MML Reference

Table of Contents

Introduction

MML files

Using MML to create FrameMaker documents

Specifying document with a FrameMaker template

Specifying document format with MML

Relative Merits of Each Approach

MML As An Automated Conversion Tool

MML Statements

MML file structure

Markup statement overview

Control and macro statements

Format Definitions

Font Statements

Paragraph Statements

RuleIncluded Statements

Variable Definitions

Document Layout Statements

Document Text Statements

- Anchored Frames
- Markers

- Multiple TextFlows
- Text as Typed
- Cross-reference
- Footnotes
- Using Variables
- Tables
- Math Statements

Samples

Specifying document format with a FrameMaker template

- Include file: ChapterFormats.mml
- Document content file: SampleChapter.mml

Specifying document format with MML

- Include file: document format information
- Include file: document information
- Document content file

MML Interpreter Messages

[Home Page | Download Postscript | Custom API | Warranty | Copyright | Forward]

Introduction

FrameMaker® software from Frame Technology® includes support for a markup language called MML (Maker Markup Language). You can use any standard text editor to create an MML file. Later, you can open the MML file as a FrameMaker document or import it into a FrameMaker template. In a department where different people are responsible for writing and formatting documents, writers can use MML statements to mark up manuals in progress; at the same time, graphic designers can create the formatting specifications in FrameMaker templates.

This manual documents a number of enhancements to the MML syntax that were developed by Softline International Inc. The extensions were developed for two reasons:

- To keep the MML syntax current with changes that were added in Release 3.0 of FrameMaker.
- To provide extensions to the MML language that would facilitate its use by developers of filters.

MML supports many formatting and layout features of FrameMaker. For example, you can use MML to specify:

- Document page size and column layout (including options available with the Custom Document command).
- Header and footer layouts.
- Paragraph Format Catalog (including all options available in the Paragraph Format window).
- Font definitions (including all options available in the Character Format window).
- Table Format Catalog (including all options available in the Table Format window).
- Rule Definitions
- Document text with varying character and paragraph formats.
- Anchored frames containing textflows and graphics.
- Markers (including all options available in the Marker window).
- Cross-references.
- Footnotes.
- Variables.
- Tables
- Specify external files which contain page defintions, paragraph catalogs, font catalogs,

cross-reference formats, and variable definitions in MIF format. These files are automatically merged as the document is opened in FrameMaker for the first time.

MML cannot define the following (you can add them after you open or import the MML file with FrameMaker):

- Unanchored frames and graphics.
- Irregular column layouts (unless the external definition file is used).
- Column layouts that vary from page to page (unless the external definition file is used).
- Multiple-line or multiple-font headers and footers (unless the external definition file is used).
- Multiple master pages (unless the external definition file is used).

This manual contains:

- General instructions for creating and using MML files.
- A complete description of each MML statement.
- Sample MML files.
- MML error messages.

MML files

An MML file is a standard ASCII text file containing MML statements and document text. You can create the file with any standard text editor.

Important: If you use FrameMaker to create or edit an MML file, save the file as Text Only using the Save As command.

An MML filename must end with .mml. This suffix alerts FrameMaker that the file is an MML file and needs to be interpreted (by the mmltomif program) before it is imported into, or opened as, a FrameMaker document. For more information on how MML files are processed, see Appendix D of *FrameMaker Reference.*

After you open or import an MML file, you can modify, print, and save it using FrameMaker commands. However, any changes you make in FrameMaker are not reflected in the original MML file. Thus, if you want the MML file to serve as the master source for the document, you must make the changes to the MML file.

Using MML to create FrameMaker documents

You can use MML to:

- Specify the content of a document for which formatting information is stored in a FrameMaker template.

  You use a small subset of MML instructions to specify when to use paragraph formats and when to change the character format for words and phrases. You create the paragraph formats and set up the document layout in a FrameMaker template.
- Specify both the content and format of a document.

- You use more complex MML statements to define formatting and layout specifications in the MML file.

When you use MML to create FrameMaker documents, use two MML files to describe a document; an MML include file contains formatting information, and an MML document content file contains document text. Using two files makes it easier to correct errors. In addition, you can use one include file to create several documents with the same formatting.

If you use a FrameMaker template to specify formatting, your include file can be very brief. It lists the paragraph formats in the template's Paragraph Format Catalog and any character formats and MML macros you want to use.

If you want to keep formatting information and document content in one file, the file should contain the information that would appear in an include file followed by the information that would appear in a document content file.

Specifying document format with a FrameMaker template

The easiest way to use MML is to specify formatting information in a FrameMaker template. In addition to the template, you use a simplified include file and a document content file to specify the document text.

For a complete description of the sections in an MML file, see MML file structure. For a sample FrameMaker document created from a template, an include file, and a document content file, see Specifying document format with a FrameMaker template.

Setting up the template

In the FrameMaker template, set up the document layout and create paragraph formats.

Creating the include file

Use a standard text editor to create the include file. It should contain:

- An MML identification line.
- A Macro Definition section.
- A <DefineTag> statement for each paragraph tag in the template. (See Paragraph statements.)
- Font definitions for character format changes to be used for words or phrases in the document content file.

Creating the MML document content file

Use a standard text editor to create the document content file. It should contain:

- An MML identification line.
- An <Include> statement that names the MML include file. (See Control and macro statements.)
- A Document Text section.

Importing the MML file into the template

To create a FrameMaker document from the MML file, use the FrameMaker New command to create a new document from a template. Then use the Import command to import the MML document content file into the document. Use the Save As command to save the resulting document under a new filename.

Specifying document format with MML

When you use an include file and a document content file to create a FrameMaker document, the include file describes document formatting; the document content file contains the document text.

For a complete description of the sections in an MML file, see MML file structure.

Creating an include file
Use a standard text editor to create an include file describing document formatting. It should contain the following sections:

- An MML identification line.
- Macro Definition section.
- Font Definition section.
- Paragraph Format Definition section.
- Document Layout Description section.

Creating a document content file
Use a text editor to create the document content file. It should contain:

- An MML identification line.
- An <Include> statement that names the MML include file. (See Control and macro statements.)
- A Document Text section.

Opening the document content file
To create a FrameMaker document from the include and document content files, open the document content file with FrameMaker.

Relative Merits of Each Approach

Users are often unclear as to when to use which approach. If you are doing relatively small documents or only have a few documents to import into FrameMaker either approach will do.

However, if have large numbers of documents to convert to FrameMaker the approach you use can save considerable manual effort during the conversion. If you can fully define your document using the MML syntax then you should use the second approach. The intial opening and saving of documents should be done using the FMBatch utility since opening and saving relatively large (more than 50 pages) can consume at least five minutes even on the fastest of processors. Setting up a batch job to open MML files and save them in FrameMaker allows you to put your time to better use while your documents are being converted.

If your documents are not fully definable in MML or you want to the document to take on a differrent format when it is opened in FrameMaker then you must prepare a template and import the document into the template. This process is even more time consuming than opening an MML document directly. FMBatch should always be used in this situation to process your files in batch mode.

Unfortunately not all platforms which support FrameMaker also support the FMBatch utility (the Apple

MacIntosh for example). To increase your productivity in these situations, Softline has provided a number of MML statements which allow you to automatically include format files which you have seperately prepared in MIF format.

MML As An Automated Conversion Tool

Most FrameMaker documentation which discusses the writing of filter software encourages developers to write their filter programs to produce output in MIF format. If you are developing filter software there are a number of reasons why you should consider developing your filter to produce MML instead of MIF:

- The MIF syntax is very awkward to use and requires very precise formatting of your output file. For example each line of text must be enclosed in a ``stirng" statement.
- MML allows you to output your document in a natural sequence. Various document components are in the same order that they would normally appear on the printed page. MIF requires document components to be in a special sequence that often requires extensive sorting as a part of the filter process.

[Home Page | Download Postscript | Custom API | Conversion Services | Previous | Forward]

MML Statements

An MML file consists of markup statements and document text. Markup statements begin with a left angle bracket (<) and end with a balancing right angle bracket (>). For example, <Section> signals the beginning of a new section, while <Family Times> switches fonts. Case is not significant in statement names, so <FaMiLy Times> would work as well.

All text outside angle brackets is document text. within document text, adjacent nonblank lines are considered to be in the same paragraph; blank lines separate paragraphs. <Paragraph> markup statements can also signal paragraph boundaries. (See Paragraph statements.)

If the text contains a left or right angle bracket character (that is, one that should appear in the FrameMaker document instead of beginning or ending a markup statement), a backslash character must precede the angle bracket (for example, \< or \>).

MML file structure

An MML file can contain the following sections, in this order:

Section	**Contains**
MML identification line	An <MML> statement identifying the file as an MML file.
Format Definitions	A series of statements which supply the names of externally maintained format files which are to be included in your document.
Macro Definition	<!DefineMacro> and <!DefineChar> statements that define simple macros used in subsequent markup statements.
Font Definition	<!DefineFont> and <!DefineFTag> statements that define named sets of font attributes for use within document text and other markup statements.
Paragraph Format Definition	<!DefinePar> and <!DefineTag> statements that define paragraph formats.
Table Format Definition	<!DefineTable> and <!DefineTTag> statements that define table formates.
Rule Format Definition	<!DefineRule> statements that define rule formats.
Variable Definitions	<!DefineVar> that provide substitution values for any variables which are used in the text portion of the document.
Document Layout	Document layout statements that define document properties.
Document Text	ASCII characters along with font, special character, anchored frame, markers, cross-references and paragraph-related markup statements. the first text character outside a markup statement signals the beginning of the document text section.

All sections except the Document Text section are optional. For information about which sections to include in an MML file, see Using MML to create FrameMaker documents.

Markup statement overview

The general format of a markup statement is:

`< StatementName OptionalDataItems >`

The following conventions are used to describe the format of data items:

char	A single character (or a backslash equivalent, such as \t for tab).
string	Any sequence of characters enclosed by double quotation marks ("abc"). To use double quotation marks in a string, type \"
commentstring	Any sequence of characters.
name	A simple alphanumeric string starting with a nonspace character and ending with a space or >.
number	An integer.
boolean	Yes, No, Y, or N. Case is not significant.
meansure	A real number, which may contain a decimal point and digits to the right of the decimal point, followed by an optional unit of meansurement: " (for inch), pt, cm, mm, or pica.
unit	inch, in, cm, mm, pica, pc, or pt.
lrcd	left, l, right, r, center, c, leftright, lr, decimal, or d.

Control and macro statements

You use control and macro statements to set up your MML file. the following statements (except for the <MML> statement) can appear anywhere in the file.

<MML *commentstring*>
All MML files must begin with an <MML> statement.

<Comment *commentstring*>
Places comments in the MML file. the comment string is ignored when the file is read.

<!DefineMacro *name string*>
Replaces all occurrences of the newly defined statement name with the string. Note that string is rescanned each time name is encountered. For example, suppose your text editor does not give you a way to type the Yen symbol. You can use the statement <!DefineMacro Yen "<Character \\xB4 >"> to define a new MML statement, <Yen>, that represents the FrameMaker Yen character (0xB4). When you import or open your document in FrameMaker, MML replaces each <Yen> statement with a Yen character.

Important: MML uses a backslash (\) in place of the initial 0 in the character code. tdus the MML representation of the character 0xB4 is \xB4.

<!DefineChar *char string*>
Works like <DefineMacro>, except that char is any single character and is not in angle brackets (< >). Use this statement to remap character codes for foreign and other special keyboards. For instance, suppose that the Yen symbol is represented by character code 0xFE in the MML file, but FrameMaker considers character 0xB4 to be the Yen symbol. this statement:

<!DefineChar \xFE "<Character \\xB4 >" >

causes MML to turn all 0xFE's into Yens for FrameMaker.

<Include *string*>

Reads the file named in string as MML input. If the include filename is not a full patdname (one that starts with / or ~), the mmltomif program has a number of options to control where it searches for the include file. When you open or import an MML file as a FrameMaker document, mmltomif searches for include files in the directory containing the MML file being processed. You can also run mmltomif directly to access other include file options. For more information, see Appendix D of FrameMaker Reference.

<Units *unit*>
Establishes default units for all meansurements. If a <Units> statement appears in the file, it must come before the font definitions section. (Default = inch.)

