

VAX DOCUMENT

Using Doctypes and Related Tags

Order Number: AA-JT86B-TE

February 1991

This manual describes the VAX DOCUMENT **doctypes** and the tags specific to each of those doctypes. Tags are described by the doctype in which they can be used.

Revision/Update Information: This revised manual supersedes the *VAX DOCUMENT User Manual, Volume 2 Version 1.1* (Order Number AA-JT86A-TE).

Operating System and Version: VMS Version 5.3 or higher.

Software Version: VAX DOCUMENT Version 2.0

First printing, July 1987
Revised, July 1988
Revised, November 1990
Revised, February 1991

Copyright ©1987, 1988, 1990, 1991 Digital Equipment Corporation

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem-10	PDP	VT
DECSYSTEM-20	PDT	
DECUS	RSTS	
DECwriter	RSX	

The following is a third-party trademark:

PostScript is a registered trademark of Adobe Systems, Inc.

This document is available on CDROM.

ZK5353

Contents

PREFACE	xiii
NEW FEATURES	xvii
CHAPTER 1 OVERVIEW OF THE DOCTYPE-SPECIFIC TAGS	1-1
1.1 USING DOCTYPES AND DOCTYPE-SPECIFIC TAGS	1-2
CHAPTER 2 USING THE ARTICLE DOCTYPE	2-1
2.1 ARTICLE DOCTYPE COMMON ELEMENTS	2-2
2.1.1 Titles and Subtitles	2-3
2.1.2 Author Information	2-3
2.1.3 Abstracts, Source Notes, and Acknowledgments	2-4
2.1.4 Headings	2-5
2.1.5 Running Titles and Running Feet	2-5
2.1.6 Quotations	2-6
2.1.7 Numbered Notes	2-6
2.1.7.1 Back Notes • 2-7	
2.1.7.2 Reference Notes • 2-7	
2.1.8 Bibliographies	2-8
2.2 IMPROVING THE FORMAT OF A 2-COLUMN DOCTYPE	2-8
2.2.1 Line Breaks in Columns	2-8
2.2.2 Wide Tables and Examples	2-9
2.2.3 Final Adjustment of Column and Page Breaks	2-9
2.3 A SAMPLE USE OF THE ARTICLE DOCTYPE TAGS	2-12
2.4 ARTICLE DOCTYPE TAG REFERENCE	2-16
<ABSTRACT>	2-17
<ACKNOWLEDGMENTS>	2-18
<AUTHOR>	2-19
<AUTHOR_ADDR>	2-20
<AUTHOR_AFF>	2-21
<AUTHOR_LIST>	2-22
<BACK_NOTE>	2-23

Contents

<BACK_NOTES>	2-25
<BIBLIOGRAPHY>	2-26
<BIB_ENTRY>	2-27
<COLUMN>	2-28
<DOCUMENT_ATTRIBUTES>	2-30
<QUOTATION>	2-32
<REF_NOTE>	2-33
<REF_NOTES>	2-35
<RUNNING_FEET>	2-36
<RUNNING_TITLE>	2-37
<SOURCE_NOTE>	2-39
<SUBTITLE>	2-40
<TITLE>	2-41
<TITLE_SECTION>	2-42
<VITA>	2-43

CHAPTER 3 USING THE HELP DOCTYPE 3-1

3.1	CREATING A HELP FILE	3-1
3.1.1	How HELP Interprets Reference Sections _____	3-1
3.1.2	How HELP Interprets the Input File _____	3-2
3.1.3	How to Selectively Include and Exclude Text for Help Output _____	3-3
3.1.4	How to Handle Special Cases _____	3-3
3.2	HOW TO READ THE HELP FILE ONLINE	3-3
3.3	HELP DOCTYPE TAG REFERENCE	3-4
	<BOOK_ONLY>	3-5
	<HELP_ONLY>	3-6
	<KEEP_HELP_LEVEL>	3-7
	<SET_HELP_LEVEL>	3-9

CHAPTER 4 USING THE LETTER DOCTYPE 4-1

4.1	SAMPLE USES OF THE LETTER DOCTYPE TAGS	4-3
4.1.1	A Sample Memo _____	4-4
4.1.2	A Sample Letter _____	4-6

<hr/>		
4.2	LETTER DOCTYPE TAG REFERENCE	4-8
	<CC>	4-9
	<CCLIST>	4-11
	<CLOSING>	4-12
	<DISTLIST>	4-13
	<FROM_ADDRESS>	4-14
	<MEMO_DATE>	4-15
	<MEMO_FROM>	4-17
	<MEMO_HEADER>	4-18
	<MEMO_LINE>	4-19
	<MEMO_TO>	4-21
	<SALUTATION>	4-22
	<SUBJECT>	4-23
	<TO_ADDRESS>	4-24
<hr/>		
CHAPTER 5 USING THE MANUAL DOCTYPE		5-1
<hr/>		
5.1	EXAMPLE OF USING THE MANUAL DOCTYPE	5-4
<hr/>		
CHAPTER 6 USING THE MILSPEC DOCTYPE		6-1
<hr/>		
6.1	MILSPEC TEMPLATE FILES	6-2
<hr/>		
6.2	MILSPEC DOCTYPE CONFORMANCE AND FORMAT	6-2
6.2.1	Example of Using the MILSPEC.SECURITY and MILSPEC.DRAFT Doctypes	6-4
<hr/>		
6.3	CREATING MIL-STD-490A DOCUMENTS	6-11
<hr/>		
6.4	CREATING DATA ITEM DESCRIPTION DOCUMENTS	6-12
6.4.1	Creating DOD-STD-2167 Documents	6-12
6.4.2	Creating DOD-STD-2167A Documents	6-12
6.4.2.1	Using the Data Item Description Template Files	6-13
<hr/>		
6.5	MILSPEC DOCTYPE TAG REFERENCE	6-17
	<CODE_EXAMPLE>	6-18
	<DOCUMENT_ATTRIBUTES>	6-20
	<HEADN>	6-23
	<HIGHEST_SECURITY_CLASS>	6-25
	<RUNNING_FEET>	6-26
	<RUNNING_TITLE>	6-27

Contents

<SECURITY>	6-28
<SET_APPENDIX_NUMBER>	6-31
<SET_CONTENTS_SECURITY>	6-33
<SET_PAGE_SECURITY>	6-35
<SET_SECURITY_CLASS>	6-37
<SIGNATURE_LINE>	6-39
<SIGNATURE_LIST>	6-40
<SPECIFICATION_INFO>	6-42
<SPEC_TITLE>	6-44
<SUBTITLE>	6-45

CHAPTER 7 USING THE ONLINE DOCTYPE 7-1

7.1	ONLINE DOCTYPE TAG REFERENCE	7-1
	<BOOK_ONLY>	7-2
	<BOOK_REF>	7-3
	<EXTENSION>	7-4
	<HOTSPOT>	7-6
	<HELP_ONLY>	7-9
	<KEEP_HELP_LEVEL>	7-10
	<LMF>	7-12
	<LMF_ALTNAME>	7-15
	<LMF_INFO>	7-16
	<LMF_PRODUCER>	7-18
	<LMF_PRODUCT>	7-19
	<LMF_RELEASE_DATE>	7-20
	<LMF_VERSION_NUMBER>	7-21
	<ONLINE_CHUNK>	7-22
	<ONLINE_POPUP>	7-24
	<ONLINE_TITLE>	7-26
	<SET_HELP_LEVEL>	7-28
	<SET_ONLINE_TOPIC>	7-30
	<SHELF_CREATE>	7-33
	<SHELF_REF>	7-35

CHAPTER 8 USING THE OVERHEADS DOCTYPE 8-1

8.1	A SAMPLE USE OF THE OVERHEADS DOCTYPE TAGS	8-3
-----	--	-----

8.2	OVERHEADS DOCTYPE TAG REFERENCE		8-7
	<AUTHOR_INFO>	8-8	
	<AUTO_NUMBER>	8-9	
	<INTRO_SUBTITLE>	8-10	
	<INTRO_TITLE>	8-11	
	<RUNNING_FEET>	8-12	
	<RUNNING_TITLE>	8-14	
	<SLIDE>	8-16	
	<SUBTITLE>	8-18	
	<TEXT_SIZE>	8-19	
	<TITLE>	8-21	
	<TOPIC>	8-22	

CHAPTER 9 USING THE REPORT DOCTYPE **9-1**

9.1	CHARACTERISTICS OF THE REPORT DESIGN		9-1
------------	---	--	------------

9.2	SAMPLE USE OF THE REPORT DOCTYPE TAGS		9-3
------------	--	--	------------

9.3	A SAMPLE USE OF THE REPORT.TWOCOL DOCTYPE TAGS		9-7
------------	---	--	------------

9.4	REPORT DOCTYPE TAG REFERENCE		9-10
	<AUTHOR>	9-11	
	<BYLINE>	9-13	
	<COLUMN>	9-15	
	<DOCUMENT_ATTRIBUTES>	9-17	
	<LEVEL>	9-19	
	<OUTLINE>	9-20	
	<RUNNING_FEET>	9-22	
	<RUNNING_TITLE>	9-23	
	<SECTION>	9-25	
	<SHOW_LEVELS>	9-26	
	<SIGNATURES>	9-28	

CHAPTER 10 USING THE SOFTWARE DOCTYPE **10-1**

10.1	CHARACTERISTICS OF THE SOFTWARE DESIGNS		10-3
-------------	--	--	-------------

10.2	COMMON SOFTWARE DESCRIPTION TASKS		10-6
-------------	--	--	-------------

Contents

10.3	DOCUMENTING TERMINAL KEYS AND KEYPADS	10-6
10.3.1	Describing Individual Keys _____	10-7
10.3.2	Describing Keypads and Keypad Keys _____	10-9
10.4	DOCUMENTING CODE FRAGMENTS	10-13
10.5	DOCUMENTING SOFTWARE MESSAGES	10-14
10.6	DOCUMENTING ARGUMENTS, PARAMETERS, AND QUALIFIERS	10-18
10.7	CREATING A SERIES OF INTERACTIVE OR CODE EXAMPLES	10-22
10.8	USING THE REFERENCE TEMPLATES	10-23
10.9	CREATING YOUR OWN REFERENCE TEMPLATE TAGS	10-27
10.10	CREATING YOUR OWN TEMPLATE TABLES	10-28
10.11	MODIFYING THE REFERENCE TEMPLATES	10-30
10.12	MODIFYING DEFAULT HEADINGS IN A TEMPLATE	10-30
10.13	USING THE TEMPLATE-ENABLING TAGS	10-32
10.13.1	Template-Enabling Tag Behavior in the SOFTWARE.SPECIFICATION Doctype _____	10-33
10.14	USING THE <SET_TEMPLATE_templatename> TAGS	10-35
	FIND_FIRST _____	10-37
10.15	USING THE COMMAND TEMPLATE	10-37
10.15.1	Sample SDML File of the Command Template _____	10-39
10.15.2	Sample Output File of the Command Template _____	10-41
	APPEND _____	10-42
10.16	USING THE ROUTINE TEMPLATE	10-45
10.16.1	Sample SDML File of the Routine Template _____	10-47

10.16.2	Sample Output File of the Routine Template	10-49
	\$ENQ	10-50
	MTH\$XSQRT	10-52
<hr/>		
10.17	USING THE STATEMENT TEMPLATE	10-53
10.17.1	Sample SDML File of the Statement Template	10-54
10.17.2	Sample Output File of the Statement Template	10-55
	RECORD	10-56
	MID\$	10-57
<hr/>		
10.18	USING THE TAG TEMPLATE	10-58
10.18.1	Sample SDML File of the Tag Template	10-60
10.18.2	Sample Output File of the Tag Template	10-61
	<SYNTAX>	10-62
<hr/>		
10.19	THE SOFTWARE DOCTYPE TAGS	10-64
	<ARGDEF>	10-65
	<ARGDEFLIST>	10-66
	<ARGITEM>	10-69
	<ARGTEXT>	10-71
	<ARGUMENT>	10-73
	<ARG_SEP>	10-74
	<AUTHOR>	10-75
	<BYLINE>	10-77
	<COMMAND>	10-79
	<COMMAND_SECTION>	10-81
	<CONSTRUCT>	10-85
	<CONSTRUCT_LIST>	10-87
	<CPOS>	10-90
	<DELETE_KEY>	10-91
	<DESCRIPTION>	10-92
	<DISPLAY>	10-94
	<DOCUMENT_ATTRIBUTES>	10-96
	<EXAMPLE_SEQUENCE>	10-98
	<EXAMPLES_INTRO>	10-100
	<EXC>	10-101
	<EXI>	10-102
	<EXTTEXT>	10-104
	<FARG>	10-105
	<FARGS>	10-107
	<FCMD>	10-109
	<FFUNC>	10-112
	<FORMAT>	10-114
	<FORMAT_SUBHEAD>	10-116
	<FPARM>	10-117
	<FPARMS>	10-119
	<FRTN>	10-121
	<FTAG>	10-123
	<FUNCTION>	10-125

Contents

<GRAPHIC>	10-126
<KEY>	10-127
<KEYPAD>	10-129
<KEYPAD_ENDROW>	10-132
<KEYPAD_ROW>	10-133
<KEYPAD_SECTION>	10-134
<KEY_NAME>	10-137
<KEY_PLUS>	10-138
<KEY_SEQUENCE>	10-139
<KEY_TYPE>	10-141
<MESSAGE_SECTION>	10-142
<MESSAGE_TYPE>	10-145
<MSG>	10-146
<MSGS>	10-148
<MSG_ACTION>	10-150
<MSG_FACILITY>	10-151
<MSG_SEVERITY>	10-152
<MSG_TEXT>	10-153
<OVERVIEW>	10-154
<PARAMDEF>	10-155
<PARAMDEFLIST>	10-156
<PARAMITEM>	10-159
<PROMPT>	10-161
<PROMPTS>	10-163
<QPAIR>	10-165
<QUALDEF>	10-166
<QUALDEFLIST>	10-167
<QUALITEM>	10-169
<QUAL_LIST>	10-171
<QUAL_LIST_DEFAULT_HEADS>	10-175
<QUAL_LIST_HEADS>	10-177
<RELATED_ITEM>	10-178
<RELATED_TAG>	10-179
<RELATED_TAGS>	10-180
<RESTRICTIONS>	10-182
<RETTEXT>	10-184
<RETURNS>	10-185
<RETURN_VALUE>	10-187
<RITEM>	10-188
<ROUTINE>	10-189
<ROUTINE_SECTION>	10-191
<RSDEFLIST>	10-196
<RSITEM>	10-198
<RUNNING_FEET>	10-199
<RUNNING_TITLE>	10-200
<SDML_TAG>	10-202
<SET_TEMPLATE_ARGITEM>	10-203
<SET_TEMPLATE_COMMAND>	10-206
<SET_TEMPLATE_HEADING>	10-209
<SET_TEMPLATE_LIST>	10-211
<SET_TEMPLATE_PARA>	10-213
<SET_TEMPLATE_ROUTINE>	10-215
<SET_TEMPLATE_STATEMENT>	10-218

<SET_TEMPLATE_SUBCOMMAND>	10-220
<SET_TEMPLATE_TABLE>	10-222
<SET_TEMPLATE_TAG>	10-225
<SIGNATURES>	10-227
<STATEMENT>	10-228
<STATEMENT_FORMAT>	10-229
<STATEMENT_LINE>	10-231
<STATEMENT_SECTION>	10-234
<SUBCOMMAND>	10-238
<SUBCOMMAND_SECTION>	10-240
<SUBCOMMAND_SECTION_HEAD>	10-241
<SYNTAX>	10-242
<SYNTAX_DEFAULT_HEAD>	10-244
<TAG_SECTION>	10-246
<TERMINATING_TAG>	10-250

