Customizing Frame Products

Frame Technology Corporation
1010 Rincon Circle
San Jose, California 95131
USA

Frame Technology International Limited
Unit 52, Airways Industrial Estate
Cloghran, Dublin 17
Ireland

September 1993

Frame®

Important Notice

Customizing Frame Products Contents

Go To

To get help on using this manual, click here.
To go to a section, click on a topic below.

Print Manual

1 Introduction

Go To

This manual describes how to customize Frame® products by changing settings in initialization files and creating custom menu configurations.

Initialization files contain settings that define your Frame product environment. For example, by changing settings in an initialization file, you can change marker names or define the initial position of Frame product windows. For more information, see Chapter 2, "Changing Initialization Files."

Menu customization files define Frame product menus and commands. By changing these files, you can customize the Frame product user interface. For instructions, see Chapter 3, "Changing Menus and Commands."

Locating customization files

Throughout this manual, *install_dir* refers to the directory in which the Frame product is installed. Customization files are located in *install_dir*, *install_dir*\fminit, and *install_dir*\fminit*product*, where *product* represents a product-specific directory such as maker for FrameMaker®. Product-specific directories let several Frame products share the same installation directory.

2 Changing Initialization Files

Go To

You can customize your Frame product environment in many ways by editing initialization files. In some cases, you may also want to edit the setup files referred to in these initialization files.

This chapter describes changes you can make to Frame products:

About initialization files

Initialization files are ASCII text files that define many elements of your working environment. You can edit settings in these files using a Frame product or a text editor like Notepad.

> **Important:** If you open an initialization file in a Frame product, you'll get an alert box telling you that the file is an unknown type. Make sure that Text is selected in the alert box, and click Convert to convert the file to a Frame product file. When you're finished editing the file, save it as Text Only in the Save Document dialog box.

The Frame product initialization file is in *install_dir*, and is named after the product. For example, the FrameMaker initialization file is `maker.ini`. The initialization file contains settings such as default options for the Preferences dialog box, values for the Zoom pop-up menu, and the names of marker types. It also gives the locations of dictionaries and other files that the Frame product needs to find as you work. These initial settings are made when you install the Frame product on your system.

The settings in the Frame product initialization file are grouped under bracketed headings. Each heading tells the system how to interpret the information that follows. A line that begins with a semicolon is a comment and is not read by the system.

The Windows initialization file, `win.ini`, comes with Windows and is in the `windows` directory. This file determines low-level characteristics including cursor blink rate, desktop pattern, and filename extensions. Its settings are also grouped under bracketed headings.

When you install a Frame product, new entries are added to `win.ini`. These settings define filename extensions for Frame product files and give the location of the Frame product application and its initialization file.

If you edit the Frame product initialization file or the Frame product settings in `win.ini`, the changes will take effect the next time you start the Frame product or the next time you start Windows.

Special characters

While editing a Frame product initialization file, you may need to type a special character. If you need a character with a Frame product hex code greater than 7e, type a backslash followed by the character's hex code. For example, to create a marker called *Mañana*, you need the hex code of the n character (x96). Your entry might look like this:

```
10 = Ma\x96ana
```

The Frame product hex codes are listed in the *Quick Reference*.

System information about the Frame product

The `win.ini` file contains information that the File Manager needs to launch the Frame product and defines the location of the Frame product in your file system.

Location of the Frame product

The `[FrameMaker 4]` or `[FrameBuilder 1.0]` settings in `win.ini` give the location of the Frame product and its initialization file. For example:

```
FMHome = install_dir
IniFile = maker.ini
```

Change these settings if you move the Frame product or the initialization file to another directory, or if you're working on a network and have made your own local copy of the initialization file. You can specify an absolute pathname or a pathname relative to *`install_dir`*.

You may also want to define a new location for some setup files. For information about changing the location of these files, see "Directories and setup files" on page 10.

Filename extensions

Use the File Manager Associate command to associate filename extensions with the Frame product. If you double-click a filename with one of these extensions in the File Manager, the Frame product starts automatically and opens the file.

Basic characteristics of the Frame product

The `[Frame]` settings in the Frame product initialization file define a few basic characteristics of the Frame product on your system:

```
Version = version
Language = USEnglish
PCName = UnKnown
```

The `Language` setting specifies the default language dictionary and the default paragraph language. You can enter the name of one of the other languages available in the Frame product, but you must have the dictionary for that language installed for the Frame product to work.

You can assign your computer a name with the `PCName` setting. If Network File Locking is turned on in the Preferences dialog box or in the initialization file, when you're working on a network and someone else tries to open a file you already have open, the other user sees a message that the file is in use. If your computer has a name, the name is included in this message.

Options for your working environment

The `[Preferences]` settings in the initialization file specify several options for your working environment.

Exiting Frame products

You can specify whether or not the Frame product asks for confirmation before exiting. To do so, change this setting to `On` or `Off`:

```
AskExit = Off
```

Settings for the Preferences dialog box

The next group of lines under `[Preferences]` provides default settings for the Preferences dialog box. This is how the settings are initially defined:

```
BackupOnSave = On
AutoSave = Off 5
ShowErrors = On
ErrorFileName = consfile.txt
```

```
GreekSize = 5
FMImage = Off
NetworkLock = Off
```

You can edit the values and change any `On` setting to `Off` (and vice versa). The value for `AutoSave` is in minutes, and the value for `GreekSize` is in points. The value for `ErrorFileName` is the name of the file in which the product stores error messages.

These values are updated every time you exit the Frame product, using the current settings in the Preferences dialog box.

For information about these preferences, see your user's manual.

API clients

API clients are programs that are integrated with Frame products using the Frame Application Program Interface™ (API). The `[APIClients]` section of the initialization file contains a list of API clients to start when Frame products start. (For more information, see "API Clients" on page 15). Use the following setting to specify that these programs are started:

```
API = On
```

To start Frame products without starting API clients, change this setting to `Off`. If you do so, the WordCount document report and the online manuals' Print Manual buttons won't work.

Values for the Zoom pop-up menu

This line defines the values that appear in the Zoom pop-up menu:

```
Zoom = 25, 50, 80, 100, 120, 140, 150, 160, 200, 400
```

You can edit the line to change these values to any values between 25 and 1600. The values are percentages, with 100 representing a document's print size. If the line contains more than 10 values, only the first 10 appear in the pop-up menu.

The zoom values are updated every time you exit the Frame product, using the current settings in the Zoom pop-up menu.

Values for the Line Widths pop-up menu

This line defines the widths that appear in the Line Widths pop-up menu in the Tools window:

```
PenWidths = 0.5, 1.0, 3.0, 4.0
```

Each value specifies a line width in points. You can change a width to any value between .015 and 360 points.

These values are updated every time you exit the Frame product, using the current settings in the Line Widths pop-up menu.

