9,0,0.3,2.4,1.0,0.7,17,1,0))
```

Note that the first line segment has the same z coordinate as the group object, but that it is increased by one for the others.

As mentioned earlier, all of the diagramming objects within a frame are considered part of a group. For example, a single line segment might look like this:

```
<Frame ...
V4,
(g9,1,0
  (v4,1,0,1.0,2.2,0.6,1.8,17,1,0))>
```

A single line segment and a group of three others might look like this:

```
<Frame ...
V4,
(g9,1,0
  (v4,1,0,1.0,2.2,0.6,1.8,17,1,0)
  (g9,2,0
    (v4,2,0,0.6,1.7,0.3,2.1,17,1,0)
    (v4,3,0,1.4,1.0,1.9,0.5,17,1,0)
    (v4,4,0,0.3,2.4,1.0,0.7,17,1,0)))>
```

Polys

A poly object is a group of paths. The letter **p** specifies a poly object, and the version number is 7. After the z coordinate and flags fields, there is a texture field followed by a group object. Here is a simple poly, a box:

```
(p7,5,0,0
  (g9,5,0
    (g9,5,0
      (v4,5,0,1.0,2.0,1.0,3.0,17,1,0)
      (v4,6,0,1.0,3.0,4.0,3.0,17,1,0)
      (v4,7,0,4.0,3.0,4.0,2.0,17,1,0)
      (v4,8,0,4.0,2.0,1.0,2.0,17,1,0)))
```

where

Value	Represents
p7	object type: polygon (version 7)
5	z coordinate
0	flags
0	texture (none)
g9...	a group of 4 line segments making a 3 by 1 inch box

Here is the markup for a more complex poly, a box with an ellipse inside:

```
(p7,2,0,0
  (g9,2,0
    (e5,2,0,0.8,1,1,1,0.9,1.6,0,17,1,0)
    (p7,4,8,0
      (g9,4,0
        (g9,4,0
          (v4,4,0,0.8,1,1,1,17,1,0)
          (v4,5,0,1,1,1,2,17,1,0)
          (v4,6,0,1,2,0.8,2,17,1,0)
          (v4,7,0,0.8,2,0.8,1,17,1,0))))))
```

It is critical that the end of the first element (second x, y pair) be the same as the beginning of the second element (first x, y pair), and the end of the second matches the beginning of the third, and so on.

Ellipses

An ellipse is defined by specifying a bounding parallelogram. In Figure 2-1, the parallelogram, and consequently the ellipse, is uniquely defined by giving the coordinates of A followed by those of B and C.

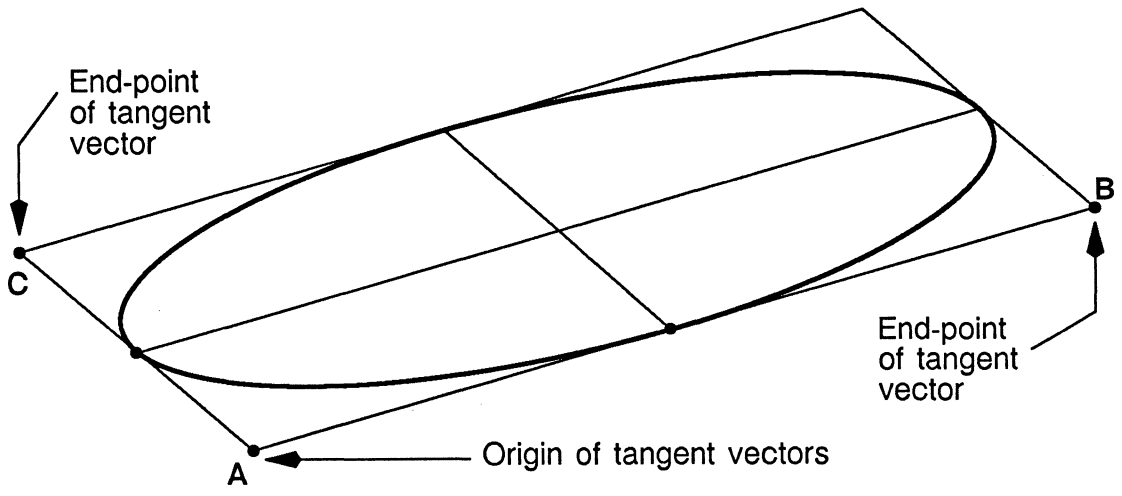


Figure 2-1. An ellipse

The letter **e** represents an ellipse object and the version number is **5**. After **e5** come the z coordinate and flags fields. Next come the x and y coordinates (in inches) of the origin of the tangent vectors, followed by the coordinates (in inches) of the two tangent vector end-points. Next is a texture field for the ellipse

interior, followed by texture, width, and pattern border fields that are analogous to the same fields in the line format. Here is a sample ellipse:

(e5,16,0,0.4,2.1,2.0,1.1,0.3,1.0,13,17,1,0)

where

Value	Represents
e5	object type: ellipse (version 5)
16	z coordinate
0	flags
0.4, 2.1	x, y coordinates of origin of tangent vectors (in inches)
2.0, 1.1	x, y coordinates of tangent vector end-point (in inches)
0.3, 1.0	x, y coordinates of tangent vector end-point (in inches)
13	interior texture (white)
17	border texture (black)
1	border width (1 pixel)
0	border pattern (solid)

Arcs

The Interleaf software supports the three types of conic arcs:

- elliptic
- parabolic
- hyperbolic

A six-point model for the arcs is used:

- *point 1* is the tangent at the beginning
- *point 2* is the beginning point
- *point 3* is a point on the arc between *point 2* and *point 4*
- *point 4* is a point on the arc (usually the end-point)
- *point 5* is the tangent at *point 4*
- *point 6* is the end-point

In the case of hyperbolic arcs, make sure that all the points are on the same half of the hyperbola.

If an elliptic arc is greater than 180°, the angular separation of the tangent vectors cannot be less than atan(2), or approximately 63°. For closed, or almost closed, arcs, *point 4* will be different from *point 6* to satisfy this condition.

In Figure 2-2, *points 4* and *6* are the same on the non-extended arc and distinct on the extended arc.

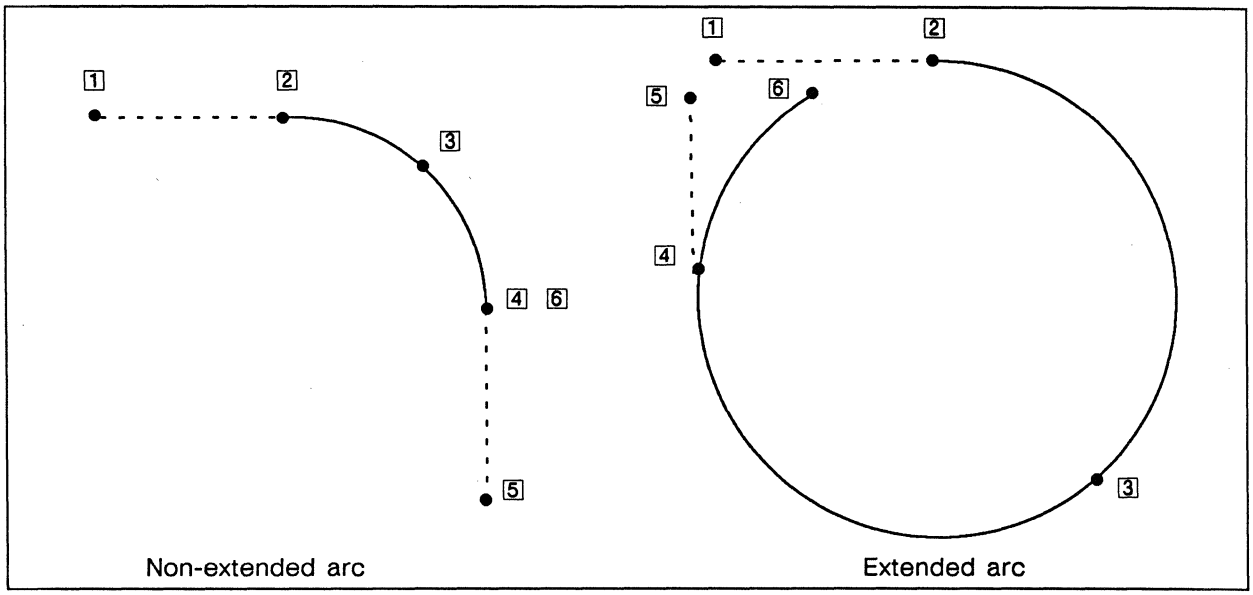


Figure 2-2. Elliptic arcs

The letter **a** represents an arc and the version number is **5**. The **z** coordinate and flags follow **a5**. The number **1** is the format; **1** is the only valid value. The next six pairs are the values in inches of the x, y coordinates of the six points of the arc. The last four entries are integers representing interior texture, border texture, border width, and border pattern.

The following markup describes a non-extended elliptic arc:

(a5,1,0,1,0.5,1,1.5,1,2.207107,1.292894,2.5,2,2.5,3.0,2.5,2,0,17,1,0)

where

Value	Represents
a5	object type: arc (version 5)
1	z coordinate
0	flags
1	format (Interleaf)
0.5,1	x,y coordinates of point 1 (tangent a beginning point)
1.5,1	x,y coordinates of point 2 (beginning point)
2.207107, 1.292894	x,y coordinates of point 3 (point between point 2 and point 4)
2.5,2	x,y coordinates of point 4 (end-point)
2.5,3.0	x,y coordinates of point 5 (tangent at point 4)
2.5,2	x,y coordinates of point 6 (same as point 4)
0	interior texture (none)
17	border texture (black)
1	border width (1 pixel)
0	border pattern (solid)

The following markup describes an extended elliptic arc:

**(a5,1,0,1,3.0,1,4,1,4.258822,2.965918,3.034083,1.741173,
3.292903,0.775247,3.5,1.133971,0,17,1,0)**

where

Value	Represents
a5	object type: arc (version 5)
1	z coordinate
0	flags
1	format (Interleaf)
3.0,1	x,y coordinates of point 1 (tangent at beginning point)
4,1	x,y coordinates of point 2 (beginning point)
4.258822, 2.965918	x,y coordinates of point 3 (a point on the arc between point 2 and point 4)
3.034083, 1.741173	x,y coordinates of point 4 (a point on the arc)
3.292903, 0.775247	x,y coordinates of point 5 (tangent at point 4)
3.5, 1.133971	x,y coordinates of point 6 (end-point)
0	interior texture (none)
17	border texture (black)
1	border width (1 pixel)
0	border pattern (solid)

Splines

Splines are smooth curved lines drawn through or near a series of control points called *knots*. The letter **S** specifies a spline object (version number is **10**), and **S10** is followed by the usual z coordinate and flags fields. The coordinates of the basis of the spline are next, followed by an interior texture ID and the total number of knots that follow. Each ensuing knot consists of its x, y coordinates in inches, relative to the basis, the line texture and line width (twice the number of screen pixels). After the last knot, there is one more field consisting of the letter "O" or "C" depending upon whether the spline is open or closed.

Here is a sample open spline:

**(S10,3,0,2.6,1.8,0,7,0,0,17,2,0,0,17,2,-0.3,-0.9,17,2,-1,-1.2,17,2,-1.6,-0.7,
17,2,-1.9,0.4,17,2,-1.9,0.4,17,2,O)**

where

Value	Represents
S10	object type: spline (version 10)
3	z coordinate
0	flags
2.6, 1.8	x, y coordinates (inches) of basis of spline
0	interior texture (none)
7	number of knots
0, 0	x, y coordinates (inches) of 1st knot relative to basis

17	line texture (1st knot)
2	line width (1st knot)
0, 0	x, y coordinates (inches) of 2nd knot relative to basis
17	line texture (2nd knot)
2	line width (2nd knot)
...etc.	
O	spline is open (end does not meet beginning)

Note that if the spline is open, the end-points must be given in duplicate. In the sample, **0,0,17,2,0,0,17,2** and **-1.9,0.4,17,2,-1.9,0.4,17,2** represent the end-points of the spline.

Here is a sample closed spline:

(S10,3,0,0.6,2.3,0,5,0,0,17,2,1.9,-0.4,17,2,1.6,-1.4,17,2,0.9,-1.7,17,2, 0.3,-1.2,17,2,C)

where

Value	Represents
S10	object type: spline (version 10)
3	z coordinate
0	flags
0.6, 2.3	x, y coordinates of basis of spline (in inches)
0	interior texture (none)
5	number of knots
0, 0	x, y coordinates of 1st knot relative to basis (in inches)
17	line texture (1st knot)
2	line width (1st knot)
1.9, -0.4	x, y coordinates of 2nd knot relative to basis (in inches)
17	line texture (2nd knot)
2	line width (2nd knot)
...etc.	
C	spline is closed (end meets beginning)

In both samples, the line width value (**2**) is double the number of screen pixels. A line width value if **4** would indicate a screen line width of 2 pixels.

Text

There are two kinds of text objects in diagrams, text strings and microdocuments.

The lowercase letter **t** indicates that the text object is a *text string* and the version number is **8**. After **t8** come the z coordinate and flags fields. The flags field should specify "locked-against-rotation" (bit 2 on, decimal **4**, if no other bits are on). Next come the x, y coordinates of the *text anchor* (in inches), which is at the left, center, or right of the text depending on the anchor type. Next comes the anchor type field which determines what sort of justification will be applied to the text. The valid values are **0** (left), **1** (center) and **2** (right). A texture field fol-

lows. The next field is a character string that specifies the font type (see the section *Font Representation* on page NO TAG). Care should be taken to use only valid names. The last field is a character string that is the text itself.

The following characters have special meanings in Interleaf ASCII Format and must be quoted if they are used:

< > ( ) , \

If you wish to use one of these characters in a text string, precede it with a backslash character (\). Also use a backslash to quote any spaces you wish to enter in a text string.

Here is a sample text string:

(t8,12,4,1.5,1.7,0,17,@nnclas12b,Hello\ there!)

where

Value	Represents
t8	object type: text — text string (version 8)
12	z coordinate
4	flags (locked against rotation)
1.5, 1.7	x, y coordinates of anchor
0	anchor type (left justification)
17	texture (black)
@nnclas12b	font (Classic 12 point bold, without underlining or strikethrough)
Hello\ there!	text string (with quoted space)

The capital letter **T** indicates that the text object is a *microdocument*, and the version number is **8**. After **T8** come the z coordinate and flags fields. The rest of the markup represents various information about the properties of the microdocument. The four vertical justification values—maximum component margin stretch, maximum component margin shrink, implied frame margin for stretch, and maximum feathering—are represented by 16-bit integers. For a discussion of the meaning of these values, see the chapter *Page Design* in Volume 2 of the *Reference Manual*. At the end of the markup for the diagramming text object, there *must* be a newline before the markup for the text itself. In the following example, the obligatory newline is after **(T8 . . . 1101,0,**.

Here is the markup for a sample microdocument:

```
(T8,1,12,0.8,0.76,0.8,1,0.333333,0,516,0,17,0,0,0,0,0,0,1,1,2048,512,102,
1101,0,
<caption,
        Alignment =             Left>
Hello, there!
<End Text>)
```

where:

Value	Represents
T8	object type: text — microdocument (version 8)
1	z coordinate
12	flags
0.8, 0.76	x, y coordinates of top left corner of microdocument (in inches)
0.8	page width (in inches, not including margins)
1	number of columns
0.333333	gutter width (in inches)
0	ladder count (number of consecutive hyphens; *0* indicates that there is no limitation)
516	page attribute flags (32-bit integer; *516* indicates that revision bars are on the left and hyphenation is on; *512* would indicate that revision bars were on the left and hyphenation was off)
0	interior texture (none)
0	border texture (none)
0	border width (none)
0	border pattern (solid)
0	top margin (in inches)
0	bottom margin (in inches)
0	left margin (in inches)
0	right margin (in inches)
1	vertical justification (on)
1	feathering (on)
2048	maximum component margin stretch (16-bit integer)
512	maximum component margin shrink (16-bit integer)
102	implied frame margin for stretch (16-bit integer)
1101	maximum feathering (16-bit integer)
0	object specific flags (for internal use; *0* indicates that the microdocument has fixed width)

The markup for text entered between the opening < and **<End Text>** is the same as the markup for any other text. See the previous chapter *Interleaf ASCII Format for Text*.

Vector-List (Plotter) Objects

When you save a vector-list object in ASCII Interleaf Format and look at the markup with another editor, the version number will be **8**. The difference between version **8** and version **7** is the introduction in version **8** of a parallelogram around the vector-list object. Since the publishing software will create a default parallelogram for objects created with version **7**, we recommend that you use version **7** when you are writing a filter and leave the parallelogram to the publishing software.

The object represented by this markup is also known as a plotter object because plotter data is often translated to this form.

The letter **V** represents a vector-list object and the version number is **7**. After **V7** are the z coordinate and the flags fields, which should be **1024** to lock gravity off. Next is a list of vector sets, each set consisting of two hexafied binary data sets.

A hexafied binary data set begins with the letter **X** and the version number is **0**. After **X0** comes the number of bytes that have been hexafied. This is followed by the string of hex characters. For example, the number 12 in 16-bit binary would be represented as:

(X0,2,000C)

The first hex set is the *number* of x, y coordinate pairs (maximum is 1024) in the following object. The second set is a series of x, y coordinates in RSU's (1,228,800 RSU's = 1 inch). After the last vector set, a hex set with two null bytes indicates the end of the list. Here is a sample plotter object that has only one vector set:

(V7,1,1024,
(X0,2,0006)
(X0,48,800000000000000000005EF220002C64A0079801F0000000000713C01003
60DA3000000000037A4D80002C64A0000CC89)
(X0,2,0000))

In this example, there are 6 sets of x, y coordinate pairs in the second hex set. Each coordinate is a 32-bit (4 byte) quantity that becomes 8 hex characters. There are 48 binary bytes which when hexafied yield 96 hex characters. The first coordinate pair is x=80000000 y=00000000. These values are special in that they are used to indicate **pen up**. The path to the next coordinate pair is not drawn. Thereafter, the path to each point *is* drawn until another **pen up** is encountered.

With version **8**, the previous sample vector-list object looks like this (with the numbers in italics representing the parallelogram; the specification for the parallelogram consists of three points with the fourth point implicit):

(V8,1,1024,*0,0,6.480025,0,0,2.967676,*
(X0,2,0006)
(X0,48,800000000000000000005EF220002C64A0079801F0000000000713C01003
60DA3000000000037A4D80002C64A0000CC89)
(X0,2,0000))

Plotter objects that require more than one vector set look like this:

```
(V7,1,1024,
 (X0,2,0080)
 (X0,1024,...)
 (X0,2,0080)
 (X0,1024,...)
 ...
 (X0,2,0000))
```

Each vector set can handle a maximum of 128 coordinate pairs (hex 0080), which is equivalent to 1024 binary bytes. Hexafication yields a string of 2048 characters.

Charts, Images, and Tracing Objects

Charts, images, and tracing objects should not be introduced into the Interleaf software via ASCII format. For charts, you can use a filter to load the chart data and then create the chart within the Interleaf software.

Field Specifications

This section specifies the values or formats allowed in the font, texture, line pattern, and flags fields.

Font Representation

The font field always has at least three entries. It has a fourth entry if the font is bold, italic, or bold italic.

The Format for the Font Field

- One of the following combinations comes first:

  @nn = no underlining and no strikethrough
  @un = underlining but no strikethough
  @ns = strikethrough but no underlining
  @us = underlining and strikethrough

- The second entry is between 3 and 8 letters representing the typeface family. The following generic names have been assigned by Interleaf:

  clas = Classic
  modn = Modern
  typw = Typewriter
  grk = Greek
  mtha = Mathematics A
  mthb = Mathematics B
  mthex = Mathematics Extension
  symb = Symbols

If you output to a typesetter, consult your system administrator for a list of the generic names of the typesetter fonts.

- The third entry is the point size (**8** for 8 point, **14** for 14 point, etc.).
- The fourth entry, if any, could be one of the following:

  the letter **b** if the font is bold
  the letter **i** if the font is italic
  the letters **b** and **i** if the font is bold italic (**b** must be before **i**)

For example:

- Classic 12 point bold that is neither underlined nor struckthrough is **@nnclas12b.**
- Modern 8 point italic that is underlined is **@unmodn8i.**
- Typewriter 10 point that is both underlined and struckthough is **@ustypw10.**

Fonts from Interleaf

The standard fonts that Interleaf ships to customers are:

- Classic and Modern in 8, 10, 12, 14, 18, and 24 point in roman, bold, and italic; 6 point in roman only
- Typewriter in 8, 10, and 12 point
- Greek in 10 and 12 point
- Symbols in 8, 10, 12, and 14 point
- Mathematics A, Mathematics B, and Mathematics Extension in 10 point

Textures

Figure 2-3 shows the valid texture values and the patterns they produce within the Interleaf publishing software.

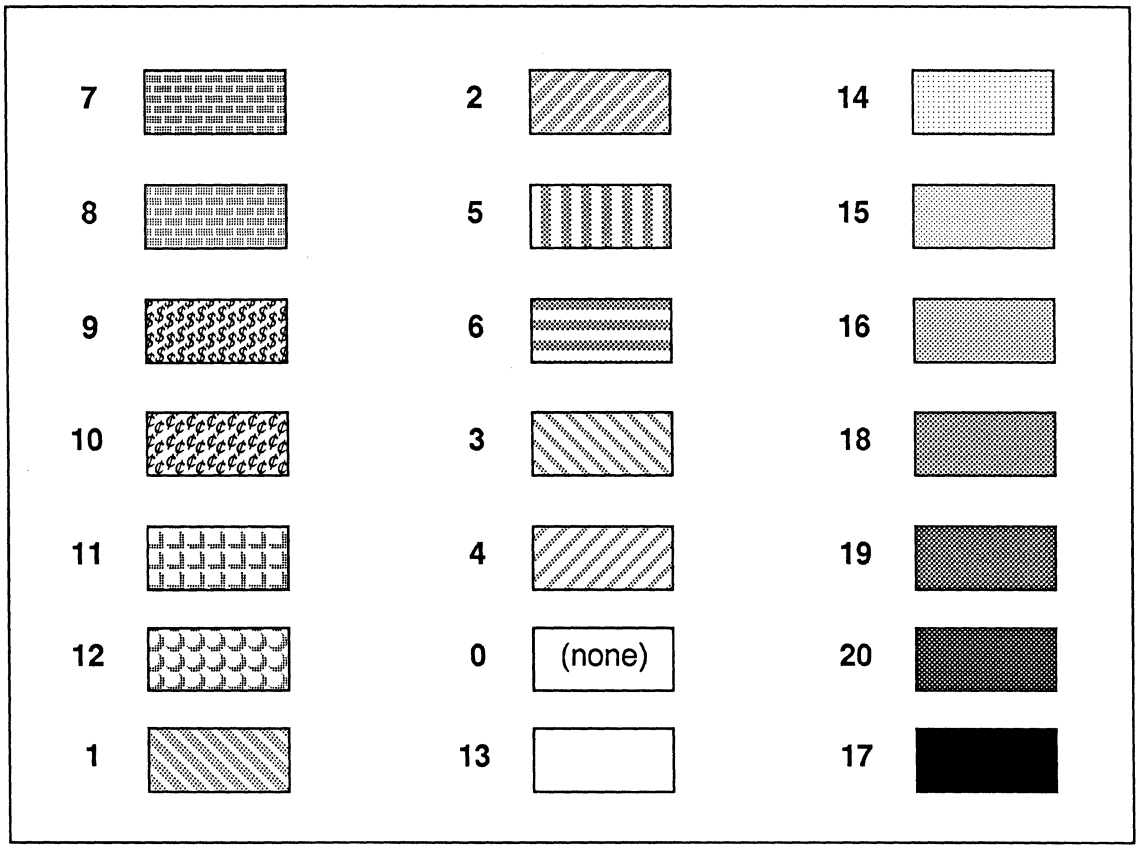


Figure 2-3. Texture field values and their patterns

The only valid values for line textures are **0** (none), **13** (white) and **17** (black). For interior textures (fill patterns), all of the values listed above are supported.

Line Patterns

Figure 2-4 shows the valid line pattern values and the lines they produce within the Interleaf publishing software.

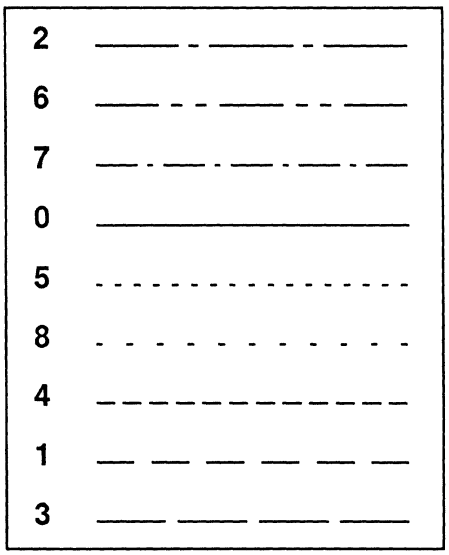


Figure 2-4. ASCII line pattern values and the lines they represent

Flags

Here is the mapping of the bits in the flags field (16-bit integer). Bit 0 is the least significant bit (LSB).

Bit	Function
0 (LSB)	lock size
1	lock position
2	lock against rotation
3	lock grouping
4	lock against selection
5	lock line widths
6	lock interior texture (fill pattern)
7	lock font
8	lock against printing
9	lock against cutting
10	lock gravity off
11	lock control points off
12	lock aspect ratio
13	lock smoothness
14	reserved for system use
15	reserved for system use

You will find most of these functions described in the chapter *Advanced Diagramming Concepts* of the *Reference Manual,* Volume 2.

Error Messages

If an error is found in a diagram when an Interleaf ASCII Format document is being loaded by the Interleaf software, a warning message such as

ASCII Loader Warning on line 29: Error in loading diagram

is displayed in the status line. The entire contents of the frame are skipped, and an empty frame is loaded. If there is more than one error, only the first message is displayed.

Appendix A

Default Document in ASCII Format

This appendix shows what the empty default document on the Interleaf desktop looks like in Interleaf ASCII Format. To see the markup for any document, open the document on your desktop, use a **Save ASCII** command, and examine the document with a non-Interleaf editor or print it from the operating system.