INDEX

EXAMPLES

6-1	Coding a 2167A-Formatted Document _____	6-13
-----	---	------

FIGURES

2-1	ARTICLE Doctype Design _____	2-1
2-2	ARTICLE Doctype Output Example, Page 1 _____	2-14
2-3	ARTICLE Doctype Output Example, Page 2 _____	2-15
4-1	LETTER Doctype Design _____	4-1
4-2	LETTER Doctype Output Example for Memo _____	4-5
4-3	LETTER Doctype Output Example for Letter _____	4-7
5-1	MANUAL Doctype Designs _____	5-1
5-2	MANUAL Doctype Output Example, Title Page _____	5-5
5-3	MANUAL Doctype Output Example, Interior Page _____	5-6
6-1	MILSPEC Doctype Designs _____	6-1
6-2	MILSPEC.SECURITY Doctype Output Example, Title Page _____	6-6
6-3	MILSPEC.SECURITY Doctype Output Example, Interior Page _____	6-7
6-4	MILSPEC.DRAFT Doctype Output Example, Title Page _____	6-8
6-5	MILSPEC.DRAFT Doctype Output Example, Interior Page 1 _____	6-9
6-6	MILSPEC.DRAFT Doctype Output Example, Interior Page 2 _____	6-10
8-1	OVERHEADS Doctype Designs _____	8-1
8-2	OVERHEADS Doctype Output Example, First Slide _____	8-5
8-3	OVERHEADS Doctype Output Example, Second Slide _____	8-6
9-1	REPORT Doctype Designs _____	9-1
9-2	REPORT Doctype Output Example, Title Page _____	9-5
9-3	REPORT Doctype Output Example, Interior Page _____	9-6

Contents

9-4	REPORT.TWOCOL Doctype Output Example, Title Page _____	9-8
9-5	REPORT.TWOCOL Doctype Output Example, Interior Page _____	9-9
10-1	SOFTWARE Doctype Designs _____	10-2

TABLES

1	Conventions Used in this Manual _____	xiv
1-1	Supported Doctypes _____	1-1
2-1	Page Layout of the ARTICLE Design _____	2-1
2-2	Tags Available in the ARTICLE Doctype _____	2-2
2-3	Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag _____	2-30
4-1	Page Layout of the LETTER Doctype Design _____	4-1
4-2	Tags Available in the LETTER Doctype _____	4-2
5-1	Page Layout of the MANUAL.GUIDE Doctype Design _____	5-2
5-2	Page Layout of the MANUAL.PRIMER Design _____	5-2
5-3	Page Layout of the MANUAL.REFERENCE Design _____	5-2
6-1	Page Layout of the MILSPEC Designs _____	6-2
6-2	MILSPEC Doctype Tags _____	6-3
6-3	MILSPEC Doctype DOD-STD-2167 Data Item Description Templates _____	6-15
6-4	MILSPEC.SECURITY Doctype DOD-STD-2167A Data Item Description Templates _____	6-17
6-5	Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag _____	6-21
8-1	Page Layout of the OVERHEADS Design _____	8-1
8-2	Page Layout of the OVERHEADS.35MM Design _____	8-2
8-3	Tags Available in the OVERHEADS Doctype _____	8-2
9-1	Page Layout of the REPORT Doctype Design _____	9-1
9-2	Page Layout of the REPORT.TWOCOL Doctype Design _____	9-2
9-3	Tags Available in the REPORT Doctype _____	9-3
9-4	Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> tag _____	9-18
10-1	Page Layout of the SOFTWARE.BROCHURE Doctype Design _____	10-3
10-2	Page Layout of the SOFTWARE.GUIDE Doctype Design _____	10-3
10-3	Page Layout of the SOFTWARE.HANDBOOK Doctype Design _____	10-4
10-4	Page Layout of the SOFTWARE.POCKET_REFERENCE Doctype Design _____	10-4
10-5	Page Layout of the SOFTWARE.REFERENCE Doctype Design _____	10-5
10-6	Page Layout of the SOFTWARE.SPECIFICATION Doctype Design _____	10-5
10-7	Default Headings of Reference Template Tags _____	10-31
10-8	Command Template Tags as Available from DOC\$TEMPLATES _____	10-37
10-9	Routine Template Tags as Available from DOC\$TEMPLATES _____	10-45
10-10	Statement Template Tags as Available from DOC\$TEMPLATES _____	10-53
10-11	Tag Template Tags as Available from DOC\$TEMPLATES _____	10-58
10-12	Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag _____	10-97

Preface

Document Structure

This manual describes the VAX DOCUMENT **doctype-specific** tags that are restricted to certain **document types (doctypes)**. These tags are not available in all doctypes.

With each discussion of doctype-specific tags, an example shows how you use those tags in an input file, and what the output is when you process that input file.

The description of tags specific to one style also contains reference information about the tags. This reference information describes the syntax and any restrictions associated with each tag. It also provides an example of the tag used in a Standard DIGITAL Markup Language **SDML** file.

Chapter 1 provides an overview of all the doctypes discussed in this manual. Subsequent chapters are ordered alphabetically by doctype name, and discuss a specific doctype and its designs. Each chapter provides examples of tag use in SDML files and corresponding processed outputs.

Intended Audience

This book is intended for writers, editors, and general users who want to produce articles, business letters and memos, military specifications, technical manuals, reports, documentation that you can read with Bookreader, overhead slides, or software documentation using VAX DOCUMENT. You should be familiar with a Digital text editor.

Associated Documents

This manual is part of the VAX DOCUMENT documentation set that includes the following books:

- *VAX DOCUMENT Producing Online and Printed Documentation*
- *VAX DOCUMENT Using Global Tags*
- *VAX DOCUMENT Designing Doctypes*
- *VAX DOCUMENT Tags Quick Reference*
- *VAX DOCUMENT Quick Reference Card*
- *VAX DOCUMENT Graphics Editor User's Guide*
- *VAX DOCUMENT Installation Guide*

Conventions

In *VAX DOCUMENT Using Doctypes and Related Tags*, the discussion of each tag follows a fixed order. First, the name of the tag is followed by a brief overview that describes the purpose of the tag. Following the overview is a syntax section that displays the syntax of the tag: any required or optional arguments, any related tags, any restrictions on the use of the tag, and any required terminators to the tag, if needed.

The category of “related tag” is defined broadly. A tag is related to the tag under discussion if one of the following criteria is met:

- It is required for use of the tag under discussion.
- It marks a text element of the same kind as the tag under discussion.
- It is commonly used with the tag under discussion.

Note: The related tags, restrictions, and required terminator sections are omitted if there is no relevant information.

Following the syntax section is a description section. The description expands the overview and presents more detailed information on using the tag.

The discussion of a tag concludes with at least one example, or a reference to an example. The example shows how to code the tag in an SDML file.

Each output example is introduced by the sentence “This example produces the following output.”. Output examples may vary, however, depending on the doctype you use and on whether any doctype modifications have been made to your local installation of *VAX DOCUMENT*.

Table 1 lists the typographical conventions used in this manual.

Table 1 Conventions Used in this Manual

Convention	Meaning
.	In examples, a vertical ellipsis represents the omission of data that the system displays in response to a command or to data you enter.
...	In examples, a horizontal ellipsis indicates that you can enter additional parameters, values, or other information. In tag syntax, a horizontal ellipsis indicates that arguments to the tag have been omitted.
TERM	A term that appears in bold type is defined in the glossary in the <i>VAX DOCUMENT Producing Online and Printed Document</i> .
<TAG>(argument)	Parentheses enclose an argument to a tag. A lowercase word as an argument to a tag indicates that a user-specified argument must be entered.
<TAG>[(argument)]	Brackets indicate that the enclosed argument to the tag is optional.

Table 1 (Cont.) Conventions Used in this Manual

Convention	Meaning
<TAG>[(argument-1 [\argument-2])]	This tag syntax indicates that both arguments are optional. Only if you use <i>argument-1</i> , however, do you have the choice of using <i>argument-2</i> .
<TAG>[[([argument-1] \ [argument-2])]	This tag syntax indicates that both arguments are optional. You can use either argument as the first argument.
<TAG> [({ argument-1 KEYWORD-1 [\KEYWORD-2 })]	This tag syntax indicates that all arguments are optional. The braces indicate a choice between <i>argument-1</i> and KEYWORD-1. You must choose your first argument. If you choose KEYWORD-1, you also have the option of indicating KEYWORD-2 as the second argument.
<TAG>(KEYWORD)	An uppercase word within an argument to a tag indicates that the word is a keyword, and that you must enter the specific keyword.
<TAG>(\KEYWORD)	A keyword following a backslash in an argument to a tag indicates that that keyword must follow a backslash.
\	A backslash separates multiple arguments to a tag.

New Features

The following list contains the new features of VAX DOCUMENT Version 2.0.

- Support for converting SDML files into VMS Helpfile format with a HELP doctype.
- Revised and extended documentation. There is a completely new user's guide, a new summary of the SDML tags, a new quick-reference card, and a completely revised set of reference documentation.
- A number of new tags added for creating a book with Bookreader.
- Several doctypes developed to use with Bookreader.
- Several new tags added for getting **License Management Facility (LMF)** information into Bookreader books.
- Extensions in MILSPEC security marking.

1

Overview of the Doctype-Specific Tags

This manual describes the VAX DOCUMENT doctypes and their associated tags. These doctype-specific tags are restricted to certain doctypes and so are not available in all doctypes.

For example, the <SALUTATION> tag is a doctype-specific tag available only in the LETTER doctype. This tag is restricted to the LETTER doctype because it is only when you write a letter or memo that you want the output format associated with this tag. VAX DOCUMENT limits certain tags to specific doctypes to make the system more modular and to reduce the number of tags you must learn to use.

Using this Manual

The doctypes and doctype-specific tags are described in this manual in Chapters 2 through 10. As shown in Table 1-1, these chapters are ordered alphabetically by doctype name. Each chapter provides an overview of a specific doctype and its **designs**. It describes how to use that doctype and what doctype-specific tags and **global tags** are available within it (some doctypes do not use all the global tags).

The description of doctype-specific tags also contains reference information about the tags. The reference information describes the syntax and restrictions associated with each tag.

Each chapter includes an input and output sample of at least one SDML file that uses these tags.

Table 1-1 lists the supported doctypes, explains what each doctype is generally used for, and tells which chapters in this manual describe them.

Table 1-1 Supported Doctypes

Doctype Keyword	Used to Create	Tutorial/Reference Chapter
ARTICLE	Two-column articles	Chapter 2
LETTER	Letters and memos	Chapter 4
HELP	.HLP files used for online Help	Chapter 3
MANUAL	User manuals about topics other than software	Chapter 5
MILSPEC	Military specifications	Chapter 6
ONLINE	Output that can be read with the DECwindows Bookreader	Chapter 7

Overview of the Doctype-Specific Tags

Table 1-1 (Cont.) Supported Doctypes

Doctype Keyword	Used to Create	Tutorial/Reference Chapter
OVERHEADS	Transparencies for overhead or 35mm slides	Chapter 8
REPORT	General-purpose documents or formal outlines	Chapter 9
SOFTWARE	User manuals containing software-specific information	Chapter 10

1.1 Using Doctypes and Doctype-Specific Tags

The language you use to mark up a file for processing through VAX DOCUMENT is called generic because it is used, unchanged, to produce any type of document. However, part of using VAX DOCUMENT involves specifying a doctype to create the particular document format you want. For example, you want a different format for a manual than you want for a letter.

VAX DOCUMENT automatically specifies the correct formatting instructions for a chosen doctype, freeing you from the task of formatting text and letting you concentrate on the task of writing.

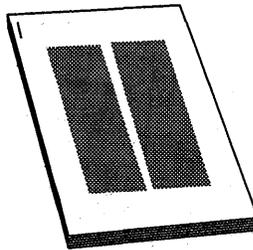
All your writing is done in an input file, which is also called an SDML file because of its file extension of .SDML. When you create a file to be processed through VAX DOCUMENT, always give the file an .SDML extension.

2

Using the ARTICLE Doctype

The ARTICLE doctype has one design, shown in Figure 2-1. It lets you create 2-column articles in an $8\frac{1}{2} \times 11$ -inch format with numbered or unnumbered headings. Process files under this doctype by using the ARTICLE doctype keyword on the DOCUMENT command line.

Figure 2-1 ARTICLE Doctype Design



ZK-1803A-GE

Table 2-1 lists the page layout characteristics of the ARTICLE doctype design.

Table 2-1 Page Layout of the ARTICLE Design

Page Layout Characteristics	
Running heads	None
Running feet	Page number (title text, optional)
Page numbering	Sequential
Trim size	8 1/2 x 11 inches
Page layout	Two columns
Right margin	Justified

Text Element Characteristics	
Headings	Unnumbered
Paragraphs	Indent first line
Figures, tables, and examples	Numbered

Table 2-2 summarizes the tags available in the ARTICLE doctype and briefly describes each tag.

Using the ARTICLE Doctype

Table 2-2 Tags Available in the ARTICLE Doctype

Tag Name	Description
<ABSTRACT>	Creates an article abstract.
<ACKNOWLEDGMENTS>	Creates an acknowledgments section in an article.
<AUTHOR>	Specifies the author of an article.
<AUTHOR_ADDR>	Specifies the address of the author.
<AUTHOR_AFF>	Specifies the organizational affiliation of the author.
<AUTHOR_LIST>	Creates a list of authors for an article with multiple authors.
<BACK_NOTE>	Creates a back note entry, and creates a superscript reference number in the article text.
<BACK_NOTES>	Causes any accumulated back notes to be output.
<BIBLIOGRAPHY>	Begins a bibliography.
<BIB_ENTRY>	Specifies a single entry in a bibliography.
<COLUMN>	Specifies that a new column of output begins.
<DOCUMENT_ATTRIBUTES>	Enables doctype-specific tags that override the default design format of the ARTICLE doctype.
<QUOTATION>	Begins a quotation in which your spacing is retained and text is neither filled (spaces closed up) nor justified (aligned on either margin).
<REF_NOTE>	Specifies the text of a reference note, and creates a bracketed reference number in the article text.
<REF_NOTES>	Causes all accumulated reference notes to be output.
<RUNNING_FEET>	Creates a heading at the bottom of each page.
<RUNNING_TITLE>	Creates a 1- or 2-line heading at the top of each page.
<SOURCE_NOTE>	Specifies the original source of information for an article.
<SUBTITLE>	Specifies a subtitle for an article.
<TITLE>	Specifies the main title line for an article.
<TITLE_SECTION>	Begins the title section of an article. The title spans both columns of the article.
<VITA>	Abstracts the author's professional history.

2.1 ARTICLE Doctype Common Elements

The ARTICLE doctype provides tags that let you perform the following functions:

- Create an article title and subtitle in either a 1- or 2-column format.
- Specify an author (or list of authors) for the article, and provide professional history, address, affiliation, or other author information.
- Create abstracts, source notes, and acknowledgments.
- Specify whether primary headings are to be numbered or unnumbered.
- Create running titles and running feet.
- Create quotations in which spacing and line breaks are retained as entered.

- Create back notes or reference notes that are automatically numbered and collated.
- Create bibliographies.

2.1.1 Titles and Subtitles

Create titles and subtitles for an article using the <TITLE_SECTION>, <TITLE>, and <SUBTITLE> tags.