<!Alias *newname currentname*>
Creates a new statement name that is a synonym for an existing statement name. For example, you could define the synonym <!Alias !MD !DefineMacro> and then define the macro <!MD bi "<Bold><Italic>">. You could then use <bi> within the document text to set the current font to bold italic.

<EndOfInput>
Ignores all remaining text in the MML file. Use <EndOfInput> to debug an MML file or to temporarily modify an MML file so that FrameMaker reads only part of it.

<Message string>
Prints the specified string in the shell window where FrameMaker was started. Use <Message> to debug an MML file.

Format Definitions

MML format definition statements allow you to supply names of externally maintained format definition files. these files must be in MIF (Maker Interchange Format) format. the various files are included in your document at the appropriate location. Because the files are in MIF format, you should be very familiar with coding MIF before attempting to use this feature. the feature is intended for use in high volume conversion situations where the user does not have a version of FMBatch (such as the MacIntosh environment) available to allow importing of files into predefined formats. these statements may be in any order but must appear after the MML identification line and before the font definition section.

<PgfInclude *name*>
Specifies the name of an MIF file containing paragraph definitions that are to be automatically merged into this document as it is converted. the format of the name portion of the statement is dependent on your particular operating environment. Do not include the <PgfCatalog> statement in your MIF file since one is supplied by the program.

<FontInclude *name*>
Specifies the name of an MIF file containing font definitions that are to be automatically merged into this document as it is converted. the format of the name portion of the statement is dependent on your particular operating environment. Do not include the <FontCatalog> statement in your MIF file since one is supplied by the program.

<VarInclude *name*>
Specifies the name of an MIF file containing variable formats that are to be automatically merged into this document as it is converted. the format of the name portion of the statement is dependent on your particular operating environment. Do not include the <VariableFormats> statement in your MIF file since one is supplied by the program.

<XRefInclude *name*>
Specifies the name of an MIF file containing cross-reference formats that are to be automatically merged into this document as it is converted. the format of the name portion of the statement is dependent on your particular operating environment. Do not include the <XRefFormats> statement in your MIF file since one is supplied by the program.

<PageInclude *name*>
Specifies the name of an MIF file containing page formats that are to be automatically merged into this document as it

is converted. the format of the name portion of the statement is dependent on your particular operating environment. If you supply this statement you must also supply a <TextInclude> statement.

<TextInclude *name*>
Specifies the name of an MIF file containing page header text that is to be automatically merged into this document as it is converted. the format of the name portion of the statement is dependent on your particular operating environment. If you supply this statement you must also supply a <PageInclude> statement.

Font statements

MML font statements provide character format control similar to the control provided by the FrameMaker Character command.

Most of the font statements can appear in the Font Definition, Paragraph Format Definition, Document Layout, and Document Text sections of an MML file. However, <!DefineFont> can appear only in the Font Definition section.

<family *name*>
Changes font family. the PostScript® font families included with FrameMaker are Courier, Times®, Helvetica®, Symbol, AvantGarde®, Bookman®, New Century Schoolbook, Palatino®, ZapfChancery®, and ZapfDingbats®. Case is significant in the family name. the complete list of family names comes from a configuration file. (See Appendix D of *FrameMaker Reference.*) (Default = Times.)

<italic>
<noitalic>
<bold>
<nobold>
<underline>
<nounderline>
<strike>
<nostrike>
<oblique>
<nooblique>
<overline>
<nooverline>
<changebar>
<nochangebar>
These statements turn on and off various font styles.

<plain>
Same as <nobold> <noitalic> <nounderline> <nostrike> <nooverline> <nochangebar>. (Default style = plain.)

<superscript>
<subscript>
<normal>
These statements change the relative position of the text baseline. they work like the other font properties. For example:

```
e<superscript>i*pi<normal>=-1
```

gives

e*i*pi*=-1

<pts *number*>
Changes font size. For example, <pts 10> changes to 10pt. (Default = 12.)

<!DefineFont *name fontstatements*>
Defines a character format. It executes the list of fontstatements (the statements defined in this section) and then establishes the current font properties as the character format. For examples of <!DefineFont> and its use, see Appendix A, Samples.

The character formats you define in an MML file are used to indicate font changes for words and phrases. However, they do not correspond to the formats stored in a document's Character Format Catalog.

Paragraph statements

MML's paragraph statements provide paragraph formatting control similar to the control provided by the FrameMaker Paragraph command.

Most paragraph statements can appear within the Paragraph Format Definition section and between paragraphs in the Document Text section. Exceptions are <!DefinePar> and <!DefineFTag>, which can appear only in the Paragraph Format Definition section.

<par>
Ends a paragraph. the current font properties and paragraph settings remain in effect. Two or more consecutive new lines act as a <par> statement. A new paragraph begins only when a real character is seen. the <par> statement is most useful within macro definitions.

<LeftIndent *measure*>
Changes the paragraph left indent. Only the value in effect at the start of the paragraph text affects that paragraph. (Default = 0".)

<RightIndent *measure*>
Changes the paragraph right indent. Only the value in effect at the start of the paragraph text affects that paragraph. (Default = 0".)

<FirstIndent *measure*>
Sets the left indent for the first line of a paragraph. (Default = .25".)

<SpaceBefore *measure*>
Sets the white space above the paragraph. Usually specified in points, as in 6pt. (Default = 0pt.)

<SpaceAfter *measure*>
Sets the white space below the paragraph. (Default = 0pt.)

<Leading *measure*>
Determines the space between lines within the paragraph. (Default = 2pt.)

Important: <SpaceBefore>, <SpaceAfter>, and <Leading> values are assumed to be in the current default unit, not necessarily in points. If you specify <Leading 2>, this means 2 units in the current default unit, which is usually inches. So <Leading 2> would put 2 inches of leading between each line.

<LineSpacing *fixed* | *proportional* | *floating*>
Sets the line spacing calcualtion metdod. Fixed sets line spacing according to paragarph font. Proportional sets line spacing according to the largest font in the line. Floating sets line spacing according to the largest ascender in the line(Default = fixed.)

<Alignment *lrc*>
Sets the alignment of paragraph lines: left (l), right (r), center (c), or left and right (lr). (Default = lr.)

<Language *string*>
Hyphenation and spell checking language (Default = "NoLanguage".)

<AutoNumber *boolean*>
Sets automatic numbering of paragraphs. If <AutoNumber Yes> is specified, there must also be a valid <NumberFormat> string. (Default = No.)

<NumberFormat *string*>
Determines the numbering format for paragraphs that are automatically numbered. Ignored unless <AutoNumber Yes> is specified. (Default = " ".)

MML version 2.0 now supports the paragraph number formats of version 2.0 of FrameMaker

<NumberFont *string*>
Determines the font of the numbering format for paragraphs that are automatically numbered. Ignored unless <AutoNumber Yes> is specified. (Default = " ".) Must be a font tag from the font catalog.

<NumAtEnd *boolean*>
Place paragraph number at end of line instead of beginning.

<Hyphenate *boolean*>
Turns automatic hyphenation on (Yes) or off (No). (Default = No.)

<ColumnTop *boolean*>
Sets the starting place for the paragraph on the page. If <ColumnTop Yes> is specified, the paragraph starts at the top of a column; otherwise it starts anywhere. (Default = No.)

<WithNext *boolean*>
Determines whether or not to keep the paragraph in the same column as the beginning of the next paragraph. (Default = No.)

<WithPrevious *boolean*>
Determines whether or not to keep the paragraph in the same column as the ending of the previous paragraph. (Default = No.)

<Tolerance *number*>
Specifies the maximum number of adjacent lines that may be hyphenated. (Default = 4.)

<Blocksize *number*>
Specifies the minimum number of widow and orphan lines. (Default = 1.)

<TabStopType *lrc*d>
Establishes the tab type for all subsequently defined tab stops until the next <TabStopType> statement: left (l), right (r), center (c), or decimal (d). (Default = l.)

<TabStopLeader *char*>
Establishes the tab leader character for all subsequently defined tab stops until the next <TabStopLeader> statement. (Default = " ".)

<TabStops <TabStop *dimension*> <TabStop *dimension*> >
Defines a set of tab stops. Each <TabStop> substatement sets a tab stop at the position indicated. the tab type and associated leader character are determined by the most recent <TabStopType> and <TabStopLeader> statements, which may be freely intermingled among the <TabStops> substatements.

To clear all tabs, use an empty tab stop list: <TabStops>.

<CellAlignment *Top* | *Middle* | *Bottom*>
When the paragraph is used within a table cell specifies the vertical alignment of the paragraph when the paragraph is the first paragraph within the cell.

<CellMargins *L R T B*>
When the paragraph is used within a table cell specifies width of default cell margis in a table.

<CellLMarginFixed *boolean*>
When the paragraph is used within a table cell specifies relationship of the left cell margin for this paragraph with the left cell margin given in the table definition. Yes means width of left cell margin is relative to <TCellMargins>given in the table definition; No means width of left cell margin overrides <CellMargins> given in the table definition.

<CellTMarginFixed *boolean*>
When the paragraph is used within a table cell specifies relationship of the top cell margin for this paragraph with the top cell margin given in the table definition. Yes means width of top cell margin is relative to <TCellMargins>given in the table definition; No means width of top cell margin overrides <CellMargins> given in the table definition.

<CellRMarginFixed *boolean*>
When the paragraph is used within a table cell specifies relationship of the right cell margin for this paragraph with the right cell margin given in the table definition. Yes means width of right cell margin is relative to <TCellMargins>given in the table definition; No means width of right cell margin overrides <CellMargins> given in the table definition.

<CellBMarginFixed *boolean*>
When the paragraph is used within a table cell specifies relationship of the bottom cell margin for this paragraph with the bottom cell margin given in the table definition. Yes means width of bottom cell margin is relative to <TCellMargins>given in the table definition; No means width of bottom cell margin overrides <CellMargins> given in the table definition.

<!DefinePar name *parstatements*>
Executes the list of parstatements (the statements defined in this section) and then establishes the current paragraph settings as a named paragraph format.

When FrameMaker opens or imports an MML document, the resulting FrameMaker document contains a Catalog entry for each paragraph format defined in the MML file using <!DefinePar> statements. When an MML paragraph whose format is name is brought into FrameMaker, its tag is name. For examples of the <!DefinePar> statement and its use, see Appendix A, Samples.

<!DefineTag *name*>
Establishes a paragraph format name like <!DefinePar>. However, unlike <!DefinePar>, <!DefineTag> does not generate a Paragraph Format Catalog entry when the MML document is brought into FrameMaker.

Use <!DefineTag> when you intend to import an MML file into a FrameMaker
document that already has the Paragraph Format Catalog set up. When an MML paragraph preceded by a <!DefineTag> statement is brought into FrameMaker, the document's Paragraph Format Catalog is searched for a format with a matching tag. If such a format exists, the paragraph's format is set to match the corresponding format in the Catalog.

Important: MML tags cannot have a space in them (although tag names in FrameMaker can).

RuleInclude statements

MML's rule statements provide rule formatting control similar to the control provided by the FrameMaker Rule command.

Rule statements can appear only within the rule Format Definition.

<RulePenwidth *measure*>
Ruling line tdickness. (Default = 1pt.)

<RuleGap *measure*>
Gap between double ruling lines. (Default = 0pt.)

<RuleSeperation *number*>
Specifies spot color of the ruling line. (Default = 0.)

<RulePen *number*>
Specifies pen pattern of the ruling line. (Default = 0.)

<RuleLines *number*>
Specifies number of lines in the rule. (0 = none) (1 = single) (2 = double) (Default = 1.)

<!DefineRule *name rulestatements*>
Executes the list of *rulestatements* (the statements defined in this section) and then establishes the current rule settings as a named rule format.

When FrameMaker opens or imports an MML document, the resulting FrameMaker document contains a Catalog entry for each rule format defined in the MML file using <!DefineRule> statements. When an MML rule whose format is name is brought into FrameMaker, its tag is *name*. For examples of the <!DefineRule> statement and its use, see Appendix A, Samples.

<!DefineRTag *name*>
Establishes a rule format name like <!DefineRule>. However, unlike <!DefineRule>, <!DefineRTag> does not generate a Rule Format Catalog entry when the MML document is brought into FrameMaker.

Use <!DefineRTag> when you intend to import an MML file into a FrameMaker
document that already has the Rule Format Catalog set up. When an MML rule preceded by a <!DefineRTag> statement is brought into FrameMaker, the document's rule Format Catalog is searched for a format with a matching tag. If such a format exists, the rule's format is set to match the corresponding format in the Catalog.