```
<!OPS, Version = 5.2>

<!Document,
        Final Output Device =          "cx",
        Default Printer =              "nearest-cx">

<!Font Definitions,
        F31 = Classic 10>

<!Page,
        Left Margin =                  1 inches,
        Right Margin =                 1 inches,
        Starting Page # =              Inherit,
        Hyphenation =                  on,
        Revision Bar Placement =       Left>

<!Autonumber Stream, List, 1>

<!Autonumber Stream, Outline, 3,
        Level 1 Symbol Type =          UPPER ROMAN,
        Level 2 Symbol Type =          UPPER ALPHA>

<!Class, paragraph,
        Top Margin =                   0.07 inches,
        Bottom Margin =                0.07 inches,
        Line Spacing =                 1.308 lines,
        Font =                         F31>
```

```
<!Master Frame,
        Name =                          "At Anchor",
        Placement =                     At Anchor,
        Width =                         0.41 inches,
        Height =                        0.137 inches,
        Vertical Alignment =            0.03 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          Bottom,
        Placement =                     Bottom of Page,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          "Following Anchor",
        Placement =                     Following Anchor,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          "Following Text",
        Placement =                     Following Text,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>
```

```
<!Master Frame,
        Name =                          Footnote,
        Placement =                     Bottom of Page,
        Horizontal Alignment =          Left,
        Same Page =                     yes,
        Width =                         6.50 inches,
        Height =                        0.204 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          Top,
        Placement =                     Top of Page,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<Page Header, Frame =
V4,
(g9,1,0
 (t8,1,4,6.493333,0.438782,2,17,@nnclas10,)
 (t8,2,4,3.246667,0.438782,1,17,@nnclas10,)
 (t8,3,4,0,0.438782,0,17,@nnclas10,)
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<Page Footer, Frame =
V4,
(g9,1,0
 (t8,1,4,6.493333,0.478772,2,17,@nnclas10,)
 (t8,2,4,3.246667,0.478772,1,17,@nnclas10,-\ \240\ -)
 (t8,3,4,0,0.478772,0,17,@nnclas10,)
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<paragraph>