Changing a line width for the Line Widths pop-up menu affects only new lines you draw with that width. To change the width of an existing line, select the line and choose a width from the pop-up menu.

Format priorities for the Windows Clipboard

An item can be stored on the Windows Clipboard in more than one format. When you paste the item into a document with the Paste command, the format used is the one with the highest priority in the `ClipboardFormatPriorities` setting.

This is how the setting initially appears:

```
ClipboardFormatsPriorities = OLE, META, DIB, BMP, MIF, RTF, TEXT
```

Priority in the list goes from left (highest) to right (lowest). To give a format higher priority, move it farther left in the list.

If you want the Frame product to use any format that appears first on the Windows Clipboard, delete all of the formats from the setting:

```
ClipboardFormatsPriorities =
```

You can override these settings while working in the Frame product by using the Paste Special command rather than the Paste command.

Monitor size

You can specify the size of your monitor so that the documents and dimensions you see on the screen (such as for margins and objects) are very close to their actual size.

Initially, this setting is 0:

```
MonitorSize = 0in
```

If you leave the setting at 0, the Frame product uses system information about the size of your monitor. But if you want greater precision in the measurements you see on the screen, fill in the diagonal measure of your monitor. For example:

```
MonitorSize = 16in
```

Important: You must include a unit of measure (`in`, `"`, or `cm`) with this value.

Menu preferences

Windows command shortcuts include mnemonic shortcuts—underlined letters in command labels. When a menu is open, you can choose a command by typing the underlined letter (for example, the X underlined in the Exit command on the File menu). Frame product menu configuration files let you specify which letter is underlined in a command label. When the product loads the menu configuration file, it automatically generates the specified mnemonic shortcuts. To prevent Frame products from creating these command shortcuts, change the `AutoMnemoniseMenus` setting from `On` to `Off`. For more information about Frame product command configuration files, see Chapter 3, “Changing Menus and Commands.”

The StickyPopupThreshold setting controls how long you must hold down the mouse button for pop-up menus, such as the Zoom menu, to remain open. The preset value, 500, specifies an interval similar to that for opening menus in the menu bar.

Default margins, gaps, and view settings

Several lines under [Preferences] provide default values for margins, the gap between columns, tick marks in the rulers, and units for the grids in new custom documents. These are the preset settings:

```
DefaultMarginInch = 1in, 1in, 1in, 1in
DefaultMarginCm = 2.5cm, 2.5cm, 2.5cm, 2.5cm
DefaultColGapInch = 0.25in
DefaultColGapCm = 1cm
DefaultRulerInch = 0.125in
DefaultRulerCm = 0.5cm
DefaultGridInch = 0.5in
DefaultGridCm = 1cm
DefaultSnapInch = 0.125in
DefaultSnapCm = 0.25cm
```

These settings specify values in both inches and centimeters. The values that the Frame product uses depend on the Country and Measurement settings in the International Control Panel in Windows.

The DefaultGrid settings are for the visible grid, and the DefaultSnap settings are for the snap grid.

Important: You must include a unit of measure (in, ", or cm) with each value.

PostScript printing

Frame products can print faster and produce higher quality documents when they use built-in methods of generating PostScript code instead of standard Windows methods. Frame products are preset to use the built-in method:

```
UsePostscript = On
```

To use Windows methods only, change this setting to Off.

Text character encoding

The next lines under [Preferences] define which form of character encoding the Frame product uses when you open text files or save documents as text files. This is initially set to ANSI encoding:

```
TextCharacterEncoding = ANSI
;TextCharacterEncoding = OEM
```

Change the setting to OEM encoding if you need to move text files back and forth between the Frame product and a DOS word-processing application.

To change to another setting, remove the semicolon from the setting you want and insert a semicolon before the other setting.

Initial view settings

The `ShowQuickAccessBar` and `ShowFormattingBar` settings control whether the QuickAccess bar and formatting bar appear when you start the Frame product:

```
ShowQuickAccessBar = On H
ShowFormattingBar = On
```

To specify a horizontal or vertical QuickAccess bar, include an *H* or *V* in the `ShowQuickAccessBar` value. These values are updated every time you exit the Frame product, using the current settings in the View menu.

Menu sets and keyboard shortcuts

Frame products have two built-in sets of menus. Complete menus are the menus and commands as described in your user's manual and online Help. Quick menus are a subset of complete menus. You can customize complete or quick menus, or you can create a custom menu set. For more information, see Chapter 3, “Changing Menus and Commands.”

The `MenuSet` setting defines which set of menus appears when you start the Frame product:

```
MenuSet = Complete
;MenuSet = Quick
```

To change the `MenuSet` setting, remove the semicolon from the setting you want and insert a semicolon before the setting you don’t want. If you’ve loaded a menu customization file and selected a completely custom menu set while using the Frame product, a third setting appears in the initialization file:

```
MenuSet = Custom
```

You can manually add this line to instruct the Frame product to display the custom menu set when it starts. To instruct the Frame product to load a menu customization file when it starts, use the `ConfigCustomUIFile` setting to specify its filename. See “Menu and command configuration files” on page 12.

Menu configuration files define keyboard shortcuts. You can specify custom menus and keyboard shortcuts in menu customization files. The next two settings in the initialization file determine whether a Frame product displays warning messages when you load custom keyboard shortcuts:

```
ConfigWarnKbdRedundant = Off
ConfigWarnKbdOverride = Off
```

To display warning messages when menu customization files contain redundant shortcuts, set `ConfigWarnKbdRedundant` to `On`. To display warning messages when shortcuts override existing ones, set `ConfigWarnKbdOverride` to `On`. The warning messages appear in the Frame console window.

Directories and setup files

The `[Directories]` and `[Files]` settings in the initialization file specify the names and locations of directories and files that the Frame product needs to find. This information is entered for you when you install the Frame product. You should change the information if you rename or move a directory or file, or if you're working on a network and have made your own local copy of a setup file.

The locations in these settings are relative to *`install_dir`*. For any of the settings, you can use an absolute pathname instead.

You may want to modify a file itself rather than its name or location, particularly the dictionaries or the custom template. For information about changing templates and editing dictionary files, see your user's manual.

Directory names

The settings under `[Directories]` assign names to several directories:

```
HelpDir = help\product
LanguageDir = dict
TemplateDir = template
FilterDllDir = filters
OpenDirOnStart =
PaletteDir =
AlwaysOnTopPaletteDir =
```

`HelpDir` has the files the Frame product uses for the online Help system.

`LanguageDir` has files with language-specific information, such as hyphenation settings and language dictionaries for spell-checking.

`TemplateDir` contains the templates that come with the Frame product, except the ones for custom documents and text files. For information about these templates, see your user's manual.

`FilterDllDir` contains the filters installed on your system.