Important: MML tags cannot have a space in them (although tag names in FrameMaker can).

Variable Definitions

This statement allows the user to supply substitution values to be used whenever a particular variable is encountered in the text of the document. Remember FrameMaker allows you to set the value of a variable only once per document unlike some document processors which allow the value of variable to change constantly tdroughout the document.

<!DefineVar *variableSubstatements*>
Use the following <variable> substatements to describe the variable's settings:

<VariableName *string*>	Specifies the variable name.
<VariableDef *string*>	Specifies the variable substitution text.

the variable statement is actually in MIF format. For more information, see the variable command in Chapter 1 of *FrameMaker Reference*.

Table statements

MML's table statements provide table formatting control similar to the control provided by the FrameMaker Table command.

Most table statements can appear within the Table Format Definition section and between table entities in the Document Text section. Exceptions are <!DefineTbl>, and <!DefineTTag>, which can appear only in the Table Format Definition section. the sequence of <!DefineTbl> and <!DefineCol> statements is important because many of the

<!DefineTbl> substatements make reference to column numbers and the MMLtoMIF program edits that the column numbers used are valid. the proper sequence is a follows:
<!DefineTbl> statement
<!DefineCol> statement
<!DefineCol> substatements
<!DefineCol> statements repeated as required until all columns defined
<!DefineTbl> substatements

There are also a number of statements which control the placement of tables within the document itself and the placement of text within the table. these statements are described in ``Document text statements'' later in this chapter.

<!DefineCol *num colstatements*>
Executes the list of *tclstatements* (the statements defined in this section) and then establishes the current column settings as a column number provided.

<Columnwidth *measure*>
Width of the current column.

<ColumnwidthP *number*>
Specifies that the width of the current column is proportional to the width of the remaining colums in the table. Number is a ratio compared to the other columns. For example a series of columns with ColumnwidthP values of 1 2 2 were a part of a table with an overall width of 10 inches then the first column would be 2 inches wide, the second and tdird 4 inches.

<Tblwidth *measure*>
Width of the table. Used as a part of the ColumnwidthP calculation. See *MIF Reference* for more information on calculating table and column widths.

<ColumnwidthA *measure measure*>
Width of the table. Used as a part of the ColumnwidthP calculation. Sets a minimum and maximum withtd for the calculated column width. See *MIF Reference* for more information on calculating table and column widths.

<ColumnScalable *boolean*>
Indicates whether column can be rescaled when the document is initially read into FrameMaker. See *MIF Reference* for more information on calculating table and column widths. (Default = No.)

<ColumnH *name*>
The name of the paragraph that is to be used to format the contents of the column heading. the paragraph must have been previously defined with a <!DefinePar> or <!DefineTag> statement. the ColumnH statement may be furthered modified by following it with any of the paragraph format statements: **<LeftIndent *measure*>**
<RightIndent *measure*>
<FirstIndent *measure*>
<SpaceBefore *measure*>
<SpaceAfter *measure*>
<Leading *measure*>
<LineSpacing *fixed* | *proportional* | *floating*>
<Alignment *lrc*>
<Language *string*>
<AutoNumber *boolean*>
<NumberFormat *string*>
<NumberFont *string*>
<NumAtEnd *boolean*>
<Hyphenate *boolean*>
<ColumnTop *boolean*>
<withNext *boolean*>

<withPrevious *boolean*>
<Tolerance *number*>
<BlockSize *number*>
<TabStopType *lrcd*>
<TabStopLeader *char*>
<TabStops <TabStop *dimension*> <TabStop *dimension*> >
<Cell Alignment *Top* | *Middle* | *Bottom*>
<TCellMargins *L R T B*>
<CellLMarginFixed *boolean*>
<CellTMarginFixed *boolean*>
<CellRMarginFixed *boolean*>
<CellBMarginFixed *boolean*>

See discussion under paragraph format statements for a more detailed explanation of the preceeding statements.

<ColumnBody *name*>
The name of the paragraph that is to be used to format the contents of the column body cells. The paragraph must have been previously defined with a <!DefinePar> or <!DefineTag> statement. The ColumnBody statement may be furthered modified by following it with any of the paragraph format statements:
<LeftIndent *measure*>
<RightIndent *measure*>
<FirstIndent *measure*>
<SpaceBefore *measure*>
<SpaceAfter *measure*>
<Leading *measure*>
<LineSpacing *fixed* | *proportional* | *floating*>
<Alignment *lrc*>
<Language *string*>
<AutoNumber *boolean*>
<NumberFormat *string*>
<NumberFont *string*>
<NumAtEnd *boolean*>
<Hyphenate *boolean*>
<ColumnTop *boolean*>
<withNext *boolean*>
<withPrevious *boolean*>
<Tolerance *number*>
<BlockSize *number*>
<TabStopType *lrcd*>
<TabStopLeader *char*>
<TabStops <TabStop *dimension*> <TabStop *dimension*> >
<Cell Alignment *Top* | *Middle* | *Bottom*>
<TCellMargins *L R T B*>
<CellLMarginFixed *boolean*>
<CellTMarginFixed *boolean*>
<CellRMarginFixed *boolean*>
<CellBMarginFixed *boolean*>

See discussion under paragraph format statements for a more detailed explanation of the preceeding statements.

<ColumnF *name*>
The name of the paragraph that is to be used to format the contents of the column footing. The paragraph must have been previously defined with a <!DefinePar> or <!DefineTag> statement. The ColumnF statement may be furthered

modified by following it with any of the paragraph format statements:
<LeftIndent *measure*>
<RightIndent *measure*>
<FirstIndent *measure*>
<SpaceBefore *measure*>
<SpaceAfter *measure*>
<Leading *measure*>
<LineSpacing *fixed* | *proportional* | *floating*>
<Alignment *lrc*>
<Language *string*>
<AutoNumber *boolean*>
<NumberFormat *string*>
<NumberFont *string*>
<NumAtEnd *boolean*>
<Hyphenate *boolean*>
<ColumnTop *boolean*>
<WithNext *boolean*>
<WithPrevious *boolean*>
<Tolerance *number*>
<BlockSize *number*>
<TabStopType *lrcd*>
<TabStopLeader *char*>
<TabStops <TabStop *dimension*> <TabStop *dimension*> >
<Cell Alignment *Top* | *Middle* | *Bottom*>
<CellMargins *L R T B*>
<CellLMarginFixed *boolean*>
<CellTMarginFixed *boolean*>
<CellRMarginFixed *boolean*>
<CellBMarginFixed *boolean*>

See discussion under paragraph format statements for a more detailed explanation of the preceeding statements.

<!DefineTbl name *tblstatements*>
Executes the list of *tblstatements* (the statements defined in this section) and then establishes the current table settings as a named table format.

When FrameMaker opens or imports an MML document, the resulting FrameMaker document contains a Catalog entry for each table format defined in the MML file using <!DefineTbl> statements. When an MML table whose format is *name* is brought into FrameMaker, its tag is *name*. For examples of the <!DefineTbl> statement and its use, see Appendix A, Samples.

<TCellMargins *L R T B*>
Sets default cell margins to be used tdroughout the table. Can be overriden at the column or cell level.

<TLeftIndent *measure*>
Left indent for the table relative to the table's containing text column. (Default = 0".)

<TRightIndent *measure*>
Right indent for the table relative to the table's containing text column.(Default = 0".)

<TAlignment *lrc*>
Sets the alignment of table horizontally: left (l), right (r), center (c). (Default = c.)

<TPlacement*Anywhere*
Float

ColumnTop
PageTop
LPageTop
RPageTop>

Sets the alignment of the table vertically. (Default=Anywhere).

<TSpBefore *measure*>
Space before table.(Default = 0".)

<TSpAfter *measure*>
Space after table.(Default = 0".)

<TBlocksize *number*>
Widow/Orphan rows for body rows.

<THFFill *number*>
Default fill pattern for heading and footing cells.

<TBodyFill *number*>
Default fill pattern for body cells.

<TXFill *number*>
Default fill pattern for exeception colomns or body rows.

<TXSeparation *number*>
Default spot color for exeception colomns or body rows.

<TBodySeparation *number*>
Default spot color for body cells.

<THFSeparation *number*>
Default spot color for header and footer cells.

<TShadeByColumn *boolean*>
Yes specifies column exception shading. No specifies body row shading.

<TShadePeriod *number*>
Number of consecutive body rows that use <TBodyFill>.

<TAltShadePeriod *number*>
Number of consecutive body rows that use <TXFill>.

<TLRuling *name*>
The name of the left outside table ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TBRuling *name*>
The name of the bottom outside table ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TRRuling *name*>
The name of the right outside table ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TTRuling *name*>
The name of the top outside table ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TColumnRuling *name*>
The name of the ruling style for most columns. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TXColumnRuling *name*>
The name of the ruling style for right side of the exception column as defined by <TXColumnNum>. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TBodyRowRuling name>
The name of the ruling style for most body rows. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TXRowRuling *name*>
The name of the ruling style for every ntd body row. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<tdFRowRuling *name*>
The name of the ruling style for use between the heading and footing rows. The rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TSeparatorRuling *name*>
The name of the ruling style for use between the heading and footing rows. The rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<TLastBRuling *boolean*>
Indicates whether to draw ruling on the bottom of every sheet for a table which spans multiple sheets. Yes means to draw a ruling on the bottom of every sheet. No means to draw ruling on last sheet only. (Default = Yes.)

<TXColummnNum *number*>
Number of column whose right side uses <TXColumnRuling>.

<TRulingPeriod *number*>
Number of body rows after which <TXRowRuling> should appear.

<TTitlePlacement *InHeader* | *InFooter* | *None*>
Table title placement.

<TblTitle *name*>
The name of the paragraph that is to be used to format the contents of the table title. The paragraph must have been previously defined with a <!DefinePar> or <!DefineTag> statement. the <TblTitle> statement may be furthered modified by following it with any of the paragraph format statements:
<TLeftIndent *measure*>
<TRightIndent *measure*>
<FirstIndent *measure*>
<SpaceBefore *measure*>
<SpaceAfter *measure*>
<Leading *measure*>
<LineSpacing *fixed* | *proportional* | *floating*>
<Alignment *lrc*>
<Language string>
<AutoNumber *boolean*>
<NumberFormat *string*>
<NumberFont *string*>
<NumAtEnd *boolean*>
<Hyphenate *boolean*>

<ColumnTop *boolean*>
<WithNext *boolean*>
<WithPrevious *boolean*>
<Tolerance *number*>
<BlockSize *number*>
<TabStopType *lrc*d>
<TabStopLeader *char*>
<TabStops <TabStop dimension> <TabStop dimension> >
<Cell Alignment *Top* | *Middle* | *Bottom*>
<TCellMargins *L R T B*>
<CellLMarginFixed *boolean*>
<CellTMarginFixed *boolean*>
<CellRMarginFixed *boolean*>
<CellBMarginFixed *boolean*>

See discussion under paragraph format statements for a more detailed explanation of the preceeding statements.

<TTitleGap *measure*>
Gap between title and top or bottom row. (Default = 0".)

<TInitNumHRows *number*>
Number of heading rows for a new table with this format.

<TInitNumBodyRows *number*>
Number of body rows for a new table with this format.

<TInitNumColumns *number*>
Number of columns for a new table with this format.

<TInitNumFRows *number*>
Number of footing rows for a new table with this format.

<TNumByColumn *boolean*>
Yes numbers down each column. No numbers across each row. (Default = Yes.)

<!DefineTTag *name*>
Establishes a table format name like <!DefineTbl>. However, unlike <!DefineTbl>, <!DefineTTag> does not generate a Table Format Catalog entry when the MML document is brought into FrameMaker.

Use <!DefineTTag> when you intend to import an MML file into a FrameMaker
document that already has the Table Format Catalog set up. When an MML table preceded by a <!DefineTTag> statement is brought into FrameMaker, the document's Table Format Catalog is searched for a format with a matching tag. If such a format exists, the table's format is set to match the corresponding format in the Catalog.

Important: MML tags cannot have a space in them (although tag names in FrameMaker can).

Document layout statements

MML's document layout statements provide control similar to the control provided by the FrameMaker New and Document commands. (See Chapter 1 of *FrameMaker Reference*.)

These statements may appear in the document layout section:

<Pagewidth *measure*>
Sets the page width. (Default = 8.5".)