<|,1>
```

Appendix B

Document with Text in ASCII Format

This appendix shows what the default document on the Interleaf desktop looks like in Interleaf ASCII Format when it has had a few changes made to it. The changes are the following:

- Two new autonumber streams, three named frames, and three new component classes have been declared.
- The master frame "filled space" has contents because it is the repository for the contents of frames named "filled space" with *Shared Contents* set to *yes*.
- The document is two-column, and some of the components straddle the columns.
- There is text, and there are commands for a special character and for autonumber, autoreference, and index tokens embedded in the text.

To see the markup for any document, open the document on your desktop, use a **Save ASCII** command, and examine the document with a non-Interleaf editor or print it from the operating system.

```
<!OPS, Version = 5.2>

<!Document,
        Final Output Device =          "cx",
        Default Printer =              "nearest-cx">

<!Font Definitions,
        F31 = Classic 10,
        F61 = Modern 18 Italic,
        F33 = Classic 10 Italic>

<!Page,
        Columns =                      2,
        Gutter =                       0.333 inches,
        Left Margin =                  1 inches,
        Right Margin =                 1 inches,
        Starting Page # =              Inherit,
        Hyphenation =                  on,
        Revision Bar Placement =       Left>

<!Autonumber Stream, List, 1>
```

```
<!Autonumber Stream, Outline, 3,
        Level 1 Symbol Type =          UPPER ROMAN,
        Level 2 Symbol Type =          UPPER ALPHA,
        Level 2 Trail =                yes>

<!Autonumber Stream, figure, 1,
        Level 1 Prefix =               "Figure ",
        Level 1 Suffix =               >

<!Autonumber Stream, invisible, 1,
        Level 1 Show =                 no>

<!Class, figure,
        Top Margin =                   0.07 inches,
        Bottom Margin =                0.07 inches,
        Line Spacing =                 1.308 lines,
        Font =                         F31,
        Straddle =                     yes>

<!Class, label,
        Top Margin =                   0.07 inches,
        Bottom Margin =                0.07 inches,
        Line Spacing =                 1.308 lines,
        Alignment =                    Center,
        Font =                         F33,
        Straddle =                     yes>

<!Class, paragraph,
        Top Margin =                   0.07 inches,
        Bottom Margin =                0.07 inches,
        Line Spacing =                 1.308 lines,
        Font =                         F31>

<!Class, title,
        Top Margin =                   0.07 inches,
        Bottom Margin =                0.07 inches,
        Line Spacing =                 1.308 lines,
        Alignment =                    Center,
        Font =                         F61,
        Straddle =                     yes,
        Allow Page Break After =       no>
```

```
<!Master Frame,
        Name =                          "At Anchor",
        Placement =                     At Anchor,
        Width =                         0.41 inches,
        Height =                        0.137 inches,
        Vertical Alignment =            0.03 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          Bottom,
        Placement =                     Bottom of Page,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          "Following Anchor",
        Placement =                     Following Anchor,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          "Following Text",
        Placement =                     Following Text,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>
```

```
<!Master Frame,
        Name =                          Footnote,
        Placement =                     Bottom of Page,
        Horizontal Alignment =          Left,
        Same Page =                     yes,
        Width =                         6.50 inches,
        Height =                        0.204 inches,
        Diagram =

V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          Top,
        Placement =                     Top of Page,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          "large space",
        Placement =                     At Anchor,
        Width =                         6.50 inches,
        Height =                        4.30 inches,
        Vertical Alignment =            0 inches,
        Diagram =
V4,
(g9,0,0
(E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
 6,6,0,0.066667,6))>

<!Master Frame,
        Name =                          "small space",
        Placement =                     At Anchor,
        Width =                         6.50 inches,
        Height =                        2.15 inches,
        Vertical Alignment =            0.03 inches,
        Diagram =
V4,
 (g9,0,0
  (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
    6,6,0,0.066667,6))>
```

```
<!Master Frame,
        Name =                    "filled space",
        Placement =               Following Anchor,
        Horizontal Alignment =    Center,
        Width =                   6.50 inches,
        Height =                  3.25 inches,
        Shared Contents =         yes,
        Diagram =
V4,
(g9,2,0
  (p7,2,8,0
    (g9,2,0
      (g9,2,0
        (v4,2,0,0.8,0.533333,1.4,0.533333,17,1,0)
        (v4,3,0,1.4,0.533333,1.4,2.4,17,1,0)
        (v4,4,0,1.4,2.4,0.8,2.4,17,1,0)
        (v4,5,0,0.8,2.4,0.8,0.533333,17,1,0))))
  (p7,7,8,0
    (g9,7,0
      (g9,7,0
        (v4,7,0,1.133333,0.666667,1.733333,0.666667,17,1,0)
        (v4,8,0,1.733333,0.666667,1.733333,2.533333,17,1,0)
        (v4,9,0,1.733333,2.533333,1.133333,2.533333,17,1,0)
        (v4,10,0,1.133333,2.533333,1.133333,0.666667,17,1,0))))
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,!0.066667,6))>

<Page Header, Frame =
V4,
(g9,1,0
 (t8,1,4,6.493333,0.438782,2,17,@nnclas10,)
 (t8,2,4,3.246667,0.438782,1,17,@nnclas10,)
 (t8,3,4,0,0.438782,0,17,@nnclas10,)
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<Page Footer, Frame =
V4,
(g9,1,0
 (t8,1,4,6.493333,0.478772,2,17,@nnclas10,)
 (t8,2,4,3.246667,0.478772,1,17,@nnclas10,-\ \240\ -)
 (t8,3,4,0,0.478772,0,17,@nnclas10,)
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<title,
        New Page = yes>

<|,1>A New Approach
```

```
<paragraph>

People often say, <#7f>There is nothing new under the sun." We say there
is<F33> if you look for it, <F0>and we'll show you how!