`OpenDirOnStart` specifies which directory should initially appear in the Open dialog box. Every time you exit the Frame product, this is updated to the directory from which you last opened a document.

`PaletteDir` and `AlwaysOnTopPaletteDir` specify palette directories. The values for these settings are a list of directory names, separated by commas. When you store a document in one of these directories, the Frame product treats it as a palette (such as the

Tools palette or Equations palette). Palettes in `AlwaysOnTopPaletteDir` always appear in front of documents.

Help directories

The remaining settings in the `[Directories]` section control where the online Help system looks for files that it uses. (If no directory name appears, the product looks in `$FMHOME`.) If you change these settings, the online Help system might not work properly:

```
Samples = samples
Clipart = clipart
OnlineManuals = manuals
ReleaseNotes =
Templates = template
```

Dictionaries and setup files

The first two settings under `[Files]` give the names and locations of dictionary files:

```
UserDictionary = user.dct
SiteDictionary = dict\site.dct
```

The rest of the entries are setup files that define some of the features in the Frame product:

```
CustomDoc = fminit\custom
CompareDoc = fminit\compare
AsciiTemplate = fminit\txttmplt
EquationDoc = fminit\equation
ThesaurusDoc = fminit\thesaurs
LayoutDoc = fminit\layout
TemplateBrowserDoc = fminit\tmpltbrw
VerticalQuickAccessBar = fminit\vertqab

EPSHeader = fminit\header.ps
FMFont = fminit\fmfont.fot
FMSmallFont = fminit\fmsmall.fon

Resources = fminit\fmres.dll
DialogResources = fminit\fmdlg.dll
AlternateResources = fminit\fmaltres.dll

ToolBarIniFile = fminit\fmtoolbr.ini
```

`CustomDoc` is the template for new custom documents, `CompareDoc` is the template for composite comparison documents, and `AsciiTemplate` is the template for text files.

The next group of settings are for locked hypertext documents that the Frame product uses. `EquationDoc` is the Equations palette, `ThesaurusDoc` is the Thesaurus window,

LayoutDoc is the Layout palette, and TemplateBrowserDoc is the Standard Templates dialog box. VerticalQuickAccessBar is the vertical QuickAccess bar.

The EPSHeader setting is set by the Frame product; do not change it. FMFont contains a font that the Frame product uses for symbols for anchored frames, markers, tabs, and so on. FMSmallFont contains the font used in the Tools palette.

Resources, DialogResources, and AlternateResources define the Frame product dialog boxes, menus, error messages, and other resources. ToolBarIniFile specifies an initialization file used to determine the layout of the QuickAccess and formatting bars.

Menu and command configuration files

Menu and command configuration files define Frame product menus and commands. Several settings in the [Files] section define the location of these files:

```
MathCharacterFile = fminit\mathchar.cfg
ConfigCommandsFile = fmint\cmds.cfg
MSWinConfigCommandsFile = fminit\wincmds.cfg
ConfigMathFile = fminit\mathcmds.cfg
ConfigMenuFile = fminit\product\menus.cfg
ConfigCustomUIFile = fminit\customui.cfg
```

The MathCharacterFile file defines special math characters. The ConfigCommandsFile file contains basic commands, and the MSWinConfigCommandsFile file contains Windows-specific commands. The ConfigMathFile file contains math commands. The ConfigMenuFile contains the standard menus, and the ConfigCustomUIFile contains custom menus to load when the product starts. You can either store custom menus in the default file, customui.cfg, or change this setting to the pathname of your menu customization file.

Recently visited files

The [RecentlyVisitedFiles] section lists the most recently opened files to display at the bottom of the File menu. These values are updated every time you exit the Frame product.

Positions for windows

The [DialogLayout] settings in the initialization file specify a default location on the screen for some of the Frame product windows:

```
MakerWin = 3, -1, -1, -1, -1, 0, 0, 640, 400
PCatalog = 500, 40, 130, 180
CCatalog = 500, 220, 130, 180
Tools = 600, 0
Equation = 0, 0
Spell = 0, 0
```

```
PFormat = 0, 0
CFormat = 0, 0
Search = 0, 0
Table = 0, 0
Ruling = 0, 0
Markers = 0, 0
CondText = 0, 0
BookKit = 0, 0
Help = 0, 0
ECatalog = 0, 40, 130 180
Thesaurus = 0, 0
Layout = 0, 0
TemplateBrowser = 0, 0
VQuickAccessBar = 0, 0
ConsoleWin =
```

If you open and move one of these windows in the Frame product, the location is modified in the `[DialogLayout]` setting when you exit. This becomes the window's default location the next time you use the Frame product.

Most lines have two values. The first value in a line is the x value and specifies the offset in pixels from the left side of the screen. The second value is the y value and specifies the offset in pixels from the top of the screen. For example,

```
Spell = 46, 98
```

specifies a location for the Spelling Checker window 46 pixels to the right of and 98 pixels down from the upper-left corner of the screen.

The settings for the Paragraph Catalog, Character Catalog, and Element Catalog can also include a default window size. The width and height in pixels appear after the x and y values.

The settings for `MakerWin` and `ConsoleWin` specify normal, minimized, and maximized positions. The first setting determines which position to use; its value is 1 for normal, 2 for minimized, or 3 for maximized. The next six values are pairs of x and y offsets that specify the different window positions; the first pair specify the minimized position, the next pair specify the maximized position, and the final pair specify the normal position. The last two values are the normal window width and height. The default `ConsoleWin` settings depend on your monitor's size and resolution.

Names of marker types

The [Markers] settings in the initialization file provide the names of types of markers for the Marker window. The names are assigned numbers from 0 to 25.

These are the preset marker names for the English version of a Frame product:

```
0 = Header/Footer $1
1 = Header/Footer $2
2 = Index
3 = Comment
4 = Subject
5 = Author
6 = Glossary
7 = Equation
8 = Hypertext
9 = Cross-Ref
10 = Conditional Text
11 = Type 11
12 = Type 12
13 = Type 13
14 = Type 14
15 = Type 15
16 = Type 16
17 = Type 17
18 = Type 18
19 = Type 19
20 = Type 20
21 = Type 21
22 = Type 22
23 = Type 23
24 = Type 24
25 = Type 25
```

You can edit the names for marker types 11 through 25, and the names will appear in the Marker window the next time you start the Frame product. If you need to use a special character, type the Frame product hex code for it shown in the character set in the *Quick Reference*.

> **Important:** Do not rename the marker types numbered 0 through 10. Frame products assign special meanings to the names for types 0 through 10.

API Clients

The [APIClients] settings list API clients to start when the Frame product starts. Each client description must be on a separate line in the following format:

client = *type*,*description*,*DLL_file*

where *client* is the client name, *type* is the type of client, *description* is a description of the client, and *DLL_file* is the pathname for the client's DLL file. Valid types are Startup, Standard, ImportFilter, and ExportFilter. To create a client, you need the Frame Developer's Kit™ (FDK) for Windows. Follow instructions provided with an API client to add its startup information to the initialization file.