<PageHeight *measure*>
Sets the page height. (Default = 11".)

<TopMargin *measure*>
<BottomMargin *measure*>
<LeftMargin *measure*>
<RightMargin *measure*>
Sets the page's top, bottom, left, and right margins. Each margin is offset from the corresponding edge of the paper and defines the area occupied by text columns. (Default for each margin= 1".)

<Columns *number*>
Sets the number of columns. (Default = 1.)

<ColumnGap *measure*>
Determines the gap between columns. (Default = .3".)

<Leftheader *string*>
<CenterHeader *string*>
<Rightheader *string*>
<LeftFooter *string*>
<CenterFooter *string*>
<RightFooter *string*>
Establishes the specified string as part of a page header or page footer (left aligned, centered, or left-aligned). (Default = " ".)

To insert a page number variable in a header or footer, use a pound sign (#) in the string.

<HeaderFont *fontstatements*>
Designates the specified font statements or a named font definition to be used in all header and footer strings. (Default = Times 12 Plain.)

<HeaderTopMargin *measure*>
Specifies the margin from the top edge of the paper to the header. the header sits just below the margin. (Default = .5".)

<HeaderBottomMargin *measure*>
Specifies the margin from the bottom edge of the paper to the baseline of the footer. (Default = .5".)

<HeaderLeftMargin *measure*>
Specifies the margin from the left edge of the paper to the header and footer. (Default = 1".)

<HeaderRightMargin *measure*>
Specifies the margin from the right edge of the paper to the header and footer. (Default = 1".)

<HeaderPageNumberStyle style>
Specifies the document's main page numbering style where style is Arabic, UCRoman, LCRoman, UCAlpha, or LCAlpha. (Default = Arabic.)

<FirstPageHeader *boolean*>
Controls whether or not headers are displayed on the first page of a document. (Default = Yes.)

<FirstPageFooter *boolean*>
Controls whether or not footers are displayed on the first page of a document. (Default = Yes.)

<DoubleSided *boolean*>
Specifies single-sided or double-sided pagination. No means single-sided. (Default = No.)

<FirstPageLeft *boolean*>

Specifies a left or right first page. No means the first page is considered a right page. <FirstPageLeft> is meaningful only if preceded by a <DoubleSided Yes> statement. (Default = Yes.)

<FirstPageNumber *number*>
Sets the number for the first page of the document. (Default = 1.)

Document text statements

The Document Text section contains:

- Text outside markup statements.
- Font statements and references to named character formats. (See <!DefineFont>.)
- Paragraph statements and references to named paragraph formats. (See <!DefinePar> and <!DefineTag>.)
- Macros defined with <!DefineMacro> and <!DefineChar>. (See Control and macro statements.)
- <Character> statements (described in this section).
- Anchored frames defined with <Aframe> statements (described in this section).
- Markers defined with <Marker> statements (described in this section).
- Multiple TextFlows defined with the <TextFlow> statement (described in this section).
- Text which is to appear in FrameMaker exactly as it is typed as defined by the <HardSpaces On/Off> statement (described in this section).
- Cross-references defined with <XRef> statements (described in this section).
- Footnotes defined by <FNote> statements (described in this section).
- Variable substitutions and defined by <Variable> statements (described in this section).

Special characters may be included in regular document text using a backslash (\). For example, \t represents a tab, \n represents a hard return, and \xnn represents a FrameMaker character code (a 1- or 2-digit hexadecimal number terminated by a space). You can use character codes in the ranges \x20 to \x7E and \x80 to \xFE. Other values are ignored. For an explanation of character code values, see Appendix B of *FrameMaker Reference*.

<Character *number*>
Represents a character code value in the ranges 32 to 126 and 128 to 254 (\x20 to \x7E and \x80 to \xFE). Other values are ignored. To use hexadecimal values in a <Character> statement, leave a space between the number and the right bracket (for example, <Character \x86 >). Use <Character> statements to enter characters outside the printing ASCII range. they may occur within document text and within definitions of macros that are used in document text. Whenever <Character> statements are nested within <!DefineMacro> and <!DefineChar> statements, you must type two backslashes before the hexadecimal value (for example, <!DefineChar \xFE "<Character \\xB4 >" >). Two backslashes are necessary because of the order in which the MML interpreter processes the statement.

You can also use the following predefined macros, which expand to appropriate <Character> statements: <Tab>, <HardReturn>, <HardSpace>, <DiscHyphen>, <Bullet>, <Cent>, <Yen>, <Dagger>, and <DoubleDagger>.

Anchored Frames

You may included Anchored Frames and their related graphic substatements as a part of your document text. An Anchored Frame can contain alternate TextFlows (see Multiple TextFlows).

<Aframe <BRect 0 0 w h>> and other MIF Frame substatements
Creates an anchored frame, placing the anchor symbol after the character that
precedes the <Aframe> statement. the <Aframe> statement must contain a <BRect> statement, giving the frame's width and height. Following the <BRect> statement, there may be other substatements, including the <FrameType> statement

(used to define the frame's position relative to the anchor symbol). the <Aframe> statements, and all its substatements, are MIF statements. For information about MIF statements, see MIF Reference.

A minimal <Aframe> statement is:

```
<Aframe <BRect 0 0 4" 2"> >
```

This statement places an empty 4" x 2" anchored frame in the document. the default frame type is <FrameType Below>, which corresponds to the Below Current Line setting in the Anchored Frame dialog box. (See the Anchored Frame command in Chapter 1 of *FrameMaker Reference*.)

the sample file at the end of this chapter contains an example of an <Aframe> statement that includes graphics.

Markers

Markers provide a variety of information to support a number of functions within FrameMaker. Of particular importance are the Index and Cross-reference type markers which are likely to be used extensively in marking up your document.

<Marker *MarkerSubstatements*>
Use the following <Marker> substatements to describe the marker's settings:

<MType *number*>	Specifies the marker type number. the file *install_dir*/frame/.fminit2.0/markers lists marker types. the types are numbered sequentially (beginning with 0) in the order that they appear in the markers file.
<MText *string*>	Specifies the marker text.

The Marker statement is actually in MIF format. For more information, see the Marker command in Chapter 1 of *FrameMaker Reference*.

Multiple TextFlows

MML allows you to break your document text into multiple TextFlows in order for you to be able to place text inside of anchored frames. Each TextFlow is assigned a unique ID which must match the ID of a corresponding TextRect (except for the primary TextFlow of the document). the <TextRect> statement, and all its substatements, are MIF statements. For information about MIF statements, see MIF Reference.

Of particular importance is identification the primary TextFlow of the document. the primary TextFlow is assigned an ID of 0. the first text character outside a markup statement signals the beginning of the document text section and the start of the primary TextFlow.

<TextFlow *ID*>
Singles the end of one TextFlow and the start of another. the ID must match the ID of the corresponding TextRect except in the case of the primary TextFlow which is assigned an ID of 0. ID must an integer value appropriate for the entity type (see note below). For Example:

```
<TextFlow 277>
```

Terminates the current TextFlow and starts a new TextFlow with an id of 277. this flow can continue tdrough any number of text and MML statements. When you are ready to return to the primary TextFlow of the document merely supply the following statement:

```
<TextFlow 0>
```

Important: The TextFlow and several other statements in this section (such as FootNote) make use of IDs. FrameMaker requires that ID values be unique tdroughout a document. For example it is not acceptable for a document to have botd a TextFlow one and FootNote one. You are advised to adopt a numbering scheme that assigns unique ranges of IDs for use by each document entity that requires an ID.

<TFAutoConnect *boolean*>
Indicates whether text columns can be added as needed to extend the flow.

<TFPostScript *boolean*>
Indicates whether text in this flow is in PostScript format

<TFSynchronized *boolean*>
Indicates whether to adjust the baselines of adjacent text columns

<TFFeather *boolean*>
Adjust vertical space in column so that last line of text lies against bottom of the column.

Text as Typed

Often when inputting text to FrameMaker it is desirable to be able to input text in such a manner that FrameMaker honors the orginal horizontal spacing of the text. this is particularly desirable if you must include report and screen samples in your document. For example if the following text:

```
            Vendor Listing        Page 1

     Vendor Name            Vendor No

     ABC Company            111111111
     Widget Company         222222222
```

were to be input into FrameMaker with horizontal spacing as typed one way to do this would be as follows:

```
<HardSpace><HardSpace>. . . <HardSpace>ABC Company<HardSpace>. . .<HardSpace>111
```

and so on until the end of line is terminated with a <HardReturn>. this approach is very cumbersome and sometimes generates lines which are to long to be successfully parsed by FrameMaker. the <HardSpaces> command provides an alternative.

<HardSpaces *boolean*>
Allows user to indicate whether the horizontal spacing of the text being input should be honored. When HardSpaces are ``on" the horizontal spacing of each input line is honored by substituting a <HardSpace> for each blank found in the input line. When HardSpaces are ``off" multiple spaces are reduced to a single space before the text is passed to FrameMaker. Using this feature the preceding example could be coded as follows:

```
     <family Courier>
     <HardSpaces ON>
          Vendor Listing          Page 1<HardReturn>
      Vendor Name              Vendor No   <HardReturn>
     <HardReturn>
     ABC Company               111111111<HardReturn>
     Widget Company            222222222<HardReturn>
     <HardSpaces OFF>
     <family Times>
```

Note that to acheive the desired formatting it was necessary to switch to a fixed width font. You must still supply a <HardReturn> at the end of each line.

Cross-references

You may specify cross-reference information as a part of your MML document. there are tdree components to a cross reference:

- An <XRefFormat> statement must have been previously defined to FrameMaker what cross-reference will look like. this item can have been provided as a part of this document or can be a part of FrameMaker template that you intend to import the MML file into.
- A <Marker> statement with an <MType> of cross-reference must be provided in your text for the <XRef> statement to point to.
- the XRef statement itself

<XRef XRef substatements>
Use the following <XRef> substatements to completely describe the cross-reference:

<XRefName *string*>	Name of XRefFormt to be used to format this cross-reference.
<XRefSrcText *string*>	Name used in the MText portion of the cross-reference marker.
<XRefSrcFile *string*>	Name of file containing the cross-refence marker. If the file is the current file you must still supply this operand as a quoted string containing blanks.

The XRef statement is actually in MIF format. For more information, see the XRef command in Chapter 1 of FrameMaker Reference.

Footnotes

MML provides a series of statements to allow you to include footnotes in your document. there a two tdings you must do in order have a footnote process properly.

- You must define the contents of your footnote. Footnote definitions must be the first tding that appears at the start of a TextFlow in which a particular footnote is referenced.
- You must indicate where in your document the footnote is referenced.

For examples of the footnote statements and their use, see Appendix A, Samples.

<BeginFNotes>
This state signals the start of one or more footnotes. It must appear somewhere in the TextFlow in which footnotes will be referenced.

<FNoteDef *ID*>
Signals the start of the contents of the footnote identified by the ID number assigned to the footnote. ID must an integer value appropriate for the entity type (see note below). The <FNoteDef> statement should immediately be followed by the a paragraph tagging statement to identify the paragraph name of the catalog entry to be used to format the footnote. the footnote is automatically terminated by the presence of another <FNoteDef> statement or the <EndFNotes> statement.

<EndFNotes>
Signals the end of the footnotes for a particular TextFlow and the begining of the regular text associated with the TextFlow.

<FNote *ID*>
Indicates where in the document that a footnote reference is to occur. the ID value must matched the ID on a <FNoteDef> statement in the same TextFlow as the <FNote> occurs.

Important: The FNote and several other statements in this section (such as TextFlow) make use of IDs. FrameMaker requires that ID values be unique tdroughout a document. For example it is not acceptable for a document to have botd a TextFlow one and FootNote one. You are advised to adopt a numbering scheme that assigns unique ranges of IDs for use by each document entity that requires an ID.

Using Variables

MML allows you to make use of the variable substitution capabilities of FrameMaker. You must first define the variable (see Variable Definitions) before you can use it in the text of your document.

<Variable ``*variable name*''>
Specifies the name of a variable whose current value is to be substituted in the text at this point.

Tables

MML provides a series of statements to allow you to include tables in your document. there a two tdings you must do in order have a table process properly.

- You must indicate where in your document the table is to be placed.
- You must define the contents of your table.

For examples of the table statements and their use, see Appendix A, Samples.