<Index, "Looking and discovering">It just takes time and a practiced eye to
discover something new and fresh. There are a number of ways to approach
the new. We'll use the relatively unfamiliar one of looking at a very blank
space (<Ref, Auto #, Tag = Z9u7W1cccal>) and imagining what might be in it.

<figure>

<Frame,
        Name =                          "large space",
        Placement =                     At Anchor,
        Width =                         6.50 inches,
        Height =                        4.30 inches,
        Vertical Alignment =            0 inches,
        Diagram =
V4,
(g9,0,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<label>

<Index, "Figures for thinking", "a large space and a small space", Typeface
= Italic><Autonum, figure, 1, First = Yes, Tagname = Z9u7W1cccal>.Blank
space to be filled with new ideas

<paragraph>

Once you get accustomed to looking at this space and filling it with new
ideas, you can move your ideas into the smaller space in <Ref, Auto #,
Tag = figure> and see what happens to them when they get crowded together.
By the time you get to page <Ref, Page #, Tag = 9et7W258cal>, you may be
ready to imagine something you never thought possible even in your wildest
dreams.
```

```
<figure,
        Allow Page Break After = no>

<Frame,
        Name =                          "small space",
        Placement =                     At Anchor,
        Width =                         6.50 inches,
        Height =                        2.15 inches,
        Vertical Alignment =            0.03 inches,
        Diagram =
V4,
(g9,0,0
 (E10,0,0,0,1,1,0.053333,1,15,0,0,1,0,0,0,1,0,1,1,0.066667,0.066667,
  6,6,0,0.066667,6))>

<label>

<Index, "Figures for thinking", Typeface = Italic><Autonum, figure, 1,
Tagname = figure>. Smaller blank space to be crowded with ideas

<paragraph>

<|,2><Autonum, invisible, 1, First = Yes, Tagname = 9et7W258cal>

<Frame,
        Name =                  "filled space",
        Placement =             Following Anchor,
        Horizontal Alignment =  Center,
        Width =                 6.50 inches,
        Height =                3.25 inches,
        Shared Contents =       yes,
        Page # =                2>
```

Appendix C

Using an Include Command

Between the two sets of double lines in this appendix is *recipe_template.doc* referred to on page 1-20. When the *<!Include Declarations* command is used in the garlic popcorn recipe, the part of *recipe_template.doc* in the dashed-line box is ignored. When the garlic popcorn recipe is saved, the declarations and the material in the solid-line box become one document.

```
<!OPS, Version = 5.2>

<!Class Defaults, Fill =                blank>

<!Document,
        Final Output Device =           "cx",
        Default Printer =               "nearest-cx">

<!Font Definitions,
        F31 = Classic 10,
        F61 = Classic 18 Bold,
        F39 = Classic 14 Italic>

<!Page>

<!Autonumber Stream, List, 1>

<!Autonumber Stream, Outline, 3,
        Level 1 Symbol Type =           UPPER ROMAN,
        Level 2 Symbol Type =           UPPER ALPHA>

<!Class, author,
        Alignment =                     Left,
        Font =                          F39>

<!Class, ingrdnts,
        Font =                          F31,
        Left Tab =                      1/2/3 inches>

<!Class, paragraph,
        Font =                          F31>

<!Class, title,
        Alignment =                     Center,
        Font =                          F61>
```

```
<!Master Frame,
        Name =                          "At Anchor",
        Placement =                     At Anchor,
        Width =                         0.41 inches,
        Height =                        0.137 inches,
        Vertical Alignment =            0.03 inches,
        Diagram =
V4,
(g9,32767,0)>

<!Master Frame,
        Name =                          Bottom,
        Placement =                     Bottom of Page,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0)>

<!Master Frame,
        Name =                          "Following Anchor",
        Placement =                     Following Anchor,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0)>

<!Master Frame,
        Name =                          "Following Text",
        Placement =                     Following Text,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0)>

<!Master Frame,
        Name =                          Footnote,
        Placement =                     Bottom of Page,
        Horizontal Alignment =          Left,
        Same Page =                     yes,
        Width =                         6.50 inches,
        Height =                        0.204 inches,
        Diagram =
V4,
(g9,32767,0)>
```

```
<!Master Frame,
        Name =                          Top,
        Placement =                     Top of Page,
        Horizontal Alignment =          Center,
        Width =                         6.50 inches,
        Height =                        3.25 inches,
        Diagram =
V4,
(g9,32767,0)>
```

```
<title>

<|,1>Roasted Pecans

<paragraph>

,Place pecans and butter in heavy pan. Cook in a 300 oven until pecans be-
,gin to brown. Stir occasionally to keep mixture from burning. Pour out on
,waxed paper, and add salt. Cool, then place in airtight container.
```

```
<title>

<|,1>Garlic Buttered Popcorn

<author>

Mrs. J. H. Smith, Jr.