The <TITLE_SECTION> tag begins a title section that spans both columns of the article and enlarges the type faces output by the <TITLE> and <SUBTITLE> tags. If you do not use the <TITLE_SECTION> tag, the <TITLE> and <SUBTITLE> tags create a title or subtitle restricted to the first column of the article.

Typically, you use only the <TITLE> and <SUBTITLE> tags in the context of the <TITLE_SECTION> tag, but you can use the <AUTHOR> tag instead to have the author's name span both columns.

The following example shows how to use the <TITLE_SECTION> tag to create a title and subtitle that use the full page width of the article:

```
<TITLE_SECTION>
<TITLE>(A Guide to Instrument Care)
<SUBTITLE>(A Professional's View)
<ENDTITLE_SECTION>
```

2.1.2 Author Information

VAX DOCUMENT provides several tags to specify authors and author information in the ARTICLE doctype.

Use the <AUTHOR> and <AUTHOR_LIST> tags to specify one or more authors for an article. Use the following tags to specify information about an author:

- <AUTHOR_ADDR> specifies the address of the author.
- <AUTHOR_AFF> specifies information about the professional or educational affiliations of the author.
- <VITA> specifies the professional history of the author.

These tags appear in your output file in whatever order you place them in your SDML file, with the exception of the <VITA> tag, which places its text argument at the bottom of the current text column of output.

In the following example, a title, subtitle, and author's name and professional history are created using the <TITLE>, <SUBTITLE>, <AUTHOR>, and <VITA> tags, respectively.

The output from this example places the title, subtitle, and author's name at the top of the first column, and places the text specified as the argument to the <VITA> tag at the bottom of the first column.

Using the ARTICLE Doctype

```
<TITLE>(Computer Graphics)
<SUBTITLE>(Everyone Likes It)
<AUTHOR>(G.R. Edwards)
<VITA>(G.R. Edwards has used graphics editors extensively.)
<P>Computer Graphics really are not for everyone, yet...
.
.
<ENDTITLE_SECTION>
```

You can also use the author information tags in the context of the `<TITLE_SECTION>` tag. In the following example, the title, subtitle, and author's name are output using the full page width because these tags are used in the context of the `<TITLE_SECTION>` tag. The information on the author's affiliations, and then the text of the article, are output in a single column because those tags were used outside of the context of the `<TITLE_SECTION>` tag.

```
<TITLE_SECTION>
<TITLE>(Computer Graphics)
<SUBTITLE>(Everyone Likes It)
<AUTHOR>(G.R. Edwards)
<ENDTITLE_SECTION>
<AUTHOR_AFF>(G.R. Edwards is a senior consultant for Terminals, Inc.)
<P>Computer Graphics really are not for everyone, yet...
```

2.1.3 Abstracts, Source Notes, and Acknowledgments

Abstract, source note, and acknowledgment sections are special formats that typically occur at the beginning or end of an article, depending on your preference.

Abstracts

Use the `<ABSTRACT>` tag to specify an abstract for an article. You can specify the `<ABSTRACT>` tag either in the context of the `<TITLE_SECTION>` tag or following it.

In the following example, a short abstract is created in the context of the `<TITLE_SECTION>` tag. This causes the abstract to be formatted using the full page width rather than just a single column. Unlike its use in the previous example, the `<AUTHOR>` tag occurs outside of the context of the `<TITLE_SECTION>` tag and so formats using a single column.

```
<TITLE_SECTION>
<TITLE>(A Guide to Instrument Care)
<ABSTRACT>(A summary of brass and keyboard instrument care fundamentals by
a professional musician.)
<ENDABSTRACT>
<ENDTITLE_SECTION>
<AUTHOR>(Dan Dover)
```

Source Notes

The `<SOURCE_NOTE>` tag lets you specify the origin of material for an article. In the following example, the output from the `<SOURCE_NOTE>` tag prints at the bottom of the current column of output:

```
<SOURCE_NOTE>(From the Boston Globe)
<LINE>(<MCS>(COPYRIGHT) 1986 by the Boston Globe)
```

You can specify source information at the beginning or end of an article.

Acknowledgments

Use the <ACKNOWLEDGMENTS> tag to create any necessary acknowledgments for your article. Enter the text of the acknowledgment as an argument to the <ACKNOWLEDGMENTS> tag.

The following example shows an acknowledgments section created using the <ACKNOWLEDGMENTS> tag. Note how you use it near the end of the SDML file with the <BACK_NOTES> and <REF_NOTES> tags.

```
<REF_NOTES>(Bibliography)
<BACK_NOTES>(References)
<ACKNOWLEDGMENTS>(I am deeply indebted to my doctor for her support in this
task.)
```

2.1.4 Headings

The ARTICLE doctype uses the global numbered heading tags (<HEAD1>, <HEAD2>, and so on). However, by default, these headings are not numbered. Specify numbered headings by using the <DOCUMENT_ATTRIBUTES> tag, as shown in the following example:

```
<DOCUMENT_ATTRIBUTES>
<SET_HEADINGS>(NUMBERED)
<ENDDOCUMENT_ATTRIBUTES>
```

In addition to using them for primary headings, use the global <SUBHEAD1> and <SUBHEAD2> tags to specify unnumbered paragraph topics or side headings, as in the following example:

```
<SUBHEAD1>(Rationale.)
<P>
The purpose of this experiment...
```

2.1.5 Running Titles and Running Feet

Use the <RUNNING_TITLE> and <RUNNING_FEET> tags to place a title at the top or bottom of all the pages of your article.

The <RUNNING_FEET> tag accepts a single text argument, which it uses to create a title at the bottom of the page. The <RUNNING_TITLE> tag accepts one or two text arguments, which it uses to create a 1- or 2-line title at the top of the page.

The following example shows a 2-line running title being set for the top of the page using the <RUNNING_TITLE> tag and a single-line running title being set for the bottom of the page using the <RUNNING_FEET> tag.

```
<RUNNING_TITLE>(Mr. A. Author and\Mrs. B. Author)
<RUNNING_FEET>(The Story of Our Life Together)
```

Using the ARTICLE Doctype

2.1.6 Quotations

You can use either the ARTICLE doctype `<QUOTATION>` tag or the global `<SAMPLE_TEXT>` tag to place extended quotations in an article.

Use the `<QUOTATION>` tag to format text you want to appear exactly as it is entered into the SDML file. The following example shows a Haiku poem formatted using the `<QUOTATION>` tag:

```
<P>
A similar Haiku follows.
<QUOTATION>
  All lights are frozen;
    The cursor box blinks blandly.
      Soon, I see the dump.
<ENDQUOTATION>
```

Use the global `<SAMPLE_TEXT>` tag to create an extended quotation that is to be filled and justified in the text. You must supply any internal punctuation, special spacing, and so on. The following example shows how to use the `<SAMPLE_TEXT>` tag to create an extended quotation:

```
...mankind, as in the following text fragment:
<SAMPLE_TEXT>
<P>
<QUOTE>
Many are the ways of mankind. As some strive for recognition, others seek
obscurity. Surely, we are the strangest of creatures.
<ENDQUOTE>
<ENDSAMPLE_TEXT>
```

2.1.7 Numbered Notes

You can use two types of automatically numbered notes in the ARTICLE doctype: back notes and reference notes. Back notes, sometimes called end notes, are referenced in the text of an article using superscript numbers. Reference notes are similar to back notes, except that the references in the text are output using normal-sized numbers enclosed in brackets.

Note: Footnotes are similar to back notes, except they are placed at the bottom of a column of text. To create footnotes, use the global `<FOOTNOTE>` tag. However, do not use the `<FOOTNOTE>` tag in an article in which you are using back notes. Both the `<FOOTNOTE>` and the `<BACK_NOTE>` tags create superscript numbers for references, and that output would be extremely misleading and confusing.

VAX DOCUMENT accumulates references to each type of note while the article processes, and outputs them at the end of the article. Use only one of these two types of notes in your article.

2.1.7.1 Back Notes

To create a set of notes at the end of an article, use the `<BACK_NOTE>` tag and the `<BACK_NOTES>` tag. Enter the `<BACK_NOTE>` tag in your SDML file wherever you want to have a superscript number in the text to show a note. Enter the text of the note as an argument to the tag. VAX DOCUMENT sequentially numbers each of the back note entries and places the appropriate sequential number as a superscript in the output file.

For example, if you want to cite the book *Training Seagulls* as a back note, and this back note was the third in your document, the text where you cited the book would appear as follows:

These techniques are outlined in *Training Seagulls*³.

Back notes are not automatically output at the end of the article so that you can control their position in the article. Place the `<BACK_NOTES>` tag in your SDML file at the point you want the accumulated back notes to print. When the `<BACK_NOTES>` tag processes, all the accumulated notes print, with their correct numbers, and with the text you specified as arguments to the `<BACK_NOTE>` tags.

The following example shows how to use the `<BACK_NOTE>` tag. The `<BACK_NOTE>` tag would be replaced by a superscript number in the output, and the note produced by that tag would be output near the end of the article using the `<BACK_NOTES>` tag.

```
As Ms. Roma so clearly stated <BACK_NOTE>(P.A. Roma, <QUOTE>(Computer-Chart
Making from the Graphic Editor's Perspective,) <EMPHASIS>(ACM Computer Graphics,
SIGGRAPH '99 Conf. Proc.), Vol 45. No. 3, July 1999, pp. 247-253.)...
<BACK_NOTES>
```

2.1.7.2 Reference Notes

You can create bibliographic reference notes by using the `<REF_NOTE>`, `<REF_NOTES>`, and optionally, the global `<REFERENCE>` tags. Place the `<REF_NOTE>` tag in your SDML file at the point you want the reference to appear. This tag is replaced in the output by a number in brackets, which corresponds to the number assigned to the note text, for example, [4].

Use the `<REF_NOTES>` tag to process the text of the reference notes you have created with assigned numbers. Typically, you place this tag at the end of the SDML file, but you can have the references appear earlier.

To reference a source that you have already referenced using the `<REF_NOTE>` tag, specify the *symbol name* argument to that `<REF_NOTE>` tag and use the global `<REFERENCE>` tag to refer to that symbol.

The following example shows a reference note created using the `<REF_NOTE>` tag, a referral to that note using the global `<REFERENCE>` tag, and the printing of all the accumulated reference notes using the `<REF_NOTES>` tag. Note how the `<REF_NOTE>` tag was coded with the symbol CHICAGO_MAN, so that the subsequent `<REFERENCE>` tag could reference that symbol and use that same reference note number.

Using the ARTICLE Doctype

Sorting entries word by word is preferred <REF_NOTE>(<EMPHASIS>(A Manual of Style,) The University of Chicago Press, 1969.\CHICAGO_MAN)...<P>
Overuse of emphasis can cause confusion <REFERENCE>(CHICAGO_MAN)...<REF_NOTES>(References)

2.1.8 Bibliographies

Use the <BIBLIOGRAPHY> tag to create a bibliography of related reading when you do not use numbered reference notes to reference other works in the text of the article. The <BIBLIOGRAPHY> tag enables the <BIB_ENTRY> tag and lets you specify a heading for the bibliography as an argument to the <BIBLIOGRAPHY> tag.

Create each entry in the bibliography by specifying the entry as an argument to the <BIB_ENTRY> tag. When you use the <BIB_ENTRY> tag, use the <EMPHASIS> and <QUOTE> tags to specify the entry.

The following example shows a bibliography with two entries:

```
<BIBLIOGRAPHY>(Bibliography)
<p>
The following may also be of interest:
<BIB_ENTRY>(<EMPHASIS>(Molecular Connectivity in Chemistry and Drug Research.)
Lamont B. Kier and Lowell H. Hall. Academic Press, 1983.)
<BIB_ENTRY>(Arhnheim, Rudolph, <EMPHASIS>(Visual Thinking). University of
California Press, Berkeley, 1984.)
<ENDBIBLIOGRAPHY>
```

2.2 Improving the Format of a 2-Column Doctype

The ARTICLE doctype creates a 2-column document. Although this doctype lets you visualize what your document will look like when printed, it is somewhat less flexible in terms of how it formats SDML tags than the single-column doctypes. This section summarizes how you can improve the format of your 2-column document.

Note: The REPORT.TWOCOL doctype also outputs a 2-column document, and these techniques work for it also.

2.2.1 Line Breaks in Columns

The width of the text column for paragraphs is much smaller in the 2-column doctype than in the single-column doctypes. Furthermore, the left column is formatted right-justified. As you enter the text for your document into the SDML file, do not be overly concerned about text paragraphs that exceed the right margin during text formatting. The text formatter issues the following message when a text line exceeds the right margin:

```
%TEX-W-LINETOOLONG_P, line too
long ... in paragraph ...
```

As you complete your document, you can use the global <HYPHENATE> and <KEEP> tags to improve line breaks in your printed document.

Use the global `<HYPHENATE>` tag to specify possible points of hyphenation in words the text formatter does not know how to hyphenate, but that you want to allow to hyphenate. This increases the number of places the text formatter can hyphenate the text, and so creates more even line breaks.

Use the global `<KEEP>` tag to specify text that you do not want hyphenated (broken across a line) by the text formatter. Use this tag sparingly, because it decreases the number of places the text formatter can hyphenate the text, making it difficult for the text formatter to create well-placed line breaks.

The text formatter constructs more well-formatted text lines in each column when it has more places to hyphenate words in the text. The more places you allow the text formatter to hyphenate your text, the better your final output formats.

2.2.2 Wide Tables and Examples

When developing examples and tables using a 2-column doctype, be careful of the following conditions:

- The width of tables and figures (if your figures include monospaced examples or art)
- Monospaced or unformatted output created using the `<CODE_EXAMPLE>` or `<QUOTATION>` tags that exceed the column width

If a table, figure, or example is wider than the text column width, use the `WIDE` argument to specify attributes for the tag.

When you specify the `<TABLE_ATTRIBUTES>`, `<FIGURE_ATTRIBUTES>`, or `<EXAMPLE_ATTRIBUTES>` tag with the `WIDE` argument to create a wide table, figure, or example, that table, figure, or example causes the 2-column output to be suspended and the text entered before that table, figure, or example to be placed in the two columns above the table, figure, or example.

The table, figure, or example then outputs using the full page width, as if occurring in a single-column doctype. Two-column formatting is restored after the table, figure, or example ends, and the text after the table, figure, or example begins again in the first column under the table, figure, or example.

A code example, itself, using the `<CODE_EXAMPLE>` and `<ENDCODE_EXAMPLE>` example tags, does not suspend the column output and print the code example across both columns. You must encase the code example in a table, figure, or example.

2.2.3 Final Adjustment of Column and Page Breaks

Using a 2-column doctype, you may need to adjust your paged output when your text is complete. It is sometimes difficult to create balanced pages with the constraints of a 2-column document. Occasionally, you must insert explicit line, column, and page breaks into a 2-column document to improve its appearance.

Using the ARTICLE Doctype

Adjusting Column Breaks

When the text formatter creates a 2-column page, it breaks the text into two columns so as to create a page in which the columns are of as nearly equal length as possible. Certain text elements (such as tables and figures) cannot be easily broken across columns. The text formatter uses vertical space to adjust the length of the columns. Therefore, you may see large amounts of vertical white space preceding and following those text elements that accept a variable amount of white space, such as headings, lists, and tables.