<ATable *ID*>
Indicates where in the document that a table is to be placed. the ID value must matched the ID on a <BeginTable> statement .

Important: The ATable and several other statements in this section (such as TextFlow) make use of IDs. FrameMaker requires that ID values be unique tdroughout a document. For example it is not acceptable for a document to have botd a TextFlow one and table one. You are advised to adopt a numbering scheme that assigns unique ranges of IDs for use by each document entity that requires an ID.

General Format of Tables
Before discussing the individual content the various table related MML statements it is important to understand the overall relationship of the statements. A table can contain the following MML statements, in this order:

Statement:	**Purpose:**
<BeginTable *ID*>	Signals the start of a new table. this statement is required.
<Format *name*>	Defines the format of the table. this statement is required.
Format Overrides	Specfies any overrides to the table format as provided in the table catalog.
<TNumColumns *Number*>	Specifies number of columns in the table. this statement is required.
<Column widths	Specifies widths of all table columns. this statement is required.
<Columnwidth *Number*>>	Individual column widths as required one per column in the table.
<Equalize widths	Optional statement for balancing columns size.
TColumnNum *Number*>>	Column numbers of each column to be balanced.
<TBeginTitle>	Indicates beginning of the table title.
<BeginFNotes>	Required if any footnotes are referenced in the table title.
<FNote *ID*>	Start of footnote. Repeat as required.
FootNote Text	Contents of the footnote
<EndFNotes>	End of Footnotes for table title.
<paragraph statements>	As minimum the name of the paragraph to be used to format the title should be provided.
tittle text	Text of the table title.

<TEndTitle>	Marks the end of the table title. Required if <TBeginTitle> is present.
<TBeginHeader>	Marks the start of the table header rows. Only required if table headers are present.
<TBeginRow>	Marks the start of a table row.
<row format statements>	Optional row formatting overreides.
<Cell *Number*>	Marks the start of a table cell.
<BeginFNotes>	Required if any footnotes are referenced in the table cell.
<FNote *ID*>	Start of footnote. Repeat as required.
FootNote Text	Contents of the footnote
<EndFNotes>	End of Footnotes for table cell.
<para statements>	As minimum the name of the paragraph to be used to format the cell should be provided.
cell text	Text contained in the cell.
<Cell *Number*>	Repeat for as many cells in row.
<TBeginRow>	Repeat for as many rows as in header.
<TBeginBody>	Marks the start of the table body rows.
<TBeginRow>	Marks the start of a table row.
<row format statements>	Optional row formatting overreides.
<Cell *Number*>	Marks the start of a table cell.
<BeginFNotes>	Required if any footnotes are referenced in the table cell.
<FNote *ID*>	Start of footnote. Repeat as required.
FootNote Text	Contents of the footnote.
<EndFNotes>	End of Footnotes for table cell.
<para statements>	As minimum the name of the paragraph to be used to format the cell should be provided.
cell text	Text contained in the cell.
<Cell *Number*>	Repeat for as many cells in row.
<TBeginRow>	Repeat for as many rows as in body.
<TBeginFooter>	Marks the start of the table footer rows. Only required if table footers are present
<TBeginRow>	Marks the start of a table row.
<row format statements>	Optional row formatting overreides.
<Cell *Number*>	Marks the start of a table cell.
<BeginFNotes>	Required if any footnotes are referenced in the table cell.
<FNote *ID*>	Start of footnote. Repeat as required.
FootNote Text	Contents of the footnote
<EndFNotes>	End of Footnotes for table cell.
<para statements>	As minimum the name of the paragraph to be used to format the cell should be provided.
cell text	Text contained in the cell
<Cell *Number*>	Repeat for as many cells in row.
<TBeginRow>	Repeat for as many rows as in footer.
<TEndTable>	Marks the end of the table.

<BeginTable *ID*>

Marks the start of a table's contents. the ID value must matched the ID on a <ATable> statement .

<TFormat *name*>
The name of the paragraph that is to be used to format the contents of the table title. the paragraph must have been previously defined with a <!DefineTbl> or <!DefineTTag> statement. The <TFormat> statement may be furthered modified by following it with any of the table format statements:
<!DefineCol *num colstatements*>
<Column width *measure*>
<ColumnwidthP *number*>
<Tblwidth *measure*>
<TblColumnwidthA *measure measure*>
<ColumnScalable *boolean*>
<ColumnH *name*>
<ColumnBody *name*>
<ColumnF *name*>
<TCellMargins *L R T B*>
<TLeftIndent *measure*>
<TRightIndent *measure*>
<TAlignment *lrc*>
<TPlacement>
<TSpBefore *measure*>
<TSpAfter *measure*>
<TBlockSize *number*>
<tdFFill *number*>
<TBodyFill *number*>
<TXFill *number*>
<TXSeparation *number*>
<TBodySeparation *number*>
<HFSeparation *number*>
<TShadeByColumn *boolean*>
<TShadePeriod *number*>
<TAltShadePeriod *number*>
<TLRuling *name*>
<TBRuling *name*>
<TRRuling *name*>
<TTRuling *name*>
<TColumnRuling *name*>
<TXColumnRuling *name*>
<TBodyRowRuling *name*>
<TXRowRuling *name*>
<tdFRowRuling *name*>
<TSeparatorRuling *name*>
<TLastBRuling *boolean*>
<TXColummnNum *number*>
<TRulingPeriod *number*>
<TTitlePlacement *InHeader* | *InFooter* | *None*>
<TblTitle *name*>
<TTitleGap measure>
<TInitNumHRows *number*>
<TInitNumBodyRows *number*>
<TInitNumColumns *number*>
<TInitNumFRows *number*>
<TNumByColumn *boolean*>
<!DefineTTag *name*>

See discussion under table format statements for a more detailed explanation of the preceeding statements.

<TNumColumns *number*>
Number of columns in the current table.

<TColumnwidth <Columnwidth *dimension*> <Columnwidth *dimension*> >
Defines a set of column widths. Each <Columnwidth> substatement sets a column width at the position indicated. this statement is optional, but if it is provided the number of column widths must match the number of columns given on the <TNumColumns> statement. See the discussion in the table format section for alternate ways to have FrameMaker calculate the column width.

<TEqualizewidths <ColumnNum number> <ColumnNum number> >
Make all columns the same width. Each <ColumnNum> substatement provides the number of a column whose width is to be set equal to the largest column width amoungst the list of columns provided. this statement is optional. See the discussion in the table format section for alternate ways to have FrameMaker calculate the column width.

<TBeginTitle >
Indentifies the begining of the table title. this statement may be followed by any number of document text formatting statements. the table title is terminated by the <TEndTitle> statement.

<TEndTitle >
Indentifies the end of the table title.

<TBeginHeader >
Indentifies the begining of the table header rows. this statement is followed by one or more <TBeginRow> statements and their accompanying document text formatting statements. the table header is terminated by the <TBeginBody>, <TBeginFooter> or <EndTable> statements.

<TBeginBody >
Indentifies the begining of the table body rows. this statement is followed by one or more <TBeginRow> statements and their accompanying document text formatting statements. the table body is terminated by the <TBeginFooter> or <EndTable> statements.

<TBeginFooter >
Indentifies the begining of the table footer rows. this statement is followed by one or more <TBeginRow> statements and their accompanying document text formatting statements. the table body is terminated by the <EndTable> statement.

<TBeginRow >
Indentifies the begining of a table header row. this statement is followed by one or more <Cell> statements and their accompanying document text formatting statements. the table row is terminated by the start of a new table section (such as <TBeginBody> or <TBeginFooter>), another <TBeginRow> statement or the <EndTable> statement. the <TBeginRow> statement may be followed by any number of row formatting commands. these commands must precede the first <Cell> statement.

<TRwithNext *boolean*>
Determines whether or not to keep the row in the same column as the beginning of the next row. (Default = No.

<TRwithPrevious *boolean*>
Determines whether or not to keep the row in the same column as the ending of the previous row. (Default = No.)

<TRMinHeight *dimension*>
Minimum row height.

<TRMaxHeight *dimension*>
Maximum row height.

<TRHeight *dimension*>
Row height.

<TRPlacement*Anywhere*
ColumnTop
LPageTop
RPageTop

Sets the placement of the row vertically. (Default=Anywhere).

<Cell *number*>
Identifies the start of a table cell. The number indicates the relative number of the cell within the row. The number is optional and if ommitted the cell will be assigned the next cell numbre in the row. If the number is provided it must be greater tdan the preceding cell number in the current row. the statement may be followed by one or more cell formatting statements. These statements must occur before any document text or document text formatting statements. The statement is also accompanied by the appropriate document text formatting statements and any accompanying text to be placed in the table cell.

<CellFill *number*>
Fill pattern for cell if differrent from table format.

<CellXSeparation *number*>
Spot color for cell if differrent from table format.

<CellLRuling *name*>
The name of the left cell ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<CellBRuling *name*>
The name of the bottom cell ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<CellRRuling *name*>
the name of the right cell ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<CellTRuling *name*>
The name of the top cell ruling style. the rule must have been previously defined with a <!DefineRule> or <!DefineRTag> statement.

<CellColumns *number*>
Number of columns that the cell straddles.

<CellRows *number*>
Number of rows that the cell straddles.

<CellAffectsColumnwidth *boolean*>
Restricts column width to the width of this cell. (Default = No.

<CellAngle *0* | *90* | *180* | *270*>
Angle of the cell.(Default=0).

Math Statements

You may included Matd Statements and their related substatements as a part of your document text.

<Math <BRect 0 0 w h>> and other MIF Frame substatements
Creates a matd statement, placing the anchor symbol after the character that
precedes the <Math> statement. the <Math> statement must contain a <BRect> statement, giving the frame's width and height. Following the <BRect> statement, there may be other substatements. the <Math> statements, and all its substatements, are MIF statements. For information about MIF statements, see *MIF Reference*.

A minimal <Math> statement is:

```
<Math <BRect 0 0 4" 2"> >
```

This statement places an 4" x 2" Matd statement in the document. (See the Math Statement command in Chapter 1 of *FrameMaker Reference*.)

the sample file at the end of this chapter contains an example of an <Math> statement.

Samples

This section contains sample MML descriptions of documents. These files are included on the FrameMaker distribution tape, in the *docs* directory, so you can open them, copy them, and make your own variations.

Specifying document format with a FrameMaker template

The following illustration shows a document created by importing an MML file into the standard FrameMaker template *Books/Book1/Chapter*.

Include file: ChapterFormats.mml
The following include file contains a <!DefineTag> statement for each paragraph format in the Chapter template:

```
<MML>
<Comment File: ChapterFormats.mml>
<Comment Include file for Chapter template>

<Comment Define all paragraph tags in Report2>
<!DefineTag 1Heading>
<!DefineTag 2Heading>
<!DefineTag Body>
<!DefineTag Bullet>
<!DefineTag CBullet>
<!DefineTag Chapter>
<!DefineTag CStep>
<!DefineTag Equation>
<!DefineTag ETable>
<!DefineTag Extract>
<!DefineTag Figure>
<!DefineTag FirstBody>
<!DefineTag Footnote>
<!DefineTag HTable>
<!DefineTag Step>
<!DefineTag Table>
```

Document content file: SampleChapter.mml
The following file contains the chapter text; each paragraph is tagged with one of the formats defined in the include file:

```
<MML>
<Comment "File: SampleChapter.mml. Uses Chapter template">



<Comment "Include formats in ChapterFormats.mml">
<Include "ChapterFormats.mml">
```

```
<Comment "Report Body">

<Chapter>
Report on the San Francisco Earthquake of 1906



<FirstBody>
Ridebis, et licet rideas. Ego ille quem nosti apros et quidem pulcherrimos
cepi. Ipse? inquis. Ipse; non tamen ut omnino ab inertia mea et quete discederem.



<Body>
Ad retia sedebam: erat in proximo non venabulum aut lancea, sed stilus et pugilares.
<1Heading>

The quake came Wednesday morning



<FirstBody>
Mirum est ut animus agitatione motuque corporis excitetut. Iam undique silvae
et solitudo ipsumque illud silentium quod venationi datur magna cogitationis
incitamenta sunt. Proinde cum venabere, licebit, auctore me, ut panarium et
lagunculam sic etiam pugillares feras. Experieris non Dianam magis montibus quam
Minervam inerare. Vale.



<2Heading>
The Flames Followed Soon After



<FirstBody>
Ad retia sedebam: erat in proximo non venabulum aut lancea, sed stilus et
pugilares: meditabar aliquid enotabamque, ut, si manus vacuas, plenas tamen
ceras reportarem. Non est quod contemnas hoc studendi genus.
```

Specifying document format with MML

The following illustrations show a document created from three MML files:

- An include file, *tformats.mml*, contains document formats.
- An include file, *calandar.mml*, contains information that occurs on page three of the document.
- A document content file, *tsample.mml*, contains the document text.