<paragraph>

Place butter and garlic in a small saucepan. Heat until butter is
melted. Remove garlic and stir in parsley. Pour butter mixture over
popcorn and toss lightly.
```

Index

Italicized numbers refer to pages on which figures appear.

A

Arcs, 2-7—2-9, *2-8*

ASCII. *See* Interleaf ASCII Format

Autonumber Command, 1-12—1-13, 1-35

Autonumber Stream Declarations, 1-7

Autoreference Command, 1-12—1-13, 1-38

C

Canceling an ASCII Load, 1-17

Charts, 2-14

Commands, 1-6, 1-35—1-40
- Autonumber, 1-12—1-13, 1-35
- Autoreference, 1-12—1-13, 1-38
- Component, 1-35
- Fonts and Font Attributes, 1-11—1-12, 1-35—1-36
- Frame, 1-14, 1-35
- Hard Return, 1-36
- Hard Space, 1-38
- Header and Footer, 1-38—1-40
- Include, 1-18—1-22
- Include Declarations, 1-18—1-22
- Index Token, 1-14—1-15, 1-37
- Page Break, 1-11, 1-37
- Special Characters, 1-12, 1-37
- Tab, 1-13, 1-38
- Tables of Contents, 1-15

Component Class
- declarations, 1-8
- default values, 1-8—1-9
- definition of, 1-8
- during paste operations, 1-9

Component Command, 1-35

Coordinates, of diagramming objects, 2-2

Creating a Marked-up Document, 1-16

D

Declarations, 1-6, 1-30—1-35
- autonumber stream, 1-7, 1-32
- class, 1-33—1-34
- class defaults, 1-32
- comment, 1-35
- component class, 1-8
- document, 1-30
- font, 1-6—1-7
- font definitions, 1-32
- master frame, 1-10, 1-34—1-35
- OPS, 1-30
- page, 1-30—1-31

Default Values, 1-41—1-42
- autonumber, 1-42
- component, 1-42
- document, 1-41
- frame, 1-42
- page, 1-41—1-42

Diagramming Objects
- coordinates of, 2-2
- Interleaf ASCII Format for. *See* Markup, specific diagramming objects
- syntax, 2-1—2-3

Documents, externally created, 1-23, 1-24
- and desktop icons, 1-24

E

Ellipses, *2-6*

Error Messages, 1-45—1-49, 2-18

F

Field Specifications, 2-14—2-18
- flags, *2-17—2-18*
- fonts, 2-14—2-15
- line patterns, *2-17*
- textures, 2-16

Flags, 2-17—2-18

Font Declarations, 1-6—1-7

Fonts, in diagramming objects, 2-14—2-15

Fonts and Font Attributes Command, 1-11—1-12, 1-35—1-36

Frame Command, 1-14, 1-35

Frames, text objects in, 2-2

G

Groups, 2-4—2-5

H

Hard Return Command, 1-36

Hard Space Command, 1-38

Header and Footer Commands, 1-38—1-40

I

Images, 2-14

Include Commands
  details about, 1-21—1-22
  shortcuts, 1-21
  using, 1-18—1-22

Index Token Commands, 1-14, 1-37

Interleaf ASCII Format
  comparing versions of, 1-2—1-4
  definition of, 1-1
  documents, creating, 1-2
  error messages, 1-45—1-49
  specifications, 1-26—1-40

K

Keyboard Mapping, and special characters, 1-44

L

Line Patterns, values of, *2-17*

Lines, 2-4

M

Markup
  commands, definition of, 1-6
  declarations, definition of, 1-6
  for diagramming objects
    arcs, 2-7—2-9
    charts, 2-14
    ellipses, 2-6—2-7
    fonts, 2-14—2-15
    frames, 2-1, 2-3
    groups, 2-4—2-5
    images, 2-14
    lines, 2-4
    microdocuments, 2-11—2-12
    plotter objects, 2-12—2-14
    polys, 2-5—2-6
    splines, 2-9—2-10
    text objects, 2-10—2-12
    text strings, 2-10—2-12
    tracing objects, 2-14
    vector-list objects, 2-12—2-14
  within components, 1-10—1-15
    commands, 1-10—1-15
      Autoreferences, 1-12—1-13
      Fonts and Font Attributes, 1-11—1-12
      Frame, 1-14
      Index Token, 1-14—1-15
      Page Breaks, 1-11
      Special Characters, 1-12
      Tab, 1-13
      Tables of Contents, 1-15

Master Frame Declarations, 1-10

Microdocuments, 2-11—2-12

N

Non-marked-up Files, 1-24

P

Page Break Command, 1-11, 1-37

Plotter Objects, 2-12—2-18

Polys, 2-5—2-6

Printing an ASCII Document, 1-16—1-17

Q

Quoting Rules, 1-26, 2-2, 2-11

S

Save ASCII Command, 1-4—1-5, 1-17

Special Characters, and keyboard mapping, 1-43—1-44

Special Characters Command, 1-12, 1-37

Specifications for Interleaf ASCII Format, 1-26—1-40
  commands, 1-29, 1-35—1-40
    Autonumber, 1-35
    Autoreference, 1-38

Specifications for Interleaf ASCII Format (cont.)
Component, 1-35
Fonts and Font Attributes, 1-35
Frame, 1-35
Hard Return, 1-36
Hard Space, 1-38
Header and Footer, 1-38—1-40
Index Tokens, 1-37
Page Break, 1-37
Special Characters, 1-37
Tab, 1-38
declarations, 1-29, 1-30—1-35
autonumber stream, 1-32
class, 1-33—1-34
class defaults, 1-32
comment, 1-35
document, 1-30
font definitions, 1-32
master frame, 1-34—1-35
OPS, 1-30
page, 1-30—1-31
naming conventions, 1-27
quoting rules, 1-26
syntax rules, 1-27
newlines, 1-28
order of declarations and commands, 1-27
units of measurement, 1-29
within markup brackets, 1-28
within text, 1-29
Splines, 2-9—2-10

T

Tab Commands, 1-13, 1-38
Templates for ASCII Documents, 1-17—1-18
Text Objects, 2-10—2-12
Text Strings, 2-10—2-12
Textures, values of, *2-16*
Tracing Objects, 2-14
Translation Programs, hints on writing, 1-25

V

Vector-List Objects, 2-12—2-18
Version Numbers, for frames and diagramming objects, 2-2