Dash patterns

The [DashPatterns] settings define the eight dash pattern choices that appear in the Dashed Line Options dialog box. You can edit these settings to replace the standard choices with custom ones:

```
1 = Dash,8,6
2 = Hidden,4
3 = Longdash,16,10
4 = Dot,2,4
5 = Dash-Dot,12,6,2,6
6 = Dash-Dot-Dot,12,6,2,6,2,6
7 = Chain,12,6,6,6
8 = Phantom,20,6,6,6,6,6
```

Each dash pattern contains a label that identifies the dash pattern in the Dashed Line Options dialog box and a series of dash and gap segment lengths. Frame products draw a dashed line by repeating a sequence of dashes and gaps. Dashes and gaps are measured in points. The following illustration shows a dashed line and its segment description. The line is made up of a 12-point dash, a 6-point gap, a 6-point dash, and another 6-point gap. Frame products repeat this pattern to draw a dashed line of any length.

Dashed line:— – — – — – —
Dash segments: 12, 6, 6, 6

The following setting describes the same line:

```
7 = Chain,12,6,6,6
```

If a dash pattern setting contains an odd number of segment lengths, the last dash value is repeated for the final gap. For example, the following setting describes a dash pattern with 4-point dashes and 4-point gaps:

```
2 = Hidden,4
```

The dash pattern label cannot contain spaces or other punctuation marks.

Document comparison options

The [DocCompare] settings specify options for comparing documents:

```
CreateSummaryOnly = Off
MarkInsertedText = ConditionInserted
;MarkInsertedText = ConditionTag
;MarkInsertedText = Nothing
InsertConditionTag = Inserted
MarkDeletedText = ConditionDeleted
;MarkDeletedText = ConditionTag
;MarkDeletedText = ReplacementText
DeleteConditionTag = Deleted
DeleteReplacementText = ^
AddChangeBars = On
InsertHyperTextLinks = On
ThresholdFactor = 75
```

The CreateSummaryOnly setting sets the Create radio buttons in the Compare Documents and Compare Books dialog boxes. When this setting is On, the Frame product creates a summary document only. When this setting is Off, the product creates both summary and composite documents.

The MarkInsertedText settings control the standard choice for marking insertions. You can choose the standard Inserted condition, ConditionInserted; a custom condition, ConditionTag; or Nothing. Edit the file so the option you want is the only one without a semicolon before it. If you choose a custom condition for inserted text, specify its tag with the InsertConditionTag setting.

The MarkDeletedText settings control the standard choice for marking deletions. You can choose the standard Deleted condition, ConditionDeleted; a custom condition, ConditionTag; or replacement text, ReplacementText. Edit the file so the option you want is the only one without a semicolon before it. If you choose a custom condition for deleted text, specify its tag with the DeleteConditionTag setting. If you choose replacement text, add the text string to the DeleteReplacementText setting (in place of the ^).

The AddChangeBars setting controls whether change bars are added to the composite document, and the InsertHyperTextLinks setting controls whether hypertext links are added to the summary document. The ThresholdFactor setting controls when to mark an entire paragraph or table cell as changed. The preset value is 75; an entire paragraph is marked as changed if 75 percent or more of the words are changed. You can increase or decrease this percentage.

Spelling options

The [Spelling] settings in the initialization file define default options for spell-checking and specify which style of quotation marks to use.

Settings for the Spelling Checker Options dialog box

The first group of lines under [Spelling] contains default settings for the Spelling Checker Options dialog box. This is how the settings are initially defined:

```
FindRepeatedWords=On
FindUnusualHyphenation=Off
FindUnusualCap=Off
IgnoreSingleCharWords=On
IgnoreAllCaps=On
FindStraightQuotes=On
FindExtraSpaces=On
IgnoreRomanNumerals=Off
IgnoreWordsWithDigits=On
FindTwoInARow=On !,.:;?
IgnoreWordsContaining=On .
FindSpaceBefore=On !%),.:;?]}\xC8\xD3\xD5\xDD
FindSpaceAfter=On $([{\xC7\xD2\xD4\xDC\xE2\xE3
```

You can change any On setting to Off (and vice versa) and specify different characters in the bottom four lines.

These values are updated every time you exit the Frame product, using the current settings in the Spelling Checker Options dialog box.

> **Important:** If you edit FindTwoInARow, IgnoreWordsContaining, FindSpaceBefore, and FindSpaceAfter, be sure to leave a space between the On/Off toggle and the character or group of characters to the right of it.

Quotation marks for Smart Quotes

The last group of lines under [Spelling] determines which characters appear when you press the single quotation mark (') or double quotation mark (") key with Smart Quotes on. Several types of quotation mark characters are available: English (‘ ’ and “ ”), German (‘ ’ and „ “), French (‘ ’ and « »), Swedish and Finnish (’ ’ and ” ”), and Italian (‘ ’ and “ ”).

The Smart Quotes options appear in the file as comments:

```
;Smart Quote Characters
;SmartQuotes \xd4\xd5\xd2\xd3 ) English curved quotes
;SmartQuotes \xe2\xd4\xe3\xd2 ) German style quotes with base quotes
;SmartQuotes \xdc\xdd\xc7\xc8 ) French style quotes using guillemets
;SmartQuotes \xd5\xd5\xd3\xd3 ) Swedish and Finnish style quotes
;SmartQuotes \xd4\xd5\xd2\xd3 ) Italian curved quotes
```

Below the comments, one of the styles is already set for you. For example, this is how the setting initially appears in the English versions of Frame products:

```
;English curved quotes:
SmartQuotes = \xd4\xd5\xd2\xd3
```

If you want to revise the setting to use one of the other options, you can copy and paste from the comment the codes for the style you want.

Basic font settings

The `[Fonts]` settings in the initialization file determine the default fonts, the fonts not used for spell-checking, font sizes in a pop-up menu, and definitions that the Frame product needs to interpret fonts from other platforms.

Values for the Size menus

This line defines the values that appear in the Font Size submenu on the Format menu and in the Size pop-up menu in the Paragraph Designer and Character Designer windows:

```
Sizes = 7pt, 8pt, 9pt, 10pt, 11pt, 12pt, 14pt, 18pt, 24pt, 36pt
```

You can edit the line to change the values to any values between 4 and 400 points. If the line contains more than 10 values, only the first 10 appear in the menus.