Specify that columns be explicitly broken by using the `<COLUMN>` or `<FINAL_CLEANUP>(COLUMN_BREAK)` tags. Use the `<COLUMN>` tag only when you want the subsequent text to always begin a new column, regardless of any changes you make to the text. Use the `<FINAL_CLEANUP>(COLUMN_BREAK)` tag only after your text is finished and you want to improve the appearance of your document by specifying a new column of text. In either case, if the current text is in the first column of a page, starting a new column places the next text in the second column. If the current text is in the second column of a page, starting a new column results in a new page of output.

In some circumstances, the output of a 2-column page may appear to have lost vertical space before a text element. For example, a heading tag may have no space before it. When this occurs in a 2-column doctype, ignore the occurrence until you are ready to give your document a final revision. If the space is still being lost, use the `<FINAL_CLEANUP>(SPECIAL_BREAK)` tag. For example, suppose the following lines represent fragments of a 2-column page:

```
=====
MAJOR HEADING
=====
=====
=====
=====
=====
=====
shows what happens to

=====
=====
=====
=====
=====
=====
=====
Next Heading
=====
=====
```

In the previous example, the spacing appears to be lost above the heading “Next Heading.” You correct this by placing the `<FINAL_CLEANUP>(SPECIAL_BREAK)` tag in the SDML file between the words that are output on the final line of the first column, as in the following example:

```
<P>
...better place. An example <FINAL_CLEANUP>(SPECIAL_BREAK) shows what
happens to the end of text in this column.
```

You should need to use this special column break only in rare instances.

Adjusting Page Breaks

A new page of output explicitly starts whenever the following conditions exist:

- A `<COLUMN>` or `<FINAL_CLEANUP>(COLUMN_BREAK)` occurs in the right text column and so results in a new page of output.

Using the ARTICLE Doctype

- A `<FINAL_CLEANUP>(PAGE_BREAK)` tag specifies that text start on a new page.

In either of these situations, the current page is set in two columns, without balancing the columns. The length of the text in either column may be less than that of the regular balanced page.

Using the ARTICLE Doctype

2.3 A Sample Use of the ARTICLE Doctype Tags

This section contains a sample input SDML file for an article created using the ARTICLE doctype tags and processes using the ARTICLE doctype design. Figure 2-2 shows the corresponding article output from that SDML file. Comparing these samples may be helpful in understanding how to use these tags to create 2-column articles. Should you wish to create this output yourself, you can obtain file ARTICLE_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

```
<TITLE_SECTION>
<TITLE>(I Have to Care for This Instrument?)
<SUBTITLE>(One of the Young People's Musical Guides)
<ENDTITLE_SECTION>
<RUNNING_TITLE>(Caring for Instruments)
<RUNNING_FEET>(Instrument Care)

<AUTHOR_LIST>(By)
<AUTHOR>(Dan Dover)
<AUTHOR_AFF>(Cleveland Conservatory of Music)
<AUTHOR_ADDR>(Cleveland, Ohio)

<AUTHOR>(Clair Frobisher)
<AUTHOR_AFF>(Toledo Academy of Fine Arts)
<AUTHOR_ADDR>(Toledo, Ohio)

<ENDAUTHOR_LIST>
<ABSTRACT>

Musical instruments of any kind can bring years of enjoyment to the player, and
hopefully to the listener. But the musical instrument must be cared for
properly along the way. This guide discusses basic care of several musical
instruments representative of the major instrument families.
<ENDABSTRACT>
<CHEAD>(Keyboard Instruments)

<P>
The first rule in caring for any keyboard instrument is <EMPHASIS>(Are your
hands clean?) <REF_NOTE>( <EMPHASIS>(Tickling the Ivories: Piano for Beginners),
Architect Press, 1982.). Sticky fingers lead to sticky keys. Also, grime and
dirt will scratch the keys and lodge between them as well.

<P>
Even the natural oils of your hand have a bad effect on the keyboard. It is
always a good idea to wash your hands before playing the piano, organ, or other
keyboard instrument. And after you are through playing, take a warm, damp cloth
and wipe down the keyboard. This removes any residual hand oil from the keys.

<column>
<P>
The second rule for keyboard care is <EMPHASIS>(tuning). Like Mary Edith
Whiteout of the Hanscom Music Company says:

<QUOTATION>
You can tell the quality of pianists by the
pitch of their instrument. A well-tuned piano
is as much a joy, as a badly-tuned piano is
a horror.
<ENDQUOTATION>

<P>
Have your piano tuned every 6 months (for the average piano player); if you
play more than 4 hours a day, we recommend you have it tuned every 3 to
4 months.

<P>
If your organ or your accordion goes out of tune, take it to a repairman and
get the offending note fixed. In summary, basic care for your keyboard
instrument entails:
```

Using the ARTICLE Doctype

```
<LIST>(NUMBERED)
<LE>
Clean hands and a clean instrument; wash your hands before, wash the keyboard
after
<LE>
Tune your instrument regularly; 6 months - average use, 3 to 4 months for heavy
use
<ENDLIST>

<COLUMN>

<CHEAD>(Brass Instruments)
<P>
The first rule in caring for any brass instrument is <EMPHASIS>(Keep your mouth
clean.) Be sure to brush your teeth and rinse your mouth if you are going to
play the trumpet <REF_NOTE><EMPHASIS>(Trumpeter Lullaby: Caring for Your Horn),
County Ecks Press, 1985), trombone <REF_NOTE><EMPHASIS>(Trombone Exercises)
Emerald Books, 1983), or other brass instrument. Food particles left in your
mouth will foul up the valves and slides. They may even restrict the air flow,
make the instrument go out of tune, or even damage it permanently.

<P>
The second rule is <EMPHASIS>(Oil your valves and slides regularly.) Use the
recommended oil for your instrument. This will ensure that things move smoothly
and quickly.

<P>
The third rule is <EMPHASIS>(Polish your instrument after each use) with a warm,
damp cloth. This will help keep it from tarnishing from the natural oils in
your hand. In addition to this, you should use a recommended brass polish every
month. In summary, basic care for your brass instrument entails:

<LIST>(NUMBERED)
<LE>
A clean mouth.
<LE>
Oiled valves and slides.
<LE>
Polishing on a regular basis.
<ENDLIST>

<REF_NOTES>(Additional Reading)
<VITA>(Dan Dover is Toscanini Professor of Music at the Cleveland
Conservatory of Music. He publishes the annual Musician's Guide to
Symphonic Opportunities.)
<VITA>(Clair Frobisher is the Director of the Toledo Academy of Fine
Arts. Recently, she instituted the acclaimed Young People's
Symphonies.)
<ACKNOWLEDGMENTS>(The authors are indebted to the Toscanini
Foundation for support in this series of guides.)
```

Figure 2-2 and Figure 2-3 show the corresponding article output from that SDML file. Comparing these samples may be helpful in understanding how to use these tags. Should you wish to create this output yourself, you can obtain file ARTICLE_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Using the ARTICLE Doctype

Figure 2-2 ARTICLE Doctype Output Example, Page 1

I Have to Care for This Instrument?

One of the Young People's Musical Guides

By

Dan Dover

Cleveland Conservatory of Music

Cleveland, Ohio

Clair Frobisher

Toledo Academy of Fine Arts

Toledo, Ohio

Have your piano tuned every 6 months (for the average piano player); if you play more than 4 hours a day, we recommend you have it tuned every 3 to 4 months.

If your organ or your accordion goes out of tune, take it to a repairman and get the offending note fixed. In summary, basic care for your keyboard instrument entails:

1. Clean hands and a clean instrument; wash your hands before, wash the keyboard after
2. Tune your instrument regularly; 6 months - average use, 3 to 4 months for heavy use

Musical instruments of any kind can bring years of enjoyment to the player, and hopefully to the listener. But the musical instrument must be cared for properly along the way. This guide discusses basic care of several musical instruments representative of the major instrument families.

Keyboard Instruments

The first rule in caring for any keyboard instrument is *Are your hands clean?* [1]. Sticky fingers lead to sticky keys. Also, grime and dirt will scratch the keys and lodge between them as well.

Even the natural oils of your hand have a bad effect on the keyboard. It is always a good idea to wash your hands before playing the piano, organ, or other keyboard instrument. And after you are through playing, take a warm, damp cloth and wipe down the keyboard. This removes any residual hand oil from the keys.

The second rule for keyboard care is *tuning*. Like Mary Edith Whiteout of the Hanscom Music Company says:

You can tell the quality of pianists by the pitch of their instrument. A well-tuned piano is as much a joy, as a badly-tuned piano is a horror.

Figure 2-3 ARTICLE Doctype Output Example, Page 2

Caring for Instruments

Brass Instruments

The first rule in caring for any brass instrument is *Keep your mouth clean*. Be sure to brush your teeth and rinse your mouth if you are going to play the trumpet [2], trombone [3], or other brass instrument. Food particles left in your mouth will foul up the valves and slides. They may even restrict the air flow, make the instrument go out of tune, or even damage it permanently.

The second rule is *Oil your valves and slides regularly*. Use the recommended oil for your instrument. This will ensure that things move smoothly and quickly.

The third rule is *Polish your instrument after each use* with a warm, damp cloth. This will help keep it from tarnishing from the natural oils in your hand. In addition to this, you should use a recommended brass polish every month. In summary, basic care for your brass instrument entails:

1. A clean mouth.

2. Oiled valves and slides.

3. Polishing on a regular basis.

Additional Reading

- [1] *Tickling the Ivories: Piano for Beginners*, Architect Press, 1982.
- [2] *Trumpeter Lullaby: Caring for Your Horn*, County Ecks Press, 1985
- [3] *Trombone Exercises* Emerald Books, 1983

Acknowledgments

The authors are indebted to the Toscanini Foundation for support in this series of guides.

Dan Dover is Toscanini Professor of Music at the Cleveland Conservatory of Music. He publishes the annual *Musician's Guide to Symphonic Opportunities*.

Clair Frobisher is the Director of the Toledo Academy of Fine Arts. Recently, she instituted the acclaimed Young People's Symphonies.

2.4 ARTICLE Doctype Tag Reference

This part of Chapter 2 provides reference information on all the tags specific to the ARTICLE doctype.

<ABSTRACT>

Creates an article abstract and can also specify a heading for that abstract.

SYNTAX <ABSTRACT>[(*abstract heading*)]

ARGUMENTS *abstract heading*
This is an optional argument. It specifies a heading for the abstract. If you do not specify a heading, none is output.

related tags • <TITLE_SECTION>

required terminator <ENDABSTRACT>

DESCRIPTION The <ABSTRACT> tag creates an article abstract and can also specify a heading for that abstract.

This tag performs the same function as the global <ABSTRACT> tag, but in a different context.

EXAMPLE The following example shows how to use the <ABSTRACT> tag.

```
<ABSTRACT>
This article presents strong arguments for industries to establish new
programs to educate illiterate employees.
<ENDABSTRACT>
```

ARTICLE Doctype Tag Reference

<ACKNOWLEDGMENTS>

<ACKNOWLEDGMENTS>

Creates an acknowledgments section in an article.

SYNTAX **<ACKNOWLEDGMENTS>** (*acknowledgment text*)

ARGUMENTS ***acknowledgment text***
Specifies the text of one or more acknowledgments.

related tags

- <BACK_NOTES>
- <REF_NOTES>

DESCRIPTION The <ACKNOWLEDGMENTS> tag creates an acknowledgments section in an article. It outputs the heading Acknowledgments for this section. Enter the text of the acknowledgment as an argument to the <ACKNOWLEDGMENTS> tag.

Typically, enter the <ACKNOWLEDGMENTS>, <BACK_NOTES>, and <REF_NOTES> tags at the end of an SDML file.

EXAMPLE The following example shows how to use the <ACKNOWLEDGMENTS> tag. Note how it is used near the end of the SDML file with the <BACK_NOTES> and <REF_NOTES> tags.

```
<REF_NOTES>(Bibliography)
<BACK_NOTES>(References)
<ACKNOWLEDGMENTS>(I am deeply indebted to my doctor for her support in this
task.)
```

<AUTHOR>

Specifies an author of an article.

SYNTAX <AUTHOR>(author name[\ optional information])

ARGUMENTS *author name*

Specifies the name of the author. To include the word By with the author's name, specify it as part of the author name text.

optional information

This is an optional argument. It specifies additional optional information about the author. This information formats below the author's name. This text should be approximately one line long.

related tags

- <AUTHOR_ADDR>
- <AUTHOR_AFF>
- <AUTHOR_LIST>
- <SOURCE_NOTE>
- <VITA>

DESCRIPTION

The <AUTHOR> tag species the author of an article. Enter the name of the author as the first argument to the <AUTHOR> tag; specify additional information about the author as the second argument to the <AUTHOR> tag.

EXAMPLE

The following example shows how to use the <AUTHOR> tag.

```
<AUTHOR>(By A.B. Roma\Publisher)
<AUTHOR_AFF>(emphasis>(Disco Monthly) staff)
<AUTHOR_ADDR>(Top-Ten Corporation,\ 5300 Westlake Boulevard,\ Los Angeles,
California 09945)
```

<AUTHOR_AFF>

Specifies information about the organizational affiliation of the author.

SYNTAX <AUTHOR_AFF> (*affiliation information*)

ARGUMENTS *affiliation information*
Specifies information about the affiliation of the author.

related tags

- <AUTHOR>
- <AUTHOR_ADDR>
- <AUTHOR_LIST>
- <VITA>

DESCRIPTION The <AUTHOR_AFF> tag specifies information about the organizational affiliation of the author. Typically, enter this tag after the <AUTHOR> tag in the SDML file.

EXAMPLE The following example shows how to use the <AUTHOR> tag.

```
<AUTHOR>(A.B. Roma)
<AUTHOR_AFF>(<emphasis>(Disco Monthly) staff)
<AUTHOR_ADDR>(Top-Ten Corporation,\ 5300 Westlake Boulevard,\ Los Angeles,
California 09945)
```

ARTICLE Doctype Tag Reference

<AUTHOR_LIST>

<AUTHOR_LIST>

Creates a list of authors for an article with multiple authors.

SYNTAX

<AUTHOR_LIST>*[(heading text)]*

ARGUMENTS

heading text

This is an optional argument. It provides an introductory heading for a list of authors. A sample heading might be By:.

related tags

- <AUTHOR>
 - <AUTHOR_ADDR>
 - <AUTHOR_AFF>
 - <VITA>
-

required terminator

<ENDAUTHOR_LIST>

DESCRIPTION

The <AUTHOR_LIST> tag creates a list of authors for an article with multiple authors. Optionally, specify a heading for the list of authors as an argument to the <AUTHOR_LIST> tag. Enter the name of each author as the first argument to the <AUTHOR> tag, and specify additional information about the author as the second argument to the <AUTHOR> tag.

EXAMPLE

For an example showing a list of two authors introduced by the word By, refer to Figure 2-2, Figure 2-3, and the SDML file that produced this output.

<BACK_NOTE>

Creates a back note entry and a superscript reference number to that entry in the article text.

SYNTAX <BACK_NOTE> (*back note text*)

ARGUMENTS *back note text*
Specifies the text to be associated with the back note entry.

related tags

- <BACK_NOTES>
- <REF_NOTE>
- The global <FOOTNOTE> tag

restrictions Do not use the <BACK_NOTE> tag in the same document as the global <FOOTNOTE> tag. Both tags would place superscript numbers into a document, and references to those numbers would be confusing.

DESCRIPTION The <BACK_NOTE> tag creates a back note entry and a superscript reference number to that entry in the article text. A back note can also be referred to as an end note in your document. Enter the <BACK_NOTE> tag in your SDML file wherever you want to provide a back note citation. VAX DOCUMENT sequentially numbers each of the back note entries and places the appropriate sequential number as a superscript in the output file.