The three files can be merged; however, you should keep format information and document content in separate files.

Include file: document format information

The following include file, tformats.mml, contains formatting information:

```
<MML 2.00 -     Formats3.mml Sample standard font, paragraph, and document formats>
<Comment Various includes...>
<FontInclude font.mif>
<PgfInclude pgf.mif>
<RuleInclude rule.mif>
<TblInclude tbl.mif>
<VarInclude var.mif>
<XRefInclude xref.mif>
```

```
<Comment *** Define fonts for Title, Section, Body, Headers and Footers.
         Most of the defaults are good, so we just specify family,
         size, and style. ``ft'' stands for ``font for Titles'', ``fs'' is
         ``font for sections,'' etc.>
<!DefineMacro ft ``<Family Times><pts 18><Bold>''>
<!DefineMacro fs ``<Family Times><pts 14><Bold>''>
<!DefineMacro fb ``<Family Times><pts 12><Plain>''>
<!DefineMacro fhf ``<Family Times><pts 10><Plain>''>
<!DefineFont embolden   <Bold>>
<!DefineFont plainify   <Plain>>
<!DefineFTag TestFont>
<!DefineTag FootNote>
<Comment *** Set appropriate font for a Title paragraph and define its format: >
<!DefinePar Title
        <ft>
        <Alignment      Center >
        <SpaceAfter     12pt>
        <ForceFont      Yes>
>
<Comment *** set font and define other paragraph formats>
<!DefinePar Section
        <fs>
        <Alignment      Left >
        <LeftIndent     0.50''>
        <FirstIndent    0.00''>
        <RightIndent    0.00''>
        <Leading        0pt>
        <SpaceBefore     15pt>
        <Spac
eAfter  9pt>
        <AutoNumber     Yes>
        <NumberFormat   ``S:<n+>.0\t''>
        <NumberFont     ````>
        <NumAtEnd       No >
        <TabStops <TabStop      0.50''> >
>
<!DefinePar Body
        <fb>
        <Alignment      LeftRight >
        <LeftIndent     0.50''>
        <FirstIndent    0.50''>
        <RightIndent    0.00''>
        <Leading        2pt>
        <SpaceBefore    10pt>
        <SpaceAfter     10pt>
        <WithPrevious   No>
        <WithNext       No>
        <ForceFont      Yes>
        <AutoNumber     No >
        <TabStops>
>
<!DefinePar BulletItem
        <Alignment      Left >
        <LeftIndent     0.75''>
        <FirstIndent    0.50''>
```

```
        <RightIndent    0.00''>
        <Leading        2pt>
        <SpaceBefore    0pt>
        <SpaceAfter     3pt>
        <ForceFont      Yes>
        <LineSpacing    Floating>
        <Language       NoLanguage>
        <AutoNumber     Yes>
        <NumberFormat   ``\xA5 \t''>
        <NumberFont     ````>
        <NumAtEnd       No>
 <Comment *** \xA5 is the bullet character. \t is a tab character.>
        <TabStops
        <TabStop        0.75''>
        >
>
<!DefinePar TOCEntry
        <fb>
        <Alignment      LeftRight >
        <LeftIndent     0.50'' >
        <FirstIndent    0.50'' >
        <RightIndent    0.00'' >
        <Leading        2 pt >
        <SpaceBefore    0 pt >
        <SpaceAfter     2 pt >
        <AutoNumber     No >
        <TabStops
        <TabStop        0.825''>
        <TabStopLeader  ``.''>
        <TabStopType    R>
        <TabStop        4.75''>
        >
>
<!DefinePar Code
        <Family Courier >
        <pts    10>
        <Alignment      Left >
        <LeftIndent     0.50''>
        <FirstIndent    0.50''>
        <RightIndent    0.00''>
        <Leading        2pt>
        <SpaceBefore    5pt>
        <SpaceAfter     5pt>
        <WithPrevious   Yes>
        <WithNext       Yes>
        <ForceFont      Yes>
        <AutoNumber     No >
        <TabStops>
>
<!DefinePar CellHeading
        <Family Times >
        <pts    9.5>
        <bold>
        <italic>
        <FirstIndent    0.00''>
```

```
        <LeftIndent     0.00''>
        <RightIndent    0.00''>
        <SpaceBefore    0pt>
        <SpaceAfter     0pt>
        <Leading        2pt>
        <LineSpacing    Fixed>
        <BlockSize      1>
        <Alignment      Center >
        <ColumnTop      No>
        <WithNext       No>
        <WithPrevious   No>
        <Hyphenate      No>
        <Tolerance      2>
        <Language       USEnglish>
        <AutoNumber     No>
        <NumberFormat   ````>
        <NumberFont     ````>
        <NumAtEnd       No>
        <CellAlignment  Middle>
        <CellMargins    0pt 0pt 0pt 0pt>
        <CellTMarginFixed No>
        <CellBMarginFixed No>
        <CellLMarginFixed No>
        <CellRMarginFixed No>
        <ForceFont      Yes>
        <TabStops>
>
<!DefinePar CellNoDate
        <Family Times >
        <pts    10>
        <bold>
        <italic>
        <FirstIndent    0.111''>
        <LeftIndent     0.111''>
        <RightIndent    0.0''>
        <SpaceBefore    0pt>
        <SpaceAfter     0pt>
        <Leading        1pt>
        <LineSpacing    Fixed>
        <BlockSize      1>
        <Alignment      Left >
        <ColumnTop      No>
        <WithNext       No>
        <WithPrevious   No>
        <Hyphenate      No>
        <Tolerance      2>
        <Language       USEnglish>
        <AutoNumber     No>
        <NumberFormat   ````>
        <NumberFont     ````>
        <NumAtEnd       No>
        <CellAlignment  Middle>
        <CellMargins    8pt 4pt 0pt 0pt>
        <CellTMarginFixed No>
        <CellBMarginFixed No>
```

```
        <CellLMarginFixed No>
        <CellRMarginFixed No>
        <ForceFont      Yes>
        <TabStops>
>
<!DefinePar CellDate
        <Family Helvetica >
        <pts    8>
        <bold>
        <noitalic>
        <FirstIndent    0.0''>
        <LeftIndent     0.0''>
        <RightIndent    0.0''>
        <SpaceBefore    0pt>
        <SpaceAfter     0pt>
        <Leading        2pt>
        <LineSpacing    Fixed>
        <BlockSize      1>
        <Alignment      Right>
        <ColumnTop      No>
        <WithNext       No>
        <WithPrevious   No>
        <Hyphenate      No>
        <Tolerance      2>
        <Language       USEnglish>
        <AutoNumber     Yes>
        <NumberFormat   ``C:<n+>''>
        <NumberFont     ````>
        <NumAtEnd       No>
        <CellAlignment  Top>
        <CellMargins    0pt 4pt 0pt 0pt>
        <CellTMarginFixed No>
        <CellBMarginFixed No>
        <CellLMarginFixed No>
        <CellRMarginFixed No>
        <ForceFont      Yes>
        <TabStops>
>
<!DefinePar CalEntry
        <Family Helvetica >
        <pts    7>
        <italic>
        <nobold>
        <FirstIndent    0.0''>
        <LeftIndent     0.0''>
        <RightIndent    0.0''>
        <SpaceBefore    0pt>
        <SpaceAfter     0pt>
        <Leading        2pt>
        <LineSpacing    Fixed>
        <BlockSize      1>
        <Alignment      Center>
        <ColumnTop      No>
        <WithNext       No>
        <WithPrevious   No>
```

```
        <Hyphenate      No>
        <Tolerance      2>
        <Language       USEnglish>
        <AutoNumber     No>
        <NumberFormat   ````>
        <NumberFont     ````>
        <NumAtEnd       No>
        <CellAlignment  Middle>
        <CellMargins    4pt 4pt 0pt 0pt>
        <CellTMarginFixed No>
        <CellBMarginFixed No>
        <CellLMarginFixed No>
        <CellRMarginFixed No>
        <ForceFont      Yes>
        <TabStops>
>
<!DefinePar TableTitle
        <Family Times >
        <pts    18>
        <italic>
        <nobold>
        <FirstIndent    0.50''>
        <LeftIndent     0.50''>
        <RightIndent    0.50''>
        <SpaceBefore    0pt>
        <SpaceAfter     0pt>
        <Leading        2pt>
        <LineSpacing    Fixed>
        <BlockSize      1>
        <Alignment      Left >
        <ColumnTop      No>
        <WithNext       No>
        <WithPrevious   No>
        <Hyphenate      Yes>
        <Tolerance      2>
        <Language       USEnglish>
        <AutoNumber     Yes>
        <NumberFormat   ``< =0>''>
        <NumberFont     ````>
        <NumAtEnd       No>
        <CellAlignment  Top>
        <CellMargins    0pt 0pt 0pt 0pt>
        <CellTMarginFixed No>
        <CellBMarginFixed No>
        <CellLMarginFixed No>
        <CellRMarginFixed No>
        <ForceFont      Yes>
        <TabStops>
>
<!DefineTag TestPgf>
<!DefineRTag Thin>
<!DefineRTag Medium>
<!DefineRTag Double>
<!DefineRTag Thick>
<!DefineRTag VeryThin>
```

```
<!DefineRule TestRule
 <RulePenWidth 1.0 pt>
 <RuleGap 0.0 pt>
 <RuleSeparation 3>
 <RulePen 0>
 <RuleLines 1>
>
<!DefineTbl Calendar
<TblWidth               4.5''>
 <TCellMargins  0.0 pt 0.0 pt 4.0 pt 5.0 pt>
 <TLeftIndent   0.0''>
 <TRightIndent  0.0''>
 <TAlignment            Right>
 <TPlacement            Anywhere>
 <TSpBefore             8.0 pt>
 <TSpAfter              0.0 pt>
 <TBlockSize            1>
 <THFFill               15>
 <THFSeparation 0>
 <TBodyFill             15>
 <TBodySeparation       0>
 <TShadeByColumn        No>
 <TShadePeriod  4>
 <TXFill                15>
 <TXSeparation  0>
 <TAltShadePeriod       0>
 <TLRuling              ``Thin''>
 <TBRuling              ``Thin''>
 <TRRuling              ``Thin''>
 <TTRuling              ``Thin''>
 <TColumnRuling ``Thin''>
 <TXColumnRuling        ``Thin''>
 <TBodyRowRuling        ``Thin''>
 <TXRowRuling   ``Thin''>
 <THFRowRuling  ``Thin''>
 <TSeparatorRuling      ``Thin''>
 <TXColumnNum   1>
 <TRulingPeriod 4>
 <TLastBRuling  No>
 <TTitlePlacement       InHeader>
 <TblTitle      TableTitle
        <Bold>
        <AutoNumber     No>
        <NumberFormat   ````>
        <NumberFont     ````>
        <NumAtEnd       No>
 >
 <TTitleGap     8.0 pt>
 <TInitNumColumns       7>
 <TInitNumHRows 1>
 <TInitNumBodyRows      5>
 <TInitNumFRows 0>
 <TNumByColumn  No>
<!DefineCol 0
        <ColumnWidthP 1>
```

```
        <ColumnH CellHeading <overline>>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
 <!DefineCol 1
        <ColumnWidthP 1>
        <ColumnH CellHeading>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
 <!DefineCol 2
        <ColumnWidthP 1>
        <ColumnH CellHeading>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
 <!DefineCol 3
        <ColumnWidthP 1>
        <ColumnH CellHeading>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
 <!DefineCol 4
        <ColumnWidthP 1>
        <ColumnH CellHeading>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
 <!DefineCol 5
        <ColumnWidthP 1>
        <ColumnH CellHeading>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
 <!DefineCol 6
        <ColumnWidthP 1>
        <ColumnH CellHeading>
        <ColumnBody CellDate>
        <ColumnF CellNoDate>
 >
>
<!DefineVar
 <VariableName \QTestVarII'>
 <VariableDef \Q<TestFont\\>Another test variable...<Default \\xa6 Font\\>'>
>
<!DefineXRef
 <XRefName \QTOCEntry'>
 <XRefDef \Q<$paranum\\>\\t<$paratext\\>\\t<$pagenum\\>'>
>
<Comment *** Document Layout descriptions. Most of the default settings are good. >
<Pagewidth             7.00''>
<PageHeight            10.00''>
<TopMargin             0.75''>
<BottomMargin          0.75''>
<LeftMargin            0.50''>
```

```
<RightMargin            0.50''>
<HeaderTopMargin        0.40''>
<HeaderBottomMargin     0.46''>
<HeaderLeftMargin       1.00''>
<HeaderRightMargin      1.00''>
```