Definitions for the font profile

Several lines under `[Fonts]` define the terms that the Frame product uses for font information. This is how the definitions initially appear:

```
Angles = Regular, Kursiv, Slanted, Oblique, Italic, Obliqued
Variations = UltraCompressed, ExtraCompressed, Compressed, Condensed,
  Narrow, Regular, Wide, Poster, Expanded
Weights = Thin 100, ExtraLight 200, SemiLight 250, Book 300, Light 300,
  Book 300, Regular 400, SemiBold 600, Bold 700, ExtraBold 800,
  Heavy 900
```

The `Variations` and `Angles` settings define the names of variations and angles that the Frame product may encounter when opening a document from another platform.

Windows uses values to define font weights, and other platforms use terms. The `Weights` settings associate terms from other platforms with font weights appropriate for Windows; for example, Thin is interpreted as the weight 100.

You can edit these settings if your document has variations, angles, or weights not specified in the default settings.

Default fonts

Some of the [Fonts] settings define the default fonts for a new document. If the Frame product needs to remap characters for fonts that are unavailable and you do not have an appropriate font map specified, it remaps the characters to the default font.

These are the default font settings:

```
DefaultSize = 12
DefaultFamily = Times, Times New Roman, Tms Rmn
DefaultAngle = Regular
DefaultVariation = Regular
DefaultWeight = Regular
MathFamily = Symbol
```

You can edit any of these settings to use a different font or other characteristic. If you change the font family, be sure to use a family installed on your system.

For information about specifying a font map, see "Mappings for unavailable fonts" on page 21.

Menu fonts

The UICalcFont setting tells Frame products what font to use when determining the size of pop-up menus. You should not change this setting.

Fonts not used in a spell-check

This line defines the font families that the Frame product will exclude during a spell-check:

```
NonTextFamilies = ZapfDingbats, Symbol, WingDings, Monotype Sorts
```

Whenever you use the Spelling Checker command in a document, the Frame product will ignore any text in these font families.

Aliases for Windows fonts

A font in Windows does not always have the same name as the comparable font on another platform. This often happens because in Windows a variation such as Narrow is part of a font family, whereas on other platforms the variation is an option independent of the font family. Moreover, Windows uses only regular and italic for angles, while some fonts on other platforms have additional angles such as oblique. For more information about cross-platform compatibility, see the online manual *Using Frame Products on Multiple Platforms*.

Aliases for font angles and weights

Windows fonts use different font angles and weights, even when font names are the same as on other platforms. The settings under [FontAngleAliases] and [FontWeightAliases] assign angles and weights used on other platforms to Windows angles and weights. The preset assignments are:

```
[FontAngleAliases]
Obliqued = Oblique

[FontWeightAliases]
Medium = Regular
Roman = Regular
Semi = SemiBold
Demi = DemiBold
Bolded = Bold
```

Aliases for font names

The settings under `[WindowsToFrameFontAliases]` each assign a Windows font to a Frame product font name. Thus, font information appears in the Windows interface as it does in other versions of Frame products. For example, Helvetica Narrow is normally a font family in Windows, but with aliasing Helvetica appears as a font family and Narrow appears as a variation in the Character Designer and Paragraph Designer windows.

Font aliasing also makes it possible to go back and forth easily between Windows and other platforms. The Frame product automatically converts Windows fonts to their Frame product equivalents for you.

The settings under `[WindowsToFrameFontAliases]` use this syntax:

Windows-font [*angle*|*], [*weight*|*] = *Frame-font* [*angle*|*], [*weight*|*], [*variation*|*]

The Windows font is a font family available in Windows. The angle for this font can be either `Regular` or `Italic`, and the weight can be one of the weights defined in the font profile under `[Fonts]`. You can also use an asterisk (`*`) to specify no particular angle or weight.

The Frame product font is a font family available on the other platforms. The angle, weight, and variation for this family can be any of the ones defined in the font profile. If you use an asterisk (`*`), the Frame product font will use the angle, weight, or variation from the Windows font.

For example, the setting

```
HelveticaNarrow, *, * = Helvetica, *, *, Narrow
```

assigns the Windows font family Helvetica Narrow to the Frame product font name Helvetica with the Narrow variation. The two asterisks specify that angles and weights are not affected in this alias.

If you do not have an appropriate alias defined for a Windows font, the default alias is used:

Windows-font *, * = *Frame-font* *, *, *

You can add more aliases to `[WindowsToFrameFontAliases]` and change existing ones. Follow the syntax for any changes you make.

Mappings for unavailable fonts

When you try to open a document that requires fonts not available on your system, an alert box appears telling you there are unavailable fonts.

If you click OK, the Frame product opens the document and substitutes for unavailable fonts the fonts specified under `[UnknownToKnownFontMap]` in the initialization file. Initially, `[UnknownToKnownFontMap]` has a few lines of comments and some mappings for common Macintosh fonts. You can change these mappings and add others.

The mappings under `[UnknownToKnownFontMap]` must use this syntax:

unavailable-Frame-font [*angle*|*], [*weight*|*], [*variation*|*] = *available-Frame-font* [*angle*|*], [*weight*|*], [*variation*|*]

The angles, weights, and variations for these mappings can be any of the ones defined in the font profile under `[Fonts]`. You can also use an asterisk (*) to specify no particular angle, weight, or variation.

For example,

```
Lumina, *, * = Helvetica, *, *, *
Helvetica, *, Light, * = Helvetica, *, Regular, *
Helvetica, *, *, UltraCompressed = Helvetica, *, *, Narrow
```

If you open a document with unavailable fonts and don't have substitutes mapped for those fonts, the Frame product replaces them with the default fonts defined under `[Fonts]` instead.

Note that the settings under `[UnknownToKnownFontMap]` map one Frame product font to another. This is different from `[WindowsToFrameFontAliases]`, which assigns a Frame product font name to an equivalent Windows font.

Filters on your system

The initialization file contains settings for export and import filters. The Frame product provides these settings when you install filters. If you want to change any of the settings, install the filters again. For more information about filters, see the online manual *Using Frame Filters*.

Export filters

The `[Export]` settings in the initialization file contain information about the export filters installed on your system. Each setting uses this syntax:

ID = "*UI_name*" "*Clipboard_name*" *file_type*
conversion_program input_format output_format

`[Export]` settings can specify multiple programs (with input and output formats). This is an example of how an `[Export]` setting might appear in your file:

```
1= "Rich Text Format (RTF)" "RTF" DOCUMENT W4W2F.DLL MIF MSI W4W19T.DLL
MSI RTF
```

Import filters

The [Filters] settings in the initialization file contain information about the import filters installed on your system. Each setting uses this syntax:

ID = "*format_from*" "*format_to*" *file_type conversion_program extension*

This is an example of how the [Filters] settings might appear in your file:

```
100="Ami Professional" "MSI" DOCUMENT W4W33F.DLL ^.SAM
120="CGM" "WMF" OBJECT FRME_INT.DLL ^.CGM
125="CorelDRAW" "WMF" OBJECT FRME_INT.DLL ^.CDR
130="DCA/RFT" "MSI" DOCUMENT W4W15F.DLL
131="DCA/RFT - DisplayWrite 5" "MSI" DOCUMENT W4W15F.DLL
132="DIB" "DIB" OBJECT frame.exe ^.BMP
```

3 Changing Menus and Commands

Go To

Frame products are configured to display menus and commands as described in your user's manual and in online Help. You can choose to display all the menus and commands or a subset of them called quick menus. You can change both the complete and quick menu configurations by adding, removing, and rearranging menus, commands, and shortcuts. You can also create your own completely custom menu bar for document windows. And you can change the keyboard shortcuts for functions.