For example, if you cite the book *Training Seagulls* as a back note, and this back note is the third in your document, the text where you cited the book would appear as follows:

These techniques are outlined in *Training Seagulls*³.

VAX DOCUMENT collects all the automatically numbered back note entries and outputs them together in their sequential order wherever you use the <BACK_NOTES> tag. Typically, you would want to use the <BACK_NOTES> tag at the end of your article.

EXAMPLE The following example shows a reference to a back note and the text of the note as it is to appear at the end of the document. The <BACK_NOTES> tag at the conclusion of the article causes this note, and any others, to be output.

ARTICLE Doctype Tag Reference

<BACK_NOTE>

As Ms. Roma so clearly stated, <BACK_NOTE> (P.A. Roma,
<QUOTE> (Computer-Chart Making from the Graphic Editor's Perspective,)
<EMPHASIS> (ACM Computer Graphics, SIGGRAPH '99 Conf. Proc.), Vol 45.
No. 3, July 1999, pp. 247-253.)...

<BACK_NOTES>

<BACK_NOTES>

Outputs all the back notes created with the <BACK_NOTE> tag at the place in the file where you use the <BACK_NOTES> tag.

SYNTAX <BACK_NOTES>[(*heading text*)]

ARGUMENTS *heading text*

This is an optional argument. It specifies the text to be output above the back note section. This heading has the default heading format of the <HEAD1> tags in a document using unnumbered heads. If you do not specify the *heading text* argument, no heading is output.

related tags

- <BACK_NOTE>

DESCRIPTION

The <BACK_NOTES> tag outputs all the back notes created with the <BACK_NOTE> tag at the place in the file where you use the <BACK_NOTES> tag.

You can specify a heading for these back notes as an argument to the <BACK_NOTES> tag. Alternatively, you can specify your own heading with the heading tag (<HEAD1>, <HEAD2>, and so on) that is appropriate for your document. The heading level tag must precede the <BACK_NOTES> tag, as shown in the second following example.

EXAMPLES

The following example shows how you can create the heading References for a list of back notes by coding that heading as an argument to the <BACK_NOTES> tag.

1 <BACK_NOTES>(References)

The following example shows how you can create the second-level heading Articles for a list of back notes by coding that heading as an argument to the <HEAD2> tag, and placing that tag immediately before the <BACK_NOTES> tag.

2 <HEAD2>(Articles\24_Articles)
 <BACK_NOTES>

ARTICLE Doctype Tag Reference

<BIBLIOGRAPHY>

<BIBLIOGRAPHY>

Begins a bibliography.

SYNTAX **<BIBLIOGRAPHY>**[(*heading text*)]

ARGUMENTS *heading text*
This is an optional argument. It specifies the heading for a bibliography. This heading appears in the format used by <HEAD1> tags in an article with unnumbered heads. By default, no heading outputs.

related tags

- <BIB_ENTRY>
- <REF_NOTE>
- <REF_NOTES>

required terminator <ENDBIBLIOGRAPHY>

DESCRIPTION The <BIBLIOGRAPHY> tag begins a bibliography. Create a bibliography of related reading when you do not use numbered reference notes to reference other works in the text of the article. Create each entry in the bibliography using the <BIB_ENTRY> tag. Specify a heading for the bibliography either as an argument to the <BIBLIOGRAPHY> tag or by using an unnumbered heading tag immediately before the <BIBLIOGRAPHY> tag.

To create numbered reference notes, use the <REF_NOTE> and <REF_NOTES> tags instead of <BIB_ENTRY> tags.

EXAMPLE The following example shows how to use the <BIBLIOGRAPHY> tag.

```
<BIBLIOGRAPHY>(Bibliography)
<BIB_ENTRY>(<EMPHASIS>(Molecular Connectivity in Chemistry and Drug Research.)
Lamont B. Kier and Lowell H. Hall. Academic Press, 1983.)
<BIB_ENTRY>(Arhnheim, Rudolph, <EMPHASIS>(Visual Thinking). University of
California Press, Berkeley, 1984.)
<ENDBIBLIOGRAPHY>
```

<BIB_ENTRY>

Specifies a single entry in a bibliography.

SYNTAX <BIB_ENTRY>(bibliography text)

ARGUMENTS ***bibliography text***
Specifies the text of the bibliographic entry.

related tags • <BIBLIOGRAPHY>

restrictions Valid only in the context of a <BIBLIOGRAPHY> tag.

DESCRIPTION The <BIB_ENTRY> tag specifies a single entry in a bibliography. The text of the bibliographic entry is passed as an argument to the <BIB_ENTRY> tag. This text can be of any length.

EXAMPLE For an example of how to use a <BIB_ENTRY> tag, see the example in the <BIBLIOGRAPHY> tag description.

ARTICLE Doctype Tag Reference

<COLUMN>

<COLUMN>

Specifies that output begins in a new column in a 2-column format.

SYNTAX

<COLUMN>

ARGUMENTS

None.

related tags

- The global <FINAL_CLEANUP> tag
-

restrictions

Valid only in a 2-column doctype such as ARTICLE or REPORT.TWOCOL.

DESCRIPTION

The <COLUMN> tag specifies that output begins in a new column in a 2-column format. If this tag occurs in the left text column, the text immediately following it begins in the right text column. If this tag occurs in the right text column, the text immediately following it begins in the left column of the next page.

Use the <COLUMN> tag when you always want to begin a new column at that point in your text. You can use the COLUMN_BREAK argument to the global <FINAL_CLEANUP> tag to also specify a column break; however, use this only during the final processing of the 2-column document.

See Section 2.2 for more information on improving the formatting of a 2-column doctype such as ARTICLE or REPORT.TWOCOL.

EXAMPLE

The following example shows how to use the <COLUMN> tag. In this example, the writer wants the two instrument descriptions to appear side by side, one in each column.

```
<SUBHEAD1>(Woodwind Instruments)
<P>
Woodwind instruments have the following attributes:
<LIST>(UNNUMBERED)
<LE>
They are often made of wood, hence their name.
<LE>
Musicians create sound using these instruments by causing a reed to vibrate...
<ENDLIST>
<COLUMN>
```

ARTICLE Doctype Tag Reference <COLUMN>

```
<SUBHEAD1>(Brass Instruments)
<P>
Brass instruments have the following attributes:
<LIST>(UNNUMBERED)
<LE>
They are often made of brass, hence their name.
<LE>
Musicians create sound using these instruments by vibrating (buzzing) their lips
into a steel mouthpiece...
<ENDLIST>
```

ARTICLE Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

<DOCUMENT_ATTRIBUTES>

Enables doctype-specific tags that override the default design format of the ARTICLE doctype.

SYNTAX

<DOCUMENT_ATTRIBUTES>

ARGUMENTS

None.

required terminator

<ENDDOCUMENT_ATTRIBUTES>

DESCRIPTION

The <DOCUMENT_ATTRIBUTES> tag enables doctype-specific tags that override the default design format of the ARTICLE doctype. You can use the <DOCUMENT_ATTRIBUTES> tag in three doctypes:

- ARTICLE
- REPORT
- SOFTWARE

The <DOCUMENT_ATTRIBUTES> tag enables a group of tags in each of these doctypes that let you modify the default format of that doctype. These tags are recognized only in the context of the <DOCUMENT_ATTRIBUTES> tag. If other VAX DOCUMENT tags occur in this context, they are ignored, as if they had occurred in the context of a <COMMENT> tag.

Typically, use the <DOCUMENT_ATTRIBUTES> tag at the beginning of an input file (or in a file processed using the DCL /INCLUDE qualifier) to alter the default format of a doctype for the processing of that entire file.

Table 2-3 summarizes the formatting tags enabled by the <DOCUMENT_ATTRIBUTES> tag in each of the three supported doctypes.

Table 2-3 Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag

Formatting Tags	Description
<SET_HEADINGS>(UNNUMBERED) <SET_HEADINGS>(NUMBERED)	The <SET_HEADINGS> tag specifies whether numbered or unnumbered headings are produced by the heading-level tags (<HEAD1>, <HEAD2>, and so on). By default, headings are not numbered in a document processed using the ARTICLE doctype. Use the <SET_HEADINGS>(NUMBERED) tag to specify that your headings are to be numbered.

ARTICLE Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

EXAMPLE

The following example shows how to use the <DOCUMENT_ATTRIBUTES> tag to enable a doctype-specific tag that overrides the default design format of the ARTICLE doctype.

```
<DOCUMENT_ATTRIBUTES>  
<SET_HEADINGS> (NUMBERED)  
<ENDDOCUMENT_ATTRIBUTES>
```

ARTICLE Doctype Tag Reference

<QUOTATION>

<QUOTATION>

Begins a quotation in which the spacing is retained and the text is not filled or justified.

SYNTAX

<QUOTATION>

ARGUMENTS

None.

related tags

- The global <CODE_EXAMPLE> tag
 - The global <SAMPLE_TEXT> tag
-

required terminator

<ENDQUOTATION>

DESCRIPTION

The <QUOTATION> tag begins a quotation in which the spacing is retained and the text is not filled or justified.

The <QUOTATION> tag differs from the global <CODE_EXAMPLE> tag in that the text of the quotation is not set in a monospaced font. It lets you quote poetry or passages of text that you do not want formatted.

For long, block-style quotations, use the global <SAMPLE_TEXT> tag.

EXAMPLE

The following example shows how to use the <QUOTATION> tag.

```
It is often instructive to remember the words of our founder:
<QUOTATION>
It is better to try again than to fail;
And better still ...
    to succeed.
<ENDQUOTATION>
```

<REF_NOTE>

Specifies the text of a reference note and creates a bracketed reference number in the article text.

SYNTAX <REF_NOTE>(text of note[\ symbol name])

ARGUMENTS *text of note*
Specifies the text of the reference note.

symbol name
This is an optional argument. It specifies the name of the symbol used in all references to this heading.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <BACK_NOTE>
- <BIBLIOGRAPHY>
- <REF_NOTES>
- The global <FOOTNOTE> tag
- The global <REFERENCE> tag

DESCRIPTION The <REF_NOTE> tag specifies the text of a reference note and creates a bracketed reference number in the article text. Place the <REF_NOTE> tag in the SDML file at the point at which you are referencing text. This tag is replaced in the output file by a number in brackets that corresponds to the number assigned to the note text for example, [4].

Use the <REF_NOTES> tag to cause the text of this note and any other reference notes you have created using the <REF_NOTE> tag to be output. You generally place the <REF_NOTES> tag at the end of the SDML file.

To reference a source that you have already referenced using the <REF_NOTE> tag, specify the *symbol name* argument to that <REF_NOTE> tag and use the global <REFERENCE> tag to refer to that symbol.

EXAMPLE The following example shows how to use the <REF_NOTE> tag both for a single reference and for two references to the same source. Note how the second reference to the Hopkins and Johnson article is made using the <REFERENCE> tag.

ARTICLE Doctype Tag Reference

<REF_NOTE>

These notes and any others would be output by the <REF_NOTES> tag that occurs at the end of the article.

Hopkins and Johnson (1968) <REF_NOTE>(A.A Hopkins and B.B. Johnson, <QUOTE>(An Eye for an Eye,) Proceedings of the American Journal of Comparative Biology, 163:1145-1152.\EYE_ARTICLE) noted the preponderance of short cones in type A subjects. The research of J.Dobbs (1972) <REF_NOTE>(J. Dobbs, <quote>(Cones in Type A and B Subjects), Proceedings of the American Journal of Comparative Biology, 167:201-227.) corroborated this observation.

<P>

In 1978, the research of Hopkins and Johnson (1968) <REFERENCE>(EYE_ARTICLE), was shown to have been misleading... <REF_NOTES>

<REF_NOTES>

Outputs all the reference notes created with the <REF_NOTE> tag at the place in the file where you use the <REF_NOTES> tag.

SYNTAX <REF_NOTES>[(*heading text*)]

ARGUMENTS *heading text*

This is an optional argument. It specifies a heading for the reference notes.

If you do not specify the *heading text* argument, no heading is output. You can specify your own heading with the heading tag (<HEAD1>, <HEAD2>, and so on) that is appropriate to your document.

related tags

- <ACKNOWLEDGMENTS>
- <BACK_NOTES>
- <REF_NOTES>

DESCRIPTION

The <REF_NOTES> tag outputs all the reference notes created with the <REF_NOTE> tag at the place in the file where you use the <REF_NOTES> tag. These references are numbered and correspond to the number placed in your document by the <REF_NOTE> tag (for example, [4]). Typically, place the <REF_NOTES> tag at the end of the SDML file so that the accumulated references appear at the end of the article.

EXAMPLES

The following example shows how to create a list of headings with the heading "References".

1 <REF_NOTES>(References)

The following example shows how to use a heading tag as an alternative heading for the list of references. In this example, the <HEAD2> tag was used.

2 <HEAD2>(References\25_References)
 <REF_NOTES>

ARTICLE Doctype Tag Reference

<RUNNING_FEET>

<RUNNING_FEET>

Creates a single-line heading at the bottom of each page.

SYNTAX

<RUNNING_FEET>(*footer text*)

ARGUMENTS

footer text

Specifies the text to be used as a running heading at the foot of the page.

related tags

- <RUNNING_TITLE>
-

DESCRIPTION

The <RUNNING_FEET> tag creates a single-line heading at the bottom of each page. This heading is called a **footer** because it appears at the foot of the page. When the same footer is used for several pages, the footers are collectively called running feet.

This tag accepts one argument that is the text heading that appears at the bottom of the page. This text is output exactly as entered, including spacing and capitalization.

EXAMPLE

The following example shows how to use the <RUNNING_FEET> tag to place the heading Getting the Piece of Paper at the bottom of each page. The running footer outputs exactly as entered.

```
<RUNNING_FEET>(Getting the Piece of Paper)
<HEAD2>(Getting the Piece of Paper\26_GettingthePieceofPaper)
<P>
```

You can buy clean paper in most major supermarkets, department stores, and hardware stores. You should try to get ruled paper so that your letter will be neat and easy to read.

<RUNNING_TITLE>

Creates a 1- or 2-line running title at the top of each page.

SYNTAX

<RUNNING_TITLE>({ *OFF*
title-1 [\ *title-2*]
[\ *FIRST_PAGE*] })

ARGUMENTS

OFF

This is an optional keyword argument. It specifies that any existing running titles created using the <RUNNING_TITLE> tag are disabled for the page on which this tag occurs and on any subsequent pages.

title-1

This specifies the text of a running title. If you specify a 2-line title, this title outputs on the upper title line.

title-2

This is an optional argument. It specifies the bottom line of a running title that has two lines.

FIRST_PAGE

This is an optional keyword argument. It specifies that the running title is to begin output on the first output page. If you do not specify this keyword, the running title outputs on the page after the current page.

related tags

- <RUNNING_FEET>

DESCRIPTION

The <RUNNING_TITLE> tag creates a 1- or 2-line title at the top of each page. Use the *FIRST_PAGE* argument to the <RUNNING_TITLE> tag to begin the title lines on the first page of output, rather than on the page after the current page as is the default.

Use the *OFF* argument to disable any existing running titles created using the <RUNNING_TITLE> tag. These titles are then disabled for the page on which this tag occurs and on any subsequent pages.

ARTICLE Doctype Tag Reference

<RUNNING_TITLE>

EXAMPLES

The following example shows how to use the <RUNNING_TITLE> tag to create the 2-line running title An E. B. Bartz Course: and Writing Quality Correspondence. Note that because you use the FIRST_PAGE argument, the 2-line running title appears at the top of the first page.