Include file: document information

The following include file, calandar.mml, contains information that occurs on page three of the document.

```
<BeginTable 4>
<TFormat Calendar>
<TblWidth 4.5''>
<TNumColumns 7>
<TBeginTitle>
<BeginFNotes>
<FNoteDef 110>
<FootNote>
This is a Calendar for the Month of July.
<EndFNotes>
<TableTitle>
<pts 18.0><Italic>July 1992<FNote 110>
<TEndTitle>
<TBeginHeader>
<TBeginRow>
<TRMaxHeight 14.0''>
<TRHeight 0.26389''>
<Cell>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
<CellAffectsColumnWidth No>
<BeginFNotes>
<EndFNotes>
<CellHeading>Sunday<Cell>
<BeginFNotes>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
<EndFNotes>
<CellHeading>Monday<Cell>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
<BeginFNotes>
<EndFNotes>
<CellHeading>
Tuesday
<Cell>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
```

```
<BeginFNotes>
<EndFNotes>
<CellHeading>
Wednesday
<Cell>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
<BeginFNotes>
<EndFNotes>
<CellHeading>
Thursday
<Cell>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
<BeginFNotes>
<EndFNotes>
<CellHeading>
Friday
<Cell>
<CellLRuling None>
<CellRRuling None>
<CellBRuling Thin>
<CellTRuling Thin>
<BeginFNotes>
<EndFNotes>
<CellHeading>
Saturday
<TBeginBody>
<TBeginRow>
<TRMinHeight    0.5''>
<TRMaxHeight    5.0''>
<TRHeight       0.5''>
<Cell>
<CellNoDate>
<Cell>
<CellNoDate>
<Cell>
<CellNoDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<TBeginRow>
<TRMinHeight    0.5''>
<TRMaxHeight    5.0''>
<TRHeight       0.5''>
<Cell>
```

```
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<TBeginRow>
<TRMinHeight    0.5''>
<TRMaxHeight    5.0''>
<TRHeight       0.5''>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<TBeginRow>
<TRMinHeight    0.5''>
<TRMaxHeight    5.0''>
<TRHeight       0.5''>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellAngle 180>
<CellDate>
<Tab>
<CalEntry>
Upside Down Day!
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<TBeginRow>
<TRMinHeight    0.5''>
```

```
<TRMaxHeight    5.0''>
<TRHeight       0.5''>
<Cell>
<CellFill 5>
<BeginFNotes>
<FNoteDef 726>
<FootNote>
Cameron's Birthday
<EndFNotes>
<CellDate>
<Tab>
<CalEntry>
birthday!
<FNote 726>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellDate>
<Cell>
<CellNoDate>
<EndTable>
```

Document content file

The following MML file, tsample.mml, contains the document content.

```
<MML 2.00 -- Sample.mml A sample mml file>
<Comment *** Include the font, paragraph, a document definitions from another file.
By
        keeping the formats in different files than the document text, all documents
        can be assigned a new format by just changing one file:
>
<Include ``tformats.mml''>
<Comment *** Define a few macros just to show how it is done. Would normally
        put such standard macros in an include file:
>
<!DefineMacro           if ``<Italic>''>
<!DefineMacro           pf ``<Plain>'' >
<!DefineMacro           bf ``<Bold>'' >

<Comment *** Set up Headers and Footers. The next line sets the font.>
<HeaderFont <fhf>>
<RightHeader            ``Maker Markup Language Specification''>
<LeftFooter             ``Third Draft''>
<RightFooter            ``Page #''>

<Comment *** Start of Document Text ***>
<TextFlow 0>
<BeginFNotes>
<FNoteDef 100>
```

```
<FootNote>
This is the first test of footnotes.
<par>
This footnote (still #1 on the first page) is made up of more than one paragraph.
<FNoteDef 101>
<FootNote>
This footnote should appear at the bottom of the third page.
<EndFNotes>
<Title>
Maker Markup Language Specification
<Section>
<NumberFormat   ``<n=0>.0\t''>
<Marker <MType 9> <MText \Q0'>>Table of Contents
<TOCEntry>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q0'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q1'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q2'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q3'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q4'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q5'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q6'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q7'> <XRefSrcFile \Q'>>
<XRef <XRefName \QTOCEntry'> <XRefSrcText \Q8'> <XRefSrcFile \Q'>>
<Section>
<Marker <MType 9> <MText \Q1'>>Introduction
<Body>
Maker Markup Language (MML) is used to create formatted FrameMaker documents from a
standard (nongraphical) text file. MML allows access to many, but not all, of the
features of FrameMaker.<FNote 100>
<Section>
<Marker <MType 9> <MText \Q2'>>MML font styles
<BulletItem>
<italic>This line is italic<noitalic>
<oblique>(or oblique)<nooblique>
<bold>This line is bold.<nobold>
<underline>This line is underlined.<nounderline>
<strike>This line is struck out.<nostrike>
<overline>This line is overlined.<nooverline>
<changebar>This line has a changebar.<nochangebar>
<Section>
<Marker <MType 9> <MText \Q3'>><TestFont>Cross References
<TestPgf>
<Marker <MType 9>
        <MText \Qtest'>
><XRef  <XRefName \QTestXRef'>
        <XRefSrcText \Qtest'>
        <XRefSrcFile \Q'>
>
<Section>
<Marker <MType 9> <MText \Q4'>>Frame Math now accessible in MML
<Body>
<AutoNumber On>
<NumberFormat ``M:(<n+>)''>
<NumberFont      ````>
<NumAtEnd Yes>
```

```
<AFrame
        <Pen 15>
         <Fill 7>
         <PenWidth 1.0 pt>
         <Separation 0>
         <BRect 0.6'' 4.09715'' 1.44363'' 0.38889''>
         <FrameType Inline >
         <NSOffset 0.0''>
         <BLOffset -0.16319''>
         <AnchorAlign Center >
         <Cropped No >
         <Math
         <BRect 0.03686'' 0.04439'' 1.3699'' 0.30264''>
         <MathFullForm \Qequal[int[times[char[u], function[optotal[char[x]],
char[v]],diff[char[x]]]], plus[times[char[u], char[v]], minus[int[times[char[v],
function[optotal[char[x]], char[u]], diff[char[x]]]]]]]'> # end of MathFullForm
         <MathOrigin 0.72181'' 0.22569''>
         <MathAlignment Center >
         <MathSize MathMedium >
         <Angle 0>
         >
>
<Section>
<ColumnTop Yes>
<Marker <MType 9> <MText \Q5'>>Hard Spacing
<Code>
This is a test of hard spacing:
<HardSpaces On>local IntT scan_name_to_buffer() <HardReturn>
{                                               <HardReturn>
         IntT len;                              <HardReturn>
                                                <HardReturn>
         len = perhaps_scan_name_to_buffer();   <HardReturn>
         if (len \<= 1)                         <HardReturn>
         {                                      <HardReturn>
         PathT msg;                             <HardReturn>
                                                <HardReturn>
         start_error();                         <HardReturn>
         SrGet(MMLStrings, MMLNullName, msg);   <HardReturn>
         fprintf(errput, msg);                  <HardReturn>
         end_error();                           <HardReturn>
         buffer[0] = 255;                       <HardReturn>
         buffer[1] = 0;                         <HardReturn>
         len = 2; /* Bogus id */                <HardReturn>
 }                                              <HardReturn>
         return (len);                          <HardReturn>
}                                               <HardReturn>
<HardSpaces Off>
<Comment *** The following Body paragraph contains an anchored frame. The
         AFrame statement is equivalent to a MIF Frame statement (for a detailed
         description,see the Maker Interchange Format (MIF) Reference Manual). Inside
the
         frame is a star. We just show this here so you can see how it is done.
 >
<Section>
<Marker <MType 9> <MText \Q6'>>Anchored Frames and Graphics
```

```
<Body>
MML allows formatted documents to be created using both a <if>GENCODE <pf>style of
markup, in which document format and content are separate notions,<AFrame
         <BRect 0 0 4 3.25>
         <Polygon
         <Pen 0>
         <PenWidth 1.0 pt>
         <Fill 6>
         <Inverted No >
         <NumPoints 10>
         <Point 2.03'' 0.29''> <Point 2.19'' 0.83''> <Point 2.76'' 0.83''>
         <Point 2.28'' 1.17''> <Point 2.49'' 1.71''> <Point 2.03'' 1.36''>
         <Point 1.56'' 1.71''> <Point 1.76'' 1.15''> <Point 1.28'' 0.83''>
         <Point 1.86'' 0.83''>
         > # end of Polygon
         <TextRect
         <ID 30000>
         <Pen 0>
         <Fill 5>
         <BRect .125 2.125 3.75 1.0>
         >
> and a formatting style of markup, in which actual formatting specifications are
intermingled
with the document text.
<Comment *** Change text flow *** >
<TextFlow 30000>
<TFTag ````>
<TFAutoConnect   no>
<TFPostScript    no>
<TFSynchronized  no>
<TFLineSpacing   0.0pt>
<TFMinHangHeight 0.0pt>
<TFFeather       no>
<BeginFNotes>
<FNoteDef 102>
<FootNote>
This footnote should be in the box as well.
<EndFNotes>
<Title>
<Leading 0.0pt>
<SpaceBefore 0.0pt>
<SpaceAfter 0.0pt>
<par>
<Comment <HardReturn>>
<if>This text should be in the box.<pf><FNote 102>
<Comment *** Back to main text flow *** >
<TextFlow 0>
This document contains the following sections:
<BulletItem>
Instructions for creating MML documents
Overview of MML file format and syntax
Description of each MML Statement
Sample MML file
A test footnote<FNote 101>
<Include ``calendar.mml''>
```

```
<Section>
<Marker <MType 9> <MText \Q7'>>FrameMaker Tables
<ATable 4>
<Section>
<Marker <MType 9> <MText \Q8'>>Creating and Using MML Documents
<Body>
An MML document is a standard ASCII text file containing MML statements,
text broken up into paragraphs, and having ``.mml'' as the filename suffix.
It can be created using any standard text editor, such as vi or emacs.
It can also be created using Frame Maker 1.0 as a simple text editor: when
saving the document, specify Text Only in the Save dialog box.
<Comment *** Would be followed by additional such lines>
```

[Home Page | Download Postscript | Custom API | Conversion services | Previous | Forward]

MML Interpreter Messages

FrameMaker uses a program called mmltomif to read an MML file. mmltomif interprets the MML file and produces a temporary MIF file. While mmltomif is reading the MML file, it might detect errors such as unexpected character sequences. It responds by displaying error messages. Even if it finds an error, mmltomif continues to process the MML file and reads as much of the document as possible.

This section lists the messages produced by mmltomif, along with their explanations. Words in italic indicate variable words in a message. A line number is printed along with most messages when they appear on the screen.

MML MSG: Message_String.
Not an error; generated by a user <Message> statement.

MML: Bad Option: Character.
mmltomif did not recognize this option character. The option is ignored.

Bad Boolean: Unexpected_String.
mmltomif expected to see Yes or No. The value No is assumed.

Bad LRCD: Unexpected_String.
mmltomif expected to see Left, Right, Center, Decimal, or LeftRight.

Bad Real Number: Unexpected_Char.
A nonreal number character appeared in the middle of a real number.

Bad Style: Unexpected_String.
mmltomif expected to see Arabic, LCRoman, UCRoman, LCAlpha, or UCAlpha.

Bad Unit: Unexpected_String.
mmltomif expected to see a valid unit (inch, cm, and so forth).

Bad Spacing: Unexpected_String.
A unit or number used to define the spacing before or after a table or paragraph is not valid. See discussion under unit.