This chapter describes how to switch between quick, complete, and custom menus; write and use a menu customization file to change the menus and their contents; and change keyboard shortcuts. It also describes where Frame products look for menu and command configuration files when starting.

Changing menu configurations

You can display all menus and commands as they're described in your user's manual. Or you can display a subset of the menus and commands—quick menus—if you aren't using all the features. For example, the complete File menu contains the Preferences command as well as the Import and Utilities submenus, but the quick File menu does not.

You can read a menu customization file to change the menus and their contents at any time. The changes described in the file are made immediately. For example, a menu customization file might rearrange commands on a menu or put several rarely used commands on a submenu.

You can also start a Frame product with a completely custom menu bar. For details, see "Defining a completely custom menu bar" on page 27.

Using quick or complete menus

You can switch between complete and quick menus at any time.

To display quick menus, choose Menus>Quick from the View menu.

To display complete menus, choose Menus>Complete from the View menu.

Using quick menus at startup

Frame products are configured to display complete menus at startup. You can change the startup configuration by editing the `MenuSet` settings in the initialization file:

```
MenuSet = Complete
;MenuSet = Quick
```

To have quick menus appear when you start up, remove the semicolon from the `MenuSet=Quick` setting and insert a semicolon before the `MenuSet=Complete` setting. Then restart the Frame product.

Using a menu customization file

A menu customization file is a text file containing statements that change the menus and commands you see in a Frame product. For example, a customization file might change the name of a menu command or move a command from one menu to another.

To load a menu customization file, choose Menus>Modify from the View menu. Then choose a menu customization file in the dialog box that appears, and click OK.

The customizations take effect as soon as the file is read. Any errors in the customization file are reported in the Frame product console window. Each error message contains the line number in the file and a description of the problem. Even if an error is found, the rest of the file will be read.

Reading a menu customization file at startup

If you want a menu customization file to be read at startup, put the file in one of the locations that Frame products search during startup. For details, see "Search path for menu configuration files" on page 33.

Removing customizations

To remove all customizations, restart your Frame product. However, it may still read a menu customization file at startup. To prevent this, you must change settings in the initialization file. For details, see "Search path for menu configuration files" on page 33.

Creating a menu customization file

You can create a menu customization file to add, remove, rename, and rearrange menus and menu commands and to change keyboard shortcuts for commands. You can even define a completely custom menu bar. But keep in mind that changing menus and commands will make the interface inconsistent with the standard interface, so that information in printed documentation and in online Help may no longer be accurate.

In a menu customization file, you need only specify the changes you want to make—not the entire configuration. The contents of the file override the information in configuration files that were read previously.

Frame products come with sample menu customization files. For a sample file that makes a few changes to the default menus and commands, see `fminit\sample.cfg` in *`install_dir`*. For a description of the statements you can put in a menu customization file, see the rest of this chapter.

About menu customization files

A menu customization file must be a text file. If you create it with a Frame product, be sure to save it in Text Only format.

A menu customization file consists of statements. Statements are case-sensitive. Each statement is enclosed in angle brackets (< and >). Where nested angle brackets are required, every left angle bracket must have a corresponding right angle bracket. To include a right angle bracket as part of a menu or command identifier or label, precede it with a backslash (\).

Text outside angle brackets is treated as a comment. Don't include angle brackets in comments.

Statements must appear in a particular order. For example, you must define a menu before adding it to the menu bar or another menu; you must define a macro before adding it to a menu; and you must add a command to a menu before moving the command.

Statements refer to *identifiers* and *labels* for menus, commands, and macros. Frame products use the identifiers internally, and display the labels as menu and command names.

If you intend to load a menu customization file while running a Frame product, store the file with any filename you want, in any directory you want.

If you want a Frame product to read the menu customization file when it starts, specify its pathname as the value of the `ConfigCustomUI` setting in the initialization file. See "Menu and command configuration files" on page 12.

Finding menu and command identifiers

To create a menu customization file, you'll need to know the identifiers—the names that Frame products use—for existing menus and commands. You can find these identifiers by looking in the menu and command configuration files included with your Frame product.

To find the identifiers for existing menus, see the `menus.cfg` file in the `fminit\`*`product`* directory, where *`product`* represents a product-specific directory (for example, `maker` for FrameMaker).

For a list of reserved menus, see "Reserved menus" on page 32.

To find the identifiers for existing commands, see the `cmds.cfg`, `mathcmds.cfg`, and `wincmds.cfg` files in the `fminit` directory.

Debugging menu customization files

If you're writing a lengthy menu customization file, consider writing and testing the customizations a few at a time. This will make it much easier to locate problems in the statements you write.

As you create the file, you can save the file and then read it into a Frame product to test your statements.To display error messages when you load a menu customization file, set the `ShowErrors` setting in the initialization file to `On`. See "Settings for the Preferences

dialog box" on page 5. Error messages appear in the Frame console window. If you find errors, you can fix them immediately and continue writing.

When you read the same menu customization file again, you'll see error messages about redefining a command (because the same statements are being read again). Don't worry about these messages.

Use comments throughout the menu customization file to document your work. Others may need to edit the file later.

Adding a menu

You can add a menu to a menu bar. You can also add a submenu to an existing menu. For example, you may want to create a new menu for some commands you use frequently. Or you may want to add an existing menu as a submenu.

Frame products use no more than one level of submenus in their default configurations, but you can use more than one level.

If you're adding a new menu, you must first define the menu and then add it to the menu bar or to another menu as a submenu.

When you add a menu to the menu bar, it appears after the existing menus (except for the Help menu, which is always at the right of the menu bar). When you add a submenu to an existing menu, it appears at the end of the menu. After you add a menu, you can move it. For details, see "Rearranging menus or commands" on page 29.

1. **If you're creating a new menu, define it with the following statement:**

   `<Menu` *`MenuID`* `<Label` *`MenuLabel`*`>>`

   `MenuID` is the identifier used in other statements to refer to the menu. *`MenuLabel`* is the label that appears in the interface.