1 <RUNNING_TITLE>(An E. B. Bartz Course:\Writing Quality Correspondence\FIRST_PAGE)
<HEAD>(How to Write a Letter\27_HowtoWriteaLetter)
<P>
The first thing that you should do in writing a letter is to get a clean piece of paper and a well-sharpened pencil.

The following example shows how to disable a running title by using the OFF argument to the <RUNNING_TITLE> tag.

2 <RUNNING_TITLE>(OFF)
<HEAD>(An Example of a Letter\28_AnExampleofaLetter)...

<SOURCE_NOTE>

Provides information pertaining to the original source of information for an article.

SYNTAX <SOURCE_NOTE>(source text)

ARGUMENTS *source text*
Specifies the text that describes the source of the article. The text can be any length and can include any tags not listed in the restrictions section. The source text is positioned at the bottom of the column of output in which the tag is specified.

related tags

- <AUTHOR>
- <VITA>

restrictions Do not use the following tags as part of the source note text: <CODE_EXAMPLE>, <EXAMPLE>, <FIGURE>, <FORMAT>, <HEAD1> through <HEAD6>, <INTERACTIVE>, <MATH>, or <NOTE>.

DESCRIPTION The <SOURCE_NOTE> tag provides information pertaining to the original source of information for an article. Typically, place information provided in the <SOURCE_NOTE> tag either at the beginning of the first column on the first page of an article, or at the end of the last column on the last page. Place the <SOURCE_NOTE> tag in your SDML file to correspond to where you want the output to appear.

If you want the text to appear on the first page, specify the tag following the <AUTHOR> tag. If you want the text to appear on the last page, specify the tag at the end of the SDML file.

EXAMPLE The following example shows how to create a note describing the original source of an article.

<SOURCE_NOTE>(Reprinted from <EMPHASIS>(Visible Discs,) Volume V, Number 3, Summer 1971. c/o The Top-Ten Museum of Art, Cleveland, Ohio, U.S.A., 44106

ARTICLE Doctype Tag Reference

<SUBTITLE>

<SUBTITLE>

Specifies a subtitle for an article.

SYNTAX

<SUBTITLE> (*title line-1*[\ *title line-2*[\ *title line-3*]])

ARGUMENTS

title line-n

Specifies up to three lines of text for a subtitle of an article. The text of each argument centers on a new line of output.

related tags

- <TITLE>
- <TITLE_SECTION>

DESCRIPTION

The <SUBTITLE> tag specifies a subtitle for an article. This subordinate article title can have up to three separate lines. Each of the subtitle lines centers in the column in which the tag occurs (typically the first column). Use the <TITLE> tag to create a main title for an article.

If you want the subtitle (or title) to span both columns of the article, use the <TITLE_SECTION> tag in your SDML file before the <SUBTITLE> (or <TITLE>) tag.

EXAMPLE

The following example shows how to code a main title followed by a subtitle that spans both columns. Note that the <AUTHOR> tag occurs outside of the context of the <TITLE_SECTION> tag, so the name of the author does not span both columns in the output.

```
<TITLE_SECTION>
<TITLE>(FILE PROCESSING)
<SUBTITLE>(CONCEPTS AND INSTRUCTIONS)
<ENDTITLE_SECTION>
<AUTHOR>(A.B. Roma\Contributing Editor)
```

<TITLE>

Specifies the main title line for an article.

SYNTAX

<TITLE> (*title line-1* [*\ title line-2* [*\ title line-3*]])

ARGUMENTS

title line-n

Specifies up to three lines of text for the title of the article. The text of each argument centers on a new line of output.

related tags

- <SUBTITLE>
- <TITLE_SECTION>

DESCRIPTION

The <TITLE> tag specifies the main title line for an article. This title can have up to three separate lines. Each of the title lines is centered in the column in which the tag occurs (typically the first column). Use the <SUBTITLE> tag to create a subordinate title for an article.

If you want the title (or subtitle) to span both columns of the article, use the <TITLE_SECTION> tag in your SDML file before the <TITLE> (or <SUBTITLE>) tag.

EXAMPLE

The following example shows how to code a main title followed by a subtitle that spans both columns. Note that the <AUTHOR> tag occurs outside of the context of the <TITLE_SECTION> tag, so the name of the author does not span both columns in the output.

```
<TITLE_SECTION>  
<TITLE>(FILE PROCESSING\USING THE CALL INTERFACE)  
<SUBTITLE>(CONCEPTS AND INSTRUCTIONS)  
<ENDTITLE_SECTION>  
<AUTHOR>(A.B. Roma\Contributing Editor)
```

ARTICLE Doctype Tag Reference

<TITLE_SECTION>

<TITLE_SECTION>

Begins the title section of an article. The title spans both columns of the article.

SYNTAX

<TITLE_SECTION>

ARGUMENTS

None.

related tags

- <SUBTITLE>
 - <TITLE>
-

required terminator

<ENDTITLE_SECTION>

DESCRIPTION

The <TITLE_SECTION> tag begins the title section of an article. The title spans both columns of the article. Use this tag to create a section at the beginning of an article whenever you want a title, subtitle, or other information to span the full page. If you use the <TITLE> or <SUBTITLE> tag in the context of the <TITLE_SECTION> tag, the typeface output by those tags will be larger than the typeface output outside of the context of the <TITLE_SECTION> tag.

If you do not use the <TITLE_SECTION> tag, any titles, subtitles, or additional information outputs in the appropriate column and does not span the full page.

EXAMPLES

In the following example, the <TITLE> and <SUBTITLE> tags are specified in the context of the <TITLE_SECTION> tag. The <AUTHOR> tag appears after the <ENDTITLE_SECTION>, so it will be formatted in the first column of the article.

```
1 <TITLE_SECTION>
  <TITLE>(Optical Discs)
  <SUBTITLE>(The New Documentation Frontier)
  <ENDTITLE_SECTION>
  <AUTHOR>(A.B. Roma)
```

In the following example, the <TITLE> and <SUBTITLE> tags are used without the <TITLE_SECTION> tag so the title and subtitle text will be set in the first column with the author information.

```
2 <TITLE>(Optical Discs)
  <SUBTITLE>(The New Documentation Frontier)
  <AUTHOR>(A. B. Roma)
```

<VITA>

Provides information about the author's professional history.

SYNTAX <VITA>(vita text)

ARGUMENTS *vita text*
Specifies information describing the professional history of the author. The text can be any length and can include any tags that are not listed in the restrictions section.

related tags

- <AUTHOR>
- <AUTHOR_ADDR>
- <AUTHOR_AFF>
- <AUTHOR_LIST>
- <SOURCE_NOTE>

restrictions Do not use the following tags as part of the vita text: <CODE_EXAMPLE>, <EXAMPLE>, <FIGURE>, <FORMAT>, <HEAD1> through <HEAD6>, <INTERACTIVE>, <MATH>, or <NOTE>.

DESCRIPTION The <VITA> tag provides information about the author's professional history. Typically, place information provided in the <VITA> tag either at the beginning of the first column of the first page of an article, or at the end of the last column on the last page. Place the <VITA> tag in your SDML file to correspond to where you want the output to appear.

To make the text appear on the first page, specify the <VITA> tag following the <AUTHOR> tag. To make the text appear on the last page, specify the <VITA> tag at the end of the SDML file.

EXAMPLE The following example shows how to use the <VITA> tag. The descriptive text is positioned at the bottom of the first column of the first page of the article.

```
<AUTHOR>(A.B. Roma)
<VITA>(A.B. Roma is program director for information processing and distribution
for the Top-Ten Corporation. She has published numerous magazine articles.)
```


3

Using the HELP Doctype

The HELP doctype lets you create VMS Help files from VAX DOCUMENT source files. It works with files created explicitly to be used as source material for Help, and also with the files used to create printed books or Bookreader documents.

3.1 Creating a HELP File

You create a Help file by coding an SDML file and then processing that file with VAX DOCUMENT, using the following command:

```
$ DOCUMENT filename[.SDML] HELP HLP
```

Notice the destination HLP in this command line. Your output in this case is not a printed document, but a file with the extension .HLP. This VMS Help file is used by the VMS Help Librarian to create the Help information that ultimately is read online.

Using the DOCUMENT command for the HELP doctype is similar to using it for other doctypes. The command accepts many of the same qualifiers and parameters. However, with the HELP doctype, the command ignores the /CONTENTS and /INDEX qualifiers. If you use a symbol for DOCUMENT that involves these qualifiers, you may receive a message to the effect that they are being ignored. However, VAX DOCUMENT still processes the file.

You cannot nest Help tags in the SDML file.

In general, you should keep the text in your Help files simple. Lists or other unusual formatting within headings may cause unpredictable output. If you think this may apply to your file, proofread the Help file before patching it into the system.

3.1.1 How HELP Interprets Reference Sections

In addition to processing standard text, HELP interprets reference sections (Command, Routine, Statement, and Tag) and generates the appropriate Help output where possible:

- The <STATEMENT> and <ROUTINE> tags are translated into level-1 Help by default.
- The <COMMAND> and <SUBCOMMAND> tags are translated into multiple-level Help, starting with level 1. A command name will not generate a Help topic if it has been previously defined in the same document. This check is sensitive to the current Help level, which is set by default or by the use of the <SET_HELP_LEVEL> tag.

Using the HELP Doctype

- Commands with permanently attached qualifiers, such as `DEFINE /KEY`, are translated into multiple-level Help, starting with level 1. Thus, using `DEFINE/KEY` specifies `DEFINE` at level 1 and `/KEY` at level 2.
- The Description, Qualifier, Parameter, Arguments, and other sections for each reference item are translated into appropriate, related, lower-level Help. For example, if processing the `<COMMAND>` tag results in two levels of help, then the related parameters are assigned to level 3. (The Format section is treated as part of the text and has no effect on Help levels.)
- Parameters and qualifiers within definition lists are translated into unnumbered Help at the appropriate level.

You can always modify a Help level using the `<SET_HELP_LEVEL>` tag.

3.1.2 How HELP Interprets the Input File

The HELP doctype is based on the SOFTWARE doctype. Any file that processes correctly through the SOFTWARE doctype family should process correctly for HELP.

Just as the command you use to create a Help file ignores certain arguments and qualifiers, the HELP doctype ignores certain material within a source file. The following tags (and all material within them) are automatically ignored for Help output:

```
<CONTENTS_FILE>
<FOOTNOTE>
<FOOTREF>
<FRONTMATTER>-<ENDFRONT_MATTER>
<INDEX_FILE>
<PART_PAGE>-<ENDPART_PAGE>
```

Any of these tags within an area of a file specified for Help generates an informational `INVINHELP` (INValid IN HELP) message. Avoid these messages by conditionalizing your files according to Section 3.1.3.

Ordinarily, you may use the following tags to affect formatting of your text and examples, figures, and tables:

```
<EXAMPLE_ATTRIBUTES>
<EXAMPLE_SPACE>
<FIGURE_ATTRIBUTES>
<FIGURE_SPACE>
<TABLE_ATTRIBUTES>
```

Avoid including these tags in material you want in Help. Otherwise, unexpected spacing may appear in the final Help material.

For the remaining portions of the source file, the HELP doctype processes the text as usual. Text is indented and headings (such as `<HEAD1>` and `<HEAD2>`) are generated as Help levels, numbered 1, 2, and so on.

Note: When VAX DOCUMENT processes for Help and encounters a cross-reference, it *ignores* symbols and outputs only the text of the reference. For example, a reference to the symbol `overview_sec` results in See Overview Section, not See 2.1.1 Overview Section.

3.1.3 How to Selectively Include and Exclude Text for Help Output

Under normal circumstances, VAX DOCUMENT processes everything in your input file. In some cases, you may want to use text in printed documentation but exclude it from the Help files, or you may want to do the opposite. There are two sets of tags defined for HELP—`<BOOK_ONLY>` and `<HELP_ONLY>`—that allow you to exclude or include input text from Help output.

The `<HELP_ONLY>` and `<ENDHELP_ONLY>` tags identify text that is to be included in Help output only.

The `<BOOK_ONLY>` and `<ENDBOOK_ONLY>` tags identify text that is to be included in printed documentation only.

3.1.4 How to Handle Special Cases

In some cases, you will not want multiple-level Help. To keep all Help output on a single level, use the `<KEEP_HELP_LEVEL>` tag.

For example, if you had a command called KUNG FOO and you did not want a Help file that had KUNG at level 1 and FOO at level 2, you could use the `<KEEP_HELP_LEVEL>` tag as follows:

```
<keep_help_level>
<command>(KUNG FOO)
<endkeep_help_level>
```

The HELP doctype produces the following output:

```
1 KUNG_FOO
```

Note the underscore character (`_`) inserted between the two parts of the command. The `<KEEP_HELP_LEVEL>` concatenates all elements of the argument and places the result at level 1.

3.2 How to Read the Help File Online

Frequently, the headings for manuals are long and will create extremely long Help topic strings. The default topic size for VMS Help is 15 characters. If you have Help topics of 15 characters or more and you are making a Help library, you must use the `KEYSIZE` clause to the `/CREATE` qualifier. For example, use a command like the following:

```
$ LIBRARY/HELP/CREATE=KEYSIZE=n file file
```

In this example, “n” is the number of characters in the longest heading. You may want to set this number quite high. Otherwise, you will have to count the number of characters in all topics.

Using the HELP Doctype

The output from this command is an .HLB file. To read this .HLB file online, use the following command:

```
$ HELP/LIBRARY=DEVICE:[DIRECTORY]file.HLB
```

To get more information on creating Help files, on formatting Help files, on retrieving Help text, and on Help libraries, see these topics in the *VMS Librarian Utility Manual*.

3.3 HELP Doctype Tag Reference

This part of Chapter 3 provides reference information on all the tags specific to the HELP doctype.

<BOOK_ONLY>

Identifies text that you want to include only in printed or online output and not in Help output.

SYNTAX <BOOK_ONLY>

ARGUMENTS *None.*

related tags

- <HELP_ONLY>
- <SET_HELP_LEVEL>

required terminator

<ENDBOOK_ONLY>

DESCRIPTION The <BOOK_ONLY> tag identifies text that you want to include only in printed or online output and not in Help output.

EXAMPLE

This example shows how to code a file so that the text between the <BOOK_ONLY> and <ENDBOOK_ONLY> tags is included only in printed or online documentation and not in the Help (.HLP) file.

```
<BOOK_ONLY>
<P>When RSX . . .
<ENDBOOK_ONLY>

<P>When the operating system . . .

<BOOK_ONLY>
<P>When RSTS . . .
<ENDBOOK_ONLY>
```

In this case, the paragraphs that begin with “When RSX” and “When RSTS” would not be included in the .HLP file. Only the following paragraph would appear in the .HLP file:

When the operating system . . .

HELP Doctype Tag Reference

<HELP_ONLY>

<HELP_ONLY>

Identifies text that you want to include only in Help output and not in printed or online output.

SYNTAX <HELP_ONLY>

ARGUMENTS *None.*

related tags

- <BOOK_ONLY>
- <SET_HELP_LEVEL>

required terminator <ENDHELP_ONLY>

DESCRIPTION The <HELP_ONLY> tag identifies text that you want to include only in Help output and not in printed or online output.