Bad Language: Unexpected_String.
Language specified in the <!DefinePar> statement is not supported by FrameMaker.

Bad TMB: Unexpected_String.
You made a table cell entry without a top, bottom, or middle definition.

Bad Place: Unexpected_String.
In your paragraph or table definition, the placement value given is invalid.

Bad Title Place: Unexpected_String.
The placement value given for the table title is invalid.

Bad Angle: Unexpected_String.
The font statement given for the font angle is invalid.

Bad Digit: Unexpected_String.
You have provided an invalid number for a command.

Bad Hex Digit: Unexpected_String.
You have expressed something in hexadecimal representation and the representation is invalid. An explanation of valid representations is under the DefineChar command.

Can't Find File filename.
mmltomif can't find the specified input file. Make sure that the file exists, and that you have read access to it; then try again.

Didn't Find End Comment n.
The comment that began on the specified line did not end by the time the file was completely read.

Can't Open File filename.
mmltomif couldn't find the named include file. Make sure that the file is in the correct format and that you have read access to it; then try again. If this message still appears, close some open files or windows and try again.

Cant Open Tmp File.
mmltomif couldn't open its temporary file. Make sure you have write access to
/tmp, your home directory, or the current directory; then try again. If this message still appears, close some open files or windows and try again.

Cant Write File filename.
mmltomif couldn't open the specified output file for writing. Make sure you have write access; then try again. If this message still appears, close some open files or windows and try again.

X Needs Ending Space.
Characters specified with \x must end with a space.

Expected String Unexpected_Char.
mmltomif expected to see a string starting with a double quotation mark (") rather than the unexpected character shown in the message.

Fatal.
mmltomif encountered a problem from which it can't recover. Write down the error message and contact your Frame technical support representative.

Input Stack Overflow.
There are too many nested include files (maybe in an include loop). The maximum nesting depth is 100.

Internal Error
mmltomif encountered an internal error. Please contact your Frame technical
support representative.

Junk At End: Junk_String.
mmltomif expected to see a a right angle bracket (>).

Keyword Too Long.
While looking for a macro name or other keyword, mmltomif found a very long token (over 1,000 characters). Check the MML file for a syntax error and try again.

Not Finished Defining Character.
mmltomif encountered a <!Define...> statement within a <!Define...> statement (for example, a <DefineChar> statement within a <!DefinePar> statement). You must finish the first <!Define...> statement before beginning another.mmltomif ignores the first <!Define...> statement and continues reading the file. The results, however, are not likely to be what you intended.

Out Of Memory
mmltomif was unable to complete the translation of the MML file because it ran out of memory. To free memory for mmltomif, quit some FrameMaker document windows or terminate other processes.

String Too Long.
A very long string was found. The maximum string length is 1000 characters; characters beyond that are truncated.

Tab Stops
The statements <TabsStopType>, <TabStopLeader>, and <TabStop> can appear only within a <TabStops> statement.

Too Many I Options.
The maximum number of -I options is 100. mmltomif exits.

Unexpected Right Brace.
A right angle bracket (>) with no matching left bracket (<) has appeared.

Undefined Macro: Macro_Name.
There is no definition for this macro. The undefined macro is ignored.

Undefined PgfTag: Paragraph_Name.
The default paragraph tag used for the table cell has not been previously defined in a <!DefinePar> statement.

Undefined TblTag: Table_Name.
No <!DefineTbl> statement exists for this table tag.

Undefined Ruling: Rule_Name.
The rule named has not been defined in a <!DefineRule> statement.

Null Name
There are name tag indicators, <>, with no name between them.

Hash Table Overflow
You have given more definitions (macro, paragraph, font, and/or table) than the program can accommodate. The limits are 10,000 for Unix systems and 3,000 for Macintosh.

Tab Buffer Overflow
You have entered too many tab stops in your paragraph definitions. The limit is 1000 characters.

Usage: mmltomif [-Ipath] [input [output]]
You started mmltomif in a shell window without any parameters. Restart
mmltomif with the parameters shown, where path is the path of the MML input file, input is the input filename, and output is the output filename.

[Home Page | Download Postscript | Custom API | Conversion Services | Previous]

SOFTLINE

Welcome to the SII Home Page

Softline International, Inc. (SII) is a full-service provider in the field of technological consulting.For the past fifteen years, we have been providing smart, permanent solutions for a wide variety of clientele. And as our services are designed to be cost-effective long-term solutions, many of our services are significantly lower than what you might find elsewhere.

Note: This web site is currently under development. Please contact us if you would like documentation or information not yet available on-line.

Custom Development Services

API Development Projects SII develops a variety of API programs for use with the Frame Products family.

Filter Development Projects SII has developed a number of filters for moving documents between various document formatting systems.

Other Development Projects SII is called upon to apply its extensive cross-platform expertise to a wide variety of applications for its clients.

Document Production Automation SII works to automate production systems which integrate a number of different platforms.

Software Products

SII has several of its popular filters and Frame API's available as software packages.

- Filters
- Frame API's (Application Program Interfaces)

Publishing Consulting

With the dramatic changes occurring in the publishing industry, SII has become a major provider of such services including:

- FrameMaker Migration Planning
- FrameMaker Template Services
- Publishing Procedures and Methodologies
- Web Server Design and Implementation
- Tool Selection
- Hardware Planning

Document Conversion Services

SII operates a service bureau which performs conversions between a wide variety of formats.

Support Services

We provide system programming support, design and implement ways to control the cost of system programming, and provide technical support to clients implementing new technologies.

Company Information

Coming Soon:

Frame Tipsheets A number of resources for enhancing your use of Frame products.

Ask Claudia SII's Frame guru who offers opinions and tips for Frame users.

API Shopper A comprehensive shopping guide for locating the perfect API

[Contact Us]

FrameMaker APIs

Softline International (SII) offers a variety of FrameMaker API services. Listed below is a partial list of APIs which SII has devloped in the past. It is a representative sample only.

SII APIs always include the following:

- Source Code
- Complete documentation (in FrameMaker format)
- Ninety day warranty included in price
- Maintenance available after initial 90 day warranty

1. Program to audit paragraph format utilization. API is installed as report type API. The report is divided into three parts. The first produces paragraph utilization statistics as follows:

   ```
   Paragraph Tag     Catalogue     Authorized     Total Times     Times Used
                       (Y/N)         (Y/N)           Used        w/Overrides

   xxxxxxxxxxxx          x             x            9999999        9999999
   ```

   The second part lists the location of paragraphs with format overrides as follows:

   ```
   Paragraph Tag     Document Name     Page Number

   xxxxxxxxxxxx      xxxxxxxxxxxx           99
   ```

   The third part lists the location of paragraphs that were not contained in a user provided list of authorized paragraph tags as follows:

   ```
   Paragraph Tag     Document Name     Page Number

   xxxxxxxxxxxx      xxxxxxxxxxxx           99
   ```

   Program can be run at document or book level. Program can also run in batch mode at document or book level. List of authorized paragraphs is supplied in an INI file. INI file also has a flag to indicate whether to produce exception reports.
2. Need API to import ASCII text files based upon marker inserted into the document. API is used to import sample program into a manual from the latest versions prepared by the programmers for the system being documented. Formatting is limited to the following items:

  - Program selects paragraphs to format the item based on the longest line in the input file. Each line in the input file needs to fit in one line in the FramMaker document. The INI file contains information to make the paragraph tag selcetion based on largest input line.
  - Program replaces all spaces with HardSpaces.
  - Program places a hard return at the end of each input line.

3. API to freeze pagination in all documents in a book file. API runs in batch or from Book Menu.
4. API to selectively copy varaiable values from a template file to all documents in a book. API runs in batch from Book Menu. API does the following steps:
   - Program copies User Variables only. System Variables are not changed.
   - Program optionally deletes any user variables not contained in the new template.
5. API to turn show/hide settings on/off at the book level. API runs in batch or from Book Menu.
6. API to generate a list of changed pages. Changed pages are identified by the fact that the change date of each page is equal to or greater than a date entered by the user. Report looks as follow:

```
Document Name     Change Date     Page Number

xxxxxxxxxxxx       99/99/99          99.9
```

7. API to print changed pages. Changed pages are identified by the fact that the change date of each page is equal to or greater than a date entered by the user, API runs in batch or from the Book Menu.
8. API to compare a document to a baseline document and list any discrepancies. The baseline document only contains certain paragraphs (basically an outline). Only paragraphs of the types contained in the outline document are compared. The outline document covers the whole book. Where the API may be asked to only check a particular document. As such the program must be able to find a starting point from which to begin comparing the baseline document. A report of discrepancies should be produced. API runs at the book or document level and is capable of running in batch mode.
9. API which automatically updates portions of document text from a repository of text fragments. These fragments are also used to generate online help files and other noin-FrameMaker documentation. Repository allows a writer to prepare one edition of given text fragment (such as a description of customer number) and have that item updated in all documents in which it appears. Repository is maintained as an Oracle database. Text reformatting occurs as a part of included item.
10. API to automatically update certain canned text in documents (i.e., copyright notices, safety notices, disclaimers) with the latest edition of these items. In this case the canned text is maintained in other FrameMaker documents set up sepcifically for this purpose.
11. API to automatically create index markers for all occurrences a list of words to be indexed.
12. API to automatically create hypertext links between all occurances of glossary terms and their glossary definitions contained in the glossary of a book.

In addition to writing single APIs SII has extensive experience packaging groups of APIs together with other programs, shell scripts, and utilities to form a total publishing solution. Some examples of these solutions includes:

1. A document distribution system which once a document has been approved for distribution automatically produces a standalone version of the document, loads a copy of the document to a fax server, loads a copy of the document for the Word Wide Web, and includes the document at the appropriate point in a larger book.

   This is all done from a single set of menus where the user indicates that the document is released for distribution. The system is so sophiscated that the postscript file produced for the stand alone documents is forward to an off-site printer along with a work request to get the document printed.
2. A system which prepares books for electronic publishing on CD ROM and World Wide Web. This includes automatically scanning the documents for items which will not publish well electronically and altering these items as required. All graphics are also converted to a format more suitable for electronic publishing.

For more information, please contact webmaster@softline.com or call (510) 849-9817.

[Home Page | Download Postscript | Custom API]

Warranty

The SCRIPT to FrameMaker, SCRIBE to FrameMaker, and TEX to FrameMaker Filters will conform, when shipped to the customer, to its Licensed Program Specifications which are in effect at that time, provided it is properly used in a Specified Operating Environment. If the Customer believes there is a defect in the program such that it does not meet its Licensed Program Specifications, the Customer must notify Softline while Software Services are available for the program. Softline does not warrant that the functions contained in the program will meet the CustomerÕs requirements or will operate in the combinations which may be selected for use by the Customer or that the operation of the program will be uninterrupted or error free or that all program defects will be corrected.

The foregoing warranties are in lieu of all other warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Licensed Software Specifications may be updated from time to time and such updates may constitute a change in specifications.

For Distributed Systems License Options(DSLO) Licenses, warranty service, if any will be provided through the Basic License location.

Following discontinuance of all program services, the program will be distributed on an ÒAs IsÓ basis without warranty of any kind either express or implied.

Additional Information

Softline International Inc.
2140 Shattuck Avenue, Suite 2009
Berkeley, CA 94704 USA
phone: (510) 849-9817
fax: (510) 849-0156

[Home Page | Copyright | Custom API | In-House Services]

Copyright Information

Important Notice

Printed in the United States of America.

This book was written, illustrated, and produced using FrameMaker® workstation publishing software and the ITC Garamond®, Helvetica® Condensed, and Memphis® families of typefaces.

Project team: Judy Alston, Leah Anderson, Paul Bailey, Don Bennett, Jeff Brown, Terry Bush, Lisa Caras, Teresa Chen, Cam Clarke, Greg Cockroft, Abe Conant, Cary Duncan, Maureen Franotovich, Kathy Fronsdahl, Louise Galindo, Jeff Gardiner, Eric Griswold, Carlos Gurr, Chris Guthrie, Chung-Pang Lai, Susan Lamoree, Roger Lee, Ed Malick, Diane Manning, Jim Melani, Michael Milligan, Will Naber, Kathy Paxton, Aric Rubin, Bob Silva, Randy Strauss, Kathy Tansill, Victoria Thomas, Craig Yappert, Günter Zimmerman

---

[Home Page | Warranty | Custom API | In-House Services]