   The menu identifier must be unique. Make sure your menu identifier doesn't match an existing menu or command identifier. You'll find it easier to read the menu customization file later if your menu identifier uses `Menu` as a suffix. Also, you'll type the menu identifier again and again as you add commands to the menu, so keep the identifier short and easy to type.

2. **Add the menu to the menu bar or to another menu with the following statement:**

   `<Add` *`MenuID`* `<Menu` *`DestinationMenuID`*`>>`

   `MenuID` is the identifier you used in the previous step. *`DestinationMenuID`* identifies where you're adding the menu.

   The menu bar is defined as a menu, so you add a menu to the menu bar the same way you add a submenu to an existing menu.

Example
For example, to define and add a submenu labeled Other Commands to the Graphics menu, you might use the following statements:

```
<Menu ExtraGraphicsCommands <Label Other Commands>>
<Add ExtraGraphicsCommands <Menu ToolsMenu>>
```

Defining a completely custom menu bar
To define a completely custom menu bar, use the following statement:

```
<ReservedMenu !CustomMakerMainMenu <Label MyOwnMenus>>
```

Then add menus to the menu bar as described above.

To switch to a custom menu bar while a Frame product is running, use the Menus>Modify command on the View menu to read the menu customization file that defines the menu bar.

If you want a completely custom menu bar to appear on document windows at startup, save the menu customization file with a filename and in a directory that Frame products look for at startup. (For more information, see "Search path for menu configuration files" on page 33.) In the initialization file, change the `MenuSet` setting to `Custom`. Then restart the Frame product.

Adding a command to a menu

You can add a command to an existing menu or to a menu you have added. For example, you may want to add commands to the Graphics menu. Or you may want to place several commands on a new submenu.

You can also specify a Shift command—a command that replaces another command when you display the menu while holding down the Shift key. For example, when you hold down Shift and pull down the File menu, the Save command appears as Save All Open Files.

You can't add a command directly to the menu bar. The command must be on a menu.

When you add a command, it appears at the bottom of the menu. After you add a command, you can move it. For details, see "Rearranging menus or commands" on page 29.

To add a command to a menu, use the following statement:

```
<Add CommandID <Menu MenuID>>
```

CommandID is the identifier of the command you're adding. *MenuID* is the identifier of the menu to which you want to add the command.

To add a Shift command to a menu, use the following statement:

```
<ShiftCommand UnshiftedCommand ShiftedCommand>
```

UnshiftedCommand is the identifier of the command as it normally appears. *ShiftedCommand* is the identifier of the command you want to appear when you hold down the Shift key.

Examples

To add the Find command to the Special menu, you might use the following statement:

```
<Add Find/Change <Menu SpecialMenu>>
```

If you want the Ungroup command to appear in place of the Group command on the Graphics menu when you press Shift, you might use the following statement:

```
<ShiftCommand GraphicsGroup GraphicsUngroup>
```

Separators between commands on a menu

To add a separator line to a menu, use the following statement:

```
<Add !Separator <Menu MenuID>
```

MenuID is the identifier of the menu to which you want to add a separator. After you add a separator, you can't move it to a new location. If you want to move it, you need to move the surrounding commands instead. For details, see "Rearranging menus or commands" on page 29.

Creating a new command

You can create a new command on a menu. Each command must correspond to one or more existing function codes. The command identifier must be one word. For sample commands, see the `cmds.cfg` file in `fminit`.

To create a new command, you'll need to use the Application Program Interface (API). For information, see the *FDK Programmer's Guide,* available separately.

Removing a menu or command

You can remove menus from the menu bar and commands from a menu. However, you cannot remove a reserved menu. For a list of reserved menus, see "Reserved menus" on page 32.

Removing an item (menu or menu command) doesn't remove the functions or the associated keyboard shortcuts from Frame products. Only the menu changes.

Removing a menu or menu command also doesn't affect other commands, even though they may seem to be related. For example, if you remove the Group command, the Ungroup command doesn't disappear.

To remove a menu or command, use the following statement:

```
<Remove MenuItemID <Menu MenuID>>
```

MenuItemID is the identifier of the item that you're removing. *MenuID* is the identifier of the menu containing the item you want to remove.

If a command appears on more than one menu and you want to remove it from each menu, use one statement for each menu.

Examples

To remove the entire Special menu from both the quick and complete menus, use the following statements:

```
<Remove SpecialMenu <Menu !MakerMainMenu>>
<Remove QuickSpecialMenu <Menu !QuickMakerMainMenu>>
```

To remove the Reshape command from the Graphics menu in the quick menus, use the following statement:

```
<Remove GraphicsReshape <Menu QuickToolsMenu>>
```

Removing formatting bar commands and window buttons

You can remove items from the formatting bar at the top of the document window. You can also remove the buttons from the upper-right corner of the document window.

To remove the Paragraph Format pop-up menu, use the following statement:

```
<Remove !ShowRulerParagraphTags <Menu !RulerControlMenu>>
```

To remove the Alignment and Spacing pop-up menus and the tab wells, use the following statement:

```
<Remove !ShowRulerAlignmentSpacingAndTabs
    <Menu !RulerControlMenu>>
```

To remove the buttons from the upper-right corner of the document window, use the following statement:

```
<IconBarOn No>
```

To replace the buttons, use the following statement:

```
<IconBarOn Yes>
```

Rearranging menus or commands

You can rearrange menus on the menu bar and commands on a menu. You can also move a command from one menu to another. However, you cannot move the Help and Window menus.

In the statements below, *MenuID1* is the identifier of the menu that contains the item (menu or command) you want to move. *MenuItemID* is the identifier of the item you want to move. *MenuID2* is the identifier of the menu on which you want the item to appear. And *CommandID* is the identifier of the item you want the command to precede or follow.

To put an item first or last on a menu, use one of the following statements:

```
<Order MenuID1.MenuItemID <First MenuID2>>
<Order MenuID1.MenuItemID <Last MenuID2>>
```

To put one item before or after another item on a menu, use one of the following statements:

```
<Order MenuID1.MenuItemID <Before MenuID2.CommandID>>
<Order MenuID1.MenuItemID <After MenuID2.CommandID>>
```

Examples

To put the Footnote command first in the Special menu, use the following statement:

```
<Order SpecialMenu.Footnote <First SpecialMenu>>
```

To put the Special menu last on the menu bar (for complete menus), use the following statement:

```
<Order !MakerMainMenu.SpecialMenu <Last !MakerMainMenu>>
```

To put the Thesaurus command above the Find/Change command in the Edit menu, use the following statement:

```
<Order EditMenu.Thesaurus <Before EditMenu.Find/Change>>
```

To put the Graphics menu after the Format menu (for complete menus), use the following statement:

```
<Order !MakerMainMenu.ToolsMenu <After !MakerMainMenu.FormatMenu>>
```

Renaming a menu or command

You can change the name (label) of a menu that appears in the menu bar or a command that appears on a menu. You can change the label of a command everywhere it appears, or in only one of several places.