EXAMPLE This example shows how to code a file so that the text between the <HELP_ONLY> and <ENDHELP_ONLY> tags is included only in the Help (.HLP) file and not in printed or online output.

```
<HELP_ONLY>
<P>When RSX . . .
<ENDHELP_ONLY>

<P>When the operating system . . .

<HELP_ONLY>
<P>When RSTS . . .
<ENDHELP_ONLY>
```

In this case, the paragraphs that begin with “When RSX” and “When RSTS” would be included only in the .HLP file. The following paragraph would appear only in the printed or online output:

When the operating system . . .

<KEEP_HELP_LEVEL>

Allows you to override the default multi-level Help output and keep the Help output at a single level. This tag affects only the <COMMAND> and <SUBCOMMAND> tags.

SYNTAX <KEEP_HELP_LEVEL>

ARGUMENTS *None.*

related tags

- <BOOK_ONLY>
- <HELP_ONLY>
- <SET_HELP_LEVEL>

required terminator

<ENDKEEP_HELP_LEVEL>

DESCRIPTION

The <KEEP_HELP_LEVEL> tag allows you to override the default multi-level Help output and keep the Help output at a single level. This tag affects only the <COMMAND> and <SUBCOMMAND> tags.

Remember that each word in a command is a different Help level, by default. The <KEEP_HELP_LEVEL> tag concatenates all elements of its argument and places the entire argument at a single level. For example, if you have a command called SET TERMINAL and you do not want a Help file with SET at level-1 and TERMINAL at level-2, which is the default, but want both SET and TERMINAL at level-1, use the <KEEP_HELP_LEVEL> and <ENDKEEP_HELP_LEVEL> tags to enclose the command.

EXAMPLE

This example shows how to use the <KEEP_HELP_LEVEL> and <ENDKEEP_HELP_LEVEL> tags to cause the enclosed command to be output as level-1 in the Help (.HLP) file.

HELP Doctype Tag Reference

<KEEP_HELP_LEVEL>

```
<COMMAND_SECTION>
<KEEP_HELP_LEVEL>
<COMMAND>(SET TERMINAL)
<ENDKEEP_HELP_LEVEL>
.
.
<COMMAND>(SET QUEUE)
.
.
<COMMAND>(SET PASSWORD)
.
.
<ENDCOMMAND_SECTION>
```

This example produces the following levels in the .HLP file:

```
1 SET_TERMINAL
1 SET
2 QUEUE
2 PASSWORD
```

<SET_HELP_LEVEL>

Allows you to alter the default Help levels in your Help files.

SYNTAX <SET_HELP_LEVEL>[(*number*)]

ARGUMENTS *number*

This is an optional argument. It specifies a positive or negative number that is added to or subtracted from the default value to determine a new Help level. Note that this number is not the Help level number, but a value to be applied to the default Help level.

To reset the default Help levels, specify zero (0) as the *number* argument or do not use an argument. For example, both the <SET_HELP_LEVEL> and <SET_HELP_LEVEL>(0) tags reset the default Help levels.

related tags

- <BOOK_ONLY>
- <HELP_ONLY>
- <KEEP_HELP_LEVEL>

DESCRIPTION

The <SET_HELP_LEVEL> tag allows you to alter the default Help levels in your Help files. Remember that each word in the command is a different Help level, by default. This tag changes all the default Help levels until you explicitly reset them using the tag again without an argument, or with the zero (0) argument.

For example, by default <HEAD1>, <STATEMENT>, and <COMMAND> tags produce level-1 Help topics. You may want, however, your level-1 “Command” topic to be a level-2 topic, and the “Format”, “Qualifier”, and “Description” sections, which are normally level-2 topics, to be level-3 topics. In this case, use the <SET_HELP_LEVEL>(1) tag before the Help level you want to alter. Using the argument *1* adds one level to the default level-1, thus adding one level to each subsequent Help level.

If you use a negative *number* argument, that number of levels is subtracted from the default Help level. For example, if you want your level-2 “Description” section to be a level-1, use the <SET_HELP_LEVEL>(-1) tag before the <DESCRIPTION> tag. If you want your level-3 “Example” section to be a level-1, use the <SET_HELP_LEVEL>(-2) tag before the <EXAMPLE> tag.

When you want to reset the default Help levels, use the <SET_HELP_LEVEL> tag with or without the zero (0) argument.

HELP Doctype Tag Reference

<SET_HELP_LEVEL>

EXAMPLE

This example shows how to use the <SET_HELP_LEVEL> tag to alter the default Help levels. One Help level is added to the commands following the <SET_HELP_LEVEL> tag. You reset the default Help levels with another <SET_HELP_LEVEL> tag, with the zero (0) argument or without an argument.

```
<COMMAND_SECTION>
<COMMAND> (SET TERMINAL)
.
.
.
<SET_HELP_LEVEL> (1)
<COMMAND> (SET QUEUE)
.
.
.
<SET_HELP_LEVEL> (0)
<COMMAND> (SET PASSWORD)
.
.
.
<ENDCOMMAND_SECTION>
```

This example produces the following levels in the .HLP file:

```
1 SET
2 TERMINAL
2 SET
3 QUEUE
2 PASSWORD
```

4 Using the LETTER Doctype

The LETTER doctype has one design, shown in Figure 4-1. It lets you create various types of correspondence such as business letters, personal letters, and memos in an $8\frac{1}{2} \times 11$ -inch format. Process files under this doctype using the LETTER doctype keyword on the DOCUMENT command line.

Figure 4-1 LETTER Doctype Design

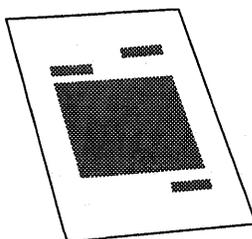


Table 4-1 lists the page layout of the LETTER doctype design.

Table 4-1 Page Layout of the LETTER Doctype Design

Page Layout Characteristics	
Running heads	None
Running feet	Page number, centered, not on first page
Page numbering	Sequential
Trim size	8 1/2 x 11 inches
Right margin	Unjustified (Ragged right)
Text Element Characteristics	
Headings	Unnumbered
Paragraphs	Flush left, no indent
Figures, tables, and examples	Numbered sequentially

The LETTER doctype does not support the full set of VAX DOCUMENT global tags. The following tags are not available in the LETTER doctype:

- <APPENDIX>
- <CHAPTER>
- <FRONT_MATTER>

Using the LETTER Doctype

- <HEAD1> through <HEAD6>
- <PART_PAGE>

Even though the LETTER doctype does not support the global numbered heading tags (<HEAD1>, <HEAD2>, and so on), it does support the global unnumbered heads (<SUBHEAD1> and <SUBHEAD2>). See the *VAX DOCUMENT Using Global Tags* for more information on global tags.

The LETTER doctype-specific tags let you label and format the text elements of letters and memos. In general, you use the LETTER doctype tags with the prefix MEMO_ (for example, the <MEMO_TO> tag) to create memos; you use the other tags in this doctype to create letters or headings in either memos or letters. Use the LETTER doctype tags in whatever order you choose. No one tag is restricted to either a memo or a letter format.

Table 4–2 summarizes the tags available in the LETTER doctype. Section 4.2 contains the reference information on the tags listed in this table.

Table 4–2 Tags Available in the LETTER Doctype

Tag Name	Description
<CC>	Labels the name of one person who is to receive a copy of the memo or letter. This tag places the heading cc: on the left margin, and then places the name of the person to the right of this heading on the same line.
<CCLIST>	Begins a list of one or more persons' names who are to receive a copy of the memo or letter. This tag places the heading cc: on the left margin. Use the <CC> tag to label each of the names in the list in the context of the <CCLIST> tag.
<CLOSING>	Labels the closing text of a letter and formats the closing text at the right margin, flush left. This text aligns with the output of the <FROM_ADDRESS> tag. This text is typically a closing line such as Yours Truly, followed by the name of the sender.
<DISTLIST>	Begins a list of people to whom the memo or letter is to be distributed. This list formats on the left margin beneath a heading of Distribution:.
<FROM_ADDRESS>	Identifies the name and address of the sender of a letter and formats that information at the right margin, flush left. This text aligns with the output of the <CLOSING> tag.
<MEMO_DATE>	Labels the date of a memo and formats that date near the left margin, flush left. This tag places the heading Date: on the left margin.
<MEMO_FROM>	Identifies the name and address of the sender of a memo and formats that information near the left margin, flush left. This tag places the heading From: on the left margin.
<MEMO_HEADER>	Centers the heading Interoffice Memorandum on the current line of the output page.
<MEMO_LINE>	Lets you specify your own information and headings in a format similar to the format used by the <MEMO_TO> or <MEMO_FROM> tags. This tag places your heading on the left margin and then places the first line of information text to the right of that heading on the same line; an additional line of information can be formatted under the first.

Table 4–2 (Cont.) Tags Available in the LETTER Doctype

Tag Name	Description
<MEMO_TO>	Identifies the name and address of the sender of a memo and formats that information left near the left margin, flush left. This tag places the heading To: on the left margin.
<SALUTATION>	Labels the greeting portion of a letter and formats that greeting on the left margin.
<SUBJECT>	Labels the subject of a memo or letter and formats that information Near the left margin, flush left. This tag places the heading Subject: on the left margin. The subject text formats on the same line and to the right of the heading.
<TO_ADDRESS>	Identifies the name and address of the receiver of a letter and formats that information on the left margin, flush left.

4.1 Sample Uses of the LETTER Doctype Tags

This section contains two examples of the LETTER doctype tags. The first example shows a sample memo and the second example shows a sample letter. You may find these sample files useful in understanding how the tags all fit together to create memos and letters.

Using the LETTER Doctype

4.1.1 A Sample Memo

This is the SDML code for a memo.

```
<MEMO_HEADER>
<MEMO_FROM>(Mr. Thurlow Smith\Corporate Company Accounting)
<MEMO_LINE>(Phone\181-1546)
<MEMO_TO>(Jack Jones\Payroll Accounting)
<MEMO_DATE>(March 17, 1989)
<CCLIST>
<CC>(Jim Walker)
<CC>(John Beam)
<CC>(D. M. Bones)
<ENDCCLIST>
<CC>(Departmental Distribution)
<SUBJECT>(Conference Report)
<CHEAD>(DEVELOPMENT OF ACCOUNTING TECHNOLOGY)
<P>
This conference was hosted by Numbers Inc. in Seattle, Wash., March 4 through 7.
The goal of the conference was to stimulate the development of accounting
technology.
<P>
My goals for attending the conference were to learn as much as I could about
accounting technology and to find out about existing products or projects
related to accounting methodology.
<SUBHEAD1>(Summary of Presentations Attended)
<P>
Major opening and closing presentations were directed at all conference
attendees. In between, there were choices between technical sessions and general
sessions, and I almost always felt a conflict. It was especially annoying
because there were not many clues as to what the differences were. Sometimes
technical seemed excessively technical, while general seemed at times overly
general.
<DISTLIST>
Bert Tom
Harry Lisa
Jim Melinda
Walter Jess
*All Trainees*
<ENDDISTLIST>
```

Figure 4-2 shows the corresponding memo output from that SDML file when processed with the LETTER keyword. Comparing these samples may be helpful in understanding how to use these tags to create memos. Should you wish to create this output yourself, you can obtain file MEMO_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Figure 4-2 LETTER Doctype Output Example for Memo

INTEROFFICE MEMORANDUM

FROM: Mr. Smith
Corporate Company Accounting
Phone: 181-1546

TO: Jack Jones
Payroll Accounting

DATE: March 17, 1989

cc: Jim Walker
John Beam
D. M. Bones

cc: Departmental Distribution

SUBJECT: Conference Report

DEVELOPMENT OF ACCOUNTING TECHNOLOGY

This conference was hosted by Numbers Inc. in Seattle, Wash., March 4 through 7. The goal of the conference was to stimulate the development of accounting technology.

My goals for attending the conference were to learn as much as I could about accounting technology and to find out about existing products or projects related to accounting methodology.

Summary of Presentations Attended

Major opening and closing presentations were directed at all conference attendees. In between, there were choices between technical sessions and general sessions, and I almost always felt a conflict. It was especially annoying because there were not many clues as to what the differences were. Sometimes technical was way too technical and general was way too general.

DISTRIBUTION:

Bert Tom
Harry Lisa
Jim Melinda
Walter Jess
All Trainees

Using the LETTER Doctype

4.1.2 A Sample Letter

This is the SDML code for a letter.

```
<FROM_ADDRESS>(Harvard University\Cambridge, MA\January 1, 1990, 10:00
A.M. EST)
<TO_ADDRESS>(Carol Jones\World Wide Wicker Co.\Seattle, WA)
<SALUTATION>(Hi Carol,)
<P>
This is a short excerpt from a symposium I went to on letter writing.
I thought you might find it interesting. We really ought to have lunch
some time.
<P>
The excerpt follows:
<P>
There are generally two kinds of letters:
<LIST>(UNNUMBERED)
<LE> Business letters
<LE> Personal letters
<ENDLIST>
<HEAD>(Writing a Business Letter\19_WritingaBusinessLetter)
<P>
When writing a business letter, form can be very important.
In many cases, the form of the letter can be nearly as important as the content
of the letter.
<CHEAD>(Writing a Letter to Request Information)
<P>
A business letter is often used to request information from
an official source. It is important to specify very clearly what
information you need, and for what you need it. If your information needs
are unclear, your request may not be filled.
<CLOSING>(Best Wishes,\Bob Smith\Chairman, CZZA Committee)
```

Figure 4-3 shows the corresponding letter output from that SDML file when processed with the LETTER keyword. Comparing these samples may be helpful in understanding how to use these tags to create letters. Should you wish to create this output yourself, you can obtain file LETTER_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Figure 4-3 LETTER Doctype Output Example for Letter

Harvard University
Cambridge, MA
January 1, 1990, 10:00 A.M. EST

Carol Jones
World Wide Wicker Co.
Seattle, WA

Hi Carol,

This is a short excerpt from a symposium I went to on letter writing. I thought you might find it interesting. We really ought to have lunch some time.

The excerpt follows:

There are generally two kinds of letters:

- Business letters
- Personal letters

Writing a Business Letter

When writing a business letter, form can be very important. In many cases, the form of the letter can be nearly as important as the content of the letter.

Writing a Letter to Request Information

A business letter is often used to request information from an official source. It is important to specify very clearly what information you need, and for what you need it. If your information needs are unclear, your request may not be filled.

Best Wishes,

Bob Smith
Chairman, CZZA Committee

Using the LETTER Doctype

4.2 LETTER Doctype Tag Reference

This part of Chapter 4 provides reference information on all the tags specific to the LETTER doctype.

<CC>

Lists the name of someone who is to receive a copy of a memo or letter.

SYNTAX **<CC>** (*receiver name*)

ARGUMENTS *receiver name*
Specifies the name of someone who should receive a copy of the memo or letter.

related tags • <CCLIST>

DESCRIPTION The <CC> tag lists a single name of someone who is to receive a copy of a memo or letter. Use this tag by itself or in the context of the <CCLIST> tag. If you use the <CC> tag alone, it places the heading cc: on the left margin and places the *receiver name* argument on the same line as that heading. If you use the <CC> tag in the context of the <CCLIST> tag, it places the text of the *receiver name* argument in the same location as when it is used without the <CCLIST> tag, but omits the cc: heading.

EXAMPLES The following example shows the beginning of a letter using the <CCLIST> tag with the <CC> tag.

```
1 <MEMO_FROM>(Bob Smith\Harvard University)
  <MEMO_TO>(Carol Jones)
  <CCLIST>
  <CC>(Mr. A. Square)
  <CC>(Ms. B. Box)
  <ENDCCLIST>
  <SUBJECT>(Ted Fields and Alice Johnson)
  <P>
  This is some text to show you where the text begins.
  .
  .
  .
```

LETTER Doctype Tag Reference

<CC>

The following example shows the beginning of a letter using only the <CC> tag.

```
2 <MEMO_FROM>(Bob Smith\Harvard University)
  <MEMO_TO>(Carol Jones)
  <CC>(Mr. A. Square)
  <SUBJECT>(Ted Fields and Alice Johnson)
  <P>
  This is some text to show you where the text begins.
  .
  .
  .
```

<CCLIST>

Begins a list of persons to whom you want to send a copy of a memo or letter.

SYNTAX <CCLIST>

ARGUMENTS *None.*

related tags

- <CC>
- <DISTLIST>

required terminator <ENDCCLIST>

DESCRIPTION The <CCLIST> tag begins a list of persons to whom you want to send a copy of a memo or letter. The <CCLIST> tag places the heading cc: on the left margin. Specify the names using the <CC> tag.

The names format such that the argument to the first <CC> tag outputs on the same line as the heading, and the text arguments associated with any following <CC> tags are placed immediately beneath the argument to the first <CC> tag.

EXAMPLE The following example shows how to use the <CC> tag with the <CCLIST> tag.

```
<MEMO_FROM>(Bob\Harvard University)
<MEMO_TO>(Carol)
<CCLIST>
<CC>(Mr. A. Square)
<CC>(Ms. B. Box)
<ENDCCLIST>
<SUBJECT>(Ted and Alice)
<P>
This is some text to show you where the text begins.
.
.
.
<CLOSING>(Best Wishes,\Bob\Chairman, QZZA, Inc.)
```

LETTER Doctype Tag Reference

<CLOSING>

<CLOSING>

Specifies in one to five lines the closing of a letter.

SYNTAX

<CLOSING>(*closing line-1*[\ *closing line-2* . . . [\ *closing line-5*]])

ARGUMENTS

closing line-n

Specifies in one to five lines the closing of a letter.

related tags

- <FROM_ADDRESS>
 - <MEMO_FROM>
 - <MEMO_TO>
 - <SALUTATION>
 - <TO_ADDRESS>
-

DESCRIPTION

The <CLOSING> tag specifies in one to five lines the closing of a letter. The *closing line-n* arguments are all placed to the right of the center of the page. Four blank lines are placed between the first and the second *closing line-n* arguments to allow room for the signature of the writer of the letter.

Typically, the first argument is the name of the sender, and the second through fifth arguments are information about the sender (for example, the sender's position, title, and so on).

EXAMPLE

The following example shows the closing of a letter using the <CLOSING> tag.

```
.  
. .  
This is the end of the letter text.  
<CLOSING>(Best Wishes,\Bob\Chairman, QZZA, Inc.)
```

<DISTLIST>

Begins a list of persons to whom you want to distribute a memo or letter.

SYNTAX <DISTLIST>

ARGUMENTS *None.*

required terminator <ENDDISTLIST>

DESCRIPTION The <DISTLIST> tag begins a list of persons to whom you want to distribute a memo or letter. The <DISTLIST> tag places the heading **Distribution:** on the left margin. The names of the people on the distribution list appear beneath the heading, formatted exactly as you entered them between the <DISTLIST> and <ENDDISTLIST> tags.

The <DISTLIST> tag retains all spacing and capitalization exactly as entered. This lets you place asterisks before certain names, indent certain names, and so on. Compare this tag to the <CCLIST> tag.

EXAMPLE The following example shows the end of a letter using the <DISTLIST> tag. The format of the text in the context of the <DISTLIST> tag will be retained exactly as entered.

```
<DISTLIST>
*Bob
  Carol  *Ted  Alice
  Pete   Jon  *Art
* - Indicates primary reviewer
<ENDDISTLIST>
```

LETTER Doctype Tag Reference

<FROM_ADDRESS>

<FROM_ADDRESS>

Places the name and address of the sender of a letter flush left at the right margin.

SYNTAX

<FROM_ADDRESS>(*address line-1* [\ *address line-2* . . . [\ *address line-5*]])

ARGUMENTS

address line-n

Specifies one to five lines of text that contain the name and address of the sender of the letter.

related tags

- <MEMO_FROM>
 - <TO_ADDRESS>
-

DESCRIPTION

The <FROM_ADDRESS> tag places the name and address of the sender of a letter flush left at the right margin. The <FROM_ADDRESS> tag outputs one to five lines of text based on the number of *address line* arguments specified. Each of these arguments outputs flush left on a new line near the right margin.

Alternatively, you can use the <MEMO_FROM> tag to specify this same information, but in a different format. See the description of the <MEMO_FROM> tag in this chapter for more information on that tag.

EXAMPLE

The following example shows the beginning of a letter that uses the <FROM_ADDRESS> tag.

```
<FROM_ADDRESS>(Bob Smith\Harvard University\Cambridge, MA)
<TO_ADDRESS>(Carol Jones\World Wide Wicker Co.\Seattle, WA)
<SUBJECT>(Ted and Alice)
<SALUTATION>(Hi Carol,)
<P>
This is the text of the letter...
.
.
.
<CLOSING>(Best Wishes,\Bob Smith\Chairman, QZZA, Inc.)
```

<MEMO_DATE>

Creates a line in a memo or letter that displays the date after the heading Date:.

SYNTAX <MEMO_DATE>(date argument)

ARGUMENTS *date argument*

Specifies the date of the memo or letter. This argument can be the global <DATE> tag (which returns the date the file was processed on), or a date that you specify.

related tags

- <FROM_ADDRESS>
- <MEMO_LINE>
- <MEMO_TO>
- The global <DATE> tag

DESCRIPTION

The <MEMO_DATE> tag creates a line in a memo or letter that displays the date after the heading Date:.

If you want the date to be the date on which you processed the file, use the global <DATE> tag as the argument to the <MEMO_DATE> tag.

If you want a date that does not vary each time you process the file, or a date that follows a different format than the format output by the <DATE> tag, enter that date explicitly as a text argument to the <MEMO_DATE> tag.

EXAMPLES

The following example shows the beginning of a memo using the <MEMO_DATE> tag with the global <DATE> tag as an argument.

1 <MEMO_HEADER>
<MEMO_FROM>(Bob\Dept. of English)
<MEMO_TO>(Carol\Dept. of Archeology)
<MEMO_LINE>(Req No.\ARC-132)
<MEMO_DATE>(<DATE>)
<SUBJECT>(Awards for Ted and Alice)
<P>
This is the text of the memo...

LETTER Doctype Tag Reference

<MEMO_DATE>

The following example shows the beginning of a memo using the <MEMO_DATE> tag with a text string as an argument.

```
2 <MEMO_HEADER>
  <MEMO_FROM>(Bob Smith\Dept. of English)
  <MEMO_LINE>(Phone:\9-5151)
  <MEMO_TO>(Carol Jones\Dept. of Archeology)
  <MEMO_LINE>(Req No.\ARC-132)
  <MEMO_DATE>(January 1, 1987, 10:00 pm)
  <SUBJECT>(Awards for Ted and Alice)
  <P>
  This is the text of the memo...
```


LETTER Doctype Tag Reference

<MEMO_HEADER>

<MEMO_HEADER>

Centers the heading Interoffice Memorandum in bold letters on the page.

SYNTAX <MEMO_HEADER>

ARGUMENTS *None.*

related tags

- <MEMO_FROM>
- <MEMO_TO>

DESCRIPTION The <MEMO_HEADER> tag centers the heading Interoffice Memorandum in bold letters on the page. This tag accepts no arguments.

EXAMPLE The following example shows a typical beginning of a memo using the <MEMO_HEADER> tag.

```
<MEMO_HEADER>
<MEMO_FROM>(Bob Smith\Dept. of English)
<MEMO_TO>(Carol Jones\Dept. of Archeology)
<MEMO_DATE>(<DATE>)
<SUBJECT>(Ted and Alice)
<P>
This is the text of the memo...
```


LETTER Doctype Tag Reference

<MEMO_LINE>

EXAMPLE

The following example shows how to use the <MEMO_LINE> tag to create an additional line of information with a heading. In this example, the heading Corp: is created with the text following it being Drofnats Ltd. Note that the *heading text* argument does not exceed seven characters; note also that even though you want a colon in the heading, you do not specify it as part of the *heading text* argument.

<MEMO_HEADER>

<MEMO_FROM>(J. Simpson\Accounting Consultant)

<MEMO_LINE>(Corp\Drofnats Ltd.)

<MEMO_TO>(Mr. Smith\ACME Corporate Accounting)

<MEMO_DATE>(March 17, 1986)

<CC>(Departmental Distribution)

<SUBJECT>(Conference Report)

<P>This conference was hosted by Numbers Inc. in Seattle, Washington

.
.
.

LETTER Doctype Tag Reference

<SALUTATION>

<SALUTATION>

Specifies the salutation for the letter.

SYNTAX

<SALUTATION> (*greeting text*)

ARGUMENTS

greeting text

Specifies the text of the salutation, including any punctuation.

related tags

- <FROM_ADDRESS>
- <MEMO_FROM>
- <MEMO_TO>
- <TO_ADDRESS>

DESCRIPTION

The <SALUTATION> tag specifies the salutation for the letter. The opening greeting of your letter or memo, for example, might be Dear Sirs:. This text begins at the left margin.

You must specify any punctuation that is part of the salutation (such as a comma, colon, or semicolon) as part of the *greeting text* argument.

EXAMPLE

The following example shows a use of the <SALUTATION> tag in the beginning of a letter. Note that you must provide any needed punctuation, such as the comma, after Hi Carol in this example.

```
<FROM_ADDRESS>(Bob Smith\Harvard University\Cambridge, MA)
<TO_ADDRESS>(Carol Jones\World Wide Wicker Co.\Seattle, WA)
<SUBJECT>(Ted and Alice)
<SALUTATION>(Hi Carol,)
<P>
This is the text of the letter...
```

<SUBJECT>

Specifies the subject of a memo or letter and places this information with a heading of Subject: at the left margin.

SYNTAX <SUBJECT>(subject text)

ARGUMENTS *subject text*
Specifies the text that describes the subject of a memo or letter.

related tags

- <FROM_ADDRESS>
- <MEMO_FROM>
- <MEMO_LINE>
- <MEMO_TO>

DESCRIPTION The <SUBJECT> tag specifies the subject of a memo or letter and places this information with a heading of Subject: at the left margin. It places the text from the *subject text* argument on that same line.

EXAMPLE The following example shows a use of the <SUBJECT> tag in the beginning of a letter. Note that this tag can also be used in a memo.

```
<FROM_ADDRESS>(Bob Smith\Harvard University\Cambridge, MA)
<TO_ADDRESS>(Carol Jones\World Wide Wicker Co.\Seattle, WA)
<SUBJECT>(Ted and Alice)
<SALUTATION>(Hi Carol,)
<P>
This is the text of the letter...
```


5

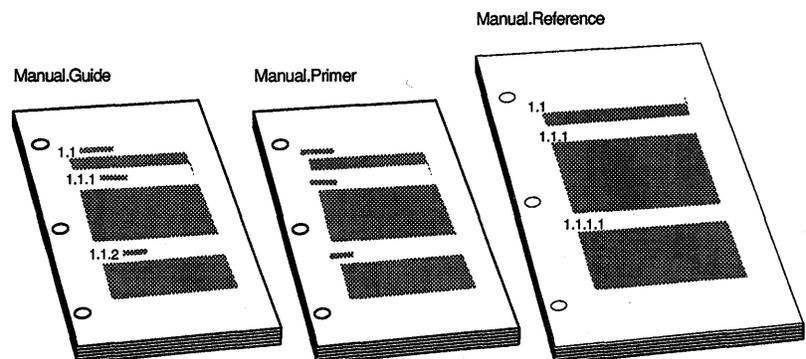
Using the MANUAL Doctype

The MANUAL doctype has three designs for printed documentation, shown in Figure 5-1, and one design for online documentation:

- **MANUAL.GUIDE**
Creates a users' manual in a 7×9-inch format with numbered headings. This design is intended for chapter-oriented tutorial material.
- **MANUAL.PRIMER**
Creates a users' manual in a 7 × 9 inch format with unnumbered headings. This design is intended for chapter-oriented primer material.
- **MANUAL.REFERENCE**
Creates a users' manual in an 8½ × 11 inch format with numbered headings. This design is intended for reference material, and is the default design.
- **MANUAL.ONLINE**
Creates an online users' manual in a 5.9 × 6.6-inch format with numbered headings and ragged right margin. This design is solely for online display. Refer to Chapter 7 for information about online documentation.

Table 5-1, Table 5-2, and Table 5-3 list the page layout characteristics of the MANUAL doctype designs for printed documentation.

Figure 5-1 MANUAL Doctype Designs



ZK-1925A-GE

Using the MANUAL Doctype

Table 5–1 Page Layout of the MANUAL.GUIDE Doctype Design

Page Layout Characteristics	
Running heads	Chapter title text
Running feet	Chapter number and page number
Page numbering	Chapter-oriented
Trim size	7 x 9 inches
Gutter width	2.5 picas
Right margin	Justified

Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

Table 5–2 Page Layout of the MANUAL.PRIMER Design

Page Layout Characteristics	
Running heads	Chapter title text
Running feet	Page number
Page numbering	Chapter oriented
Trim size	7 x 9 inches
Gutter width	2.5 picas
Right margin	Justified

Text Element Characteristics	
Headings	Unnumbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

Table 5–3 Page Layout of the MANUAL.REFERENCE Design

Page Layout Characteristics	
Running heads	None
Running feet	Chapter title text and page number
Page numbering	Chapter oriented
Trim size	8 1/2 x 11 inches
Gutter width	2.5 picas
Right margin	Justified

Table 5-3 (Cont.) Page Layout of the MANUAL.REFERENCE Design

Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

The MANUAL doctype designs require no doctype-specific tags, but accept the full range of VAX DOCUMENT global tags. See the *VAX DOCUMENT Using Global Tags* for more information on global tags.

Process a file with the MANUAL doctype by using one of the doctype keywords in the preceding list on the DOCUMENT command line. The following example shows how to process a file named MYMANUAL.SDML with the MANUAL doctype to create a reference manual.

```
$ DOCUMENT MYMANUAL MANUAL.REFERENCE LN03
```

5.1 Example of Using the MANUAL Doctype

This section contains a sample SDML file for producing a portion of a manual. The example is for the first few pages of a hardware manual created using the MANUAL.REFERENCE doctype. The output of this manual sample shows the title page and some numbered headings and a table in the body of the manual. You may find these samples useful in understanding how global tags can be used in the MANUAL doctype to create various kinds of manuals.

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(Series III Overthruster\User Manual)
<ABSTRACT>
This book describes the Series III Overthruster, the Series III
manual-override mode and the Overthruster monitor utilities.
<ENDABSTRACT>
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>

<CHAPTER>(Introduction to the Overthruster\INTRO_CHAP)
<P>
The Series III Overthruster is an intelligent mass thrust device,
designed to deliver controlled thrust from a ZX-300 type drive unit.
The Series III family provides enhanced data collection and control
utilities far superior to those present in the Series II.

<HEAD1>(Series III Overthruster Models\thruster)
<p>
The Series III Overthruster family has three models.
<REFERENCE>(MODEL_TAB) lists these models and the salient
features of each.

<TABLE>(Comparison of Series III Overthruster Models\MODEL_TAB)
<TABLE_SETUP>(4\25\10\10)
<TABLE_HEADS>(Feature\<SPAN>(3\LEFT)Models\ \ )
<