To rename a menu or to rename a menu command everywhere it appears, use the following statement:

<Modify *ItemID* <Label *NewLabel*>>

ItemID is the identifier of the menu or command you want to rename. *NewLabel* is the new label.

To rename a menu command in only one place, define a new command that duplicates the function of the old one (using the same key sequence, definition, and mode), but use a different label. Then put the new command on the menu in place of the old one. For information on finding command definitions, see "Menu and command configuration files" on page 33. For information on adding and removing commands on menus, see "Adding a command to a menu" on page 27 and "Removing a menu or command" on page 28.

Example

To rename the Pages submenu on the Format menu to Page Design, use the following statement:

```
<Modify PagesMenu <Label Page Design>>
```

Editing a command with a <ReservedLabel> statement

Some commands have a different label, and a different effect, depending on the current state—whether the Shift key is held down when the command is chosen, where the insertion point is or what is selected, and so on. In these cases, a command has several labels. Each label has an identifier, in a `<ReservedLabel>` statement. For example:

```
<ReservedLabel Save Save (not needed)>
<ReservedLabel SaveNeeded Save (needed)>
```

To rename these commands, include a `<ReservedLabel>` statement in the `<Modify>` statement, and change only the label. For example:

```
<Modify Save <ReservedLabel SaveNeeded Save (good idea)>>
```

Adding or changing a keyboard shortcut

You can add two types of shortcuts to commands: mnemonic shortcuts and key sequences.

Adding mnemonic shortcuts

A mnemonic shortcut is an underlined letter in a command label. When the menu is open, you can press the corresponding key to execute the command. For example, when the File menu is open, you can press x to exit the Frame product.

To specify a mnemonic shortcut in a command configuration file, insert the following statement within the statement that defines the command:

```
<Label CommandLabel>
```

CommandLabel specifies the command label that appears on a menu. Add the *&* character before the letter you want to use as the shortcut. For example, the following statement defines the letter Q as a mnemonic shortcut for the QuickAccess Bar command on the View menu:

```
<Label &QuickAccess Bar>
```

Adding key sequences

You can add a key sequence for any command. You can also change a shortcut that appears on a menu or change just the label on the menu.

For example, you might want to add Control-a Control-c as a shortcut for the Copy command. You might also want the new shortcut to appear on the menu.

Adding a shortcut for a command or changing the shortcut that appears on a menu doesn't disable other shortcuts for a command. As long as the shortcut is defined in a configuration or menu customization file, it will work.

In the statements below, *CommandID* is the identifier of the command for which you want to add a keyboard shortcut. *Sequence* is the keyboard shortcut for the command. For information on how to represent the key sequence, see "Specifying the key sequence for a shortcut," next.

To add a keyboard shortcut for a command:

```
<Modify CommandID <KeySequence Sequence>>
```

To change the label that appears for a shortcut on a menu without changing the shortcut itself, use the following statement:

```
<Modify CommandID <KeySeqLabel Label>>
```

Label is the label you want to appear next to the command on a menu. The label changes everywhere the command appears on a menu.

Specifying the key sequence for a shortcut

You use keysyms—abbreviations that represent specific keys on the keyboard—when specifying the key sequence. A slash (/) precedes the keysym name. You also use abbreviations for modifier keys—for example, + for Shift, ^ for Control, and ~ for Alt. Finally, the command prefix, normally Esc, is represented in the command configuration files as a backslash followed by an exclamation point (\!).

Valid keysyms are Up, Down, Left, Right, Home, End, PgUp, PgDn, Return, Tab, Bksp, Space, Delete, Insert, F1 through F16, and Escape.

Example

To add the shortcut Control-a Control-c for the Copy command and to display *Ctrl-a Ctrl-c* instead of *^a^c* as the shortcut, use these statements:

```
<Modify Copy <KeySequence ^a^c>>
<Modify Copy <KeySeqLabel Ctrl-a Ctrl-c>>
```

Displaying a menu command without a shortcut

To display a menu command with no shortcut, remove the `<KeySeqLabel>` for that command.

Removing a keyboard shortcut

To remove a keyboard shortcut, delete its definition from the command configuration files, and then restart the program.

Shortcuts on the Equations and Layout palettes

The shortcuts that appear on the Equations and Layout palettes won't change to reflect any customizations. These palettes are actually view-only documents containing hypertext commands.

Reserved menus

Frame products expect to find some menus called reserved menus. Except for the Help menu, you can't remove them.

Menu ID	Description
!MakerMainMenu	Menu bar for complete menus (document window active)
!QuickMakerMainMenu	Menu bar for quick menus (document window active)

Menu ID	Description
!CustomMakerMainMenu	Menu bar for custom menus (document window active)
!BookMainMenu	Menu bar for complete menus (book window active)
!QuickBookMainMenu	Menu bar for quick menus (book window active)
!ViewOnlyMainMenu	Menu bar for view-only document
!HelpMenu	Help menu
!RulerControlMenu	Formatting bar
!RulerAlignMenu	Alignment pop-up menu in the formatting bar
!RulerSpaceMenu	Spacing pop-up menu in the formatting bar
!RulerParaMenu	Paragraph Formats pop-up menu in the formatting bar

Menu and command configuration files

The contents of menus are determined by the `menus.cfg` file in `fminit\`*`product`*. This file specifies the complete and quick menus that appear in menu bars. It can also specify whether command buttons appear in the upper-right corner of the document window.

Each menu command corresponds to a built-in function. A command can also correspond to a call to a program that uses the Frame Application Program Interface (API). For more information, see the *FDK Programmer's Guide,* which is included in the Frame Developer's Kit.

Frame product functions and keyboard shortcuts are defined in several files in the `fminit` directory. The `cmds.cfg` file contains commands and shortcuts that are the same on all platforms. The `mathcmds.cfg` file contains the commands and platform-independent keyboard shortcuts for equations. The `wincmds.cfg` file contains the commands and shortcuts that are specific to the Windows version of Frame products. Each command definition specifies a unique identifier, keyboard shortcuts, and a corresponding function code that executes the command.

Search path for menu configuration files

When a Frame product starts, it first reads menu and command configuration files and then it reads menu customization files. The information in each file overrides the information in files read previously.

Standard menu and command configuration files

Frame products read the standard menu and command configuration files in the Frame product installation directory in the following order:

- `fminit\cmds.cfg`
- `fminit\mathcmds.cfg`
- `fminit\wincmds.cfg`
- `fminit\`*`product`*`\menus.cfg`

- `fminit\customui.cfg`

You can instruct a Frame product to read different files by changing settings in the product's initialization file. For instructions, see “Menu and command configuration files” on page 12.

Index

Go To

To go to a page, click on a page number below.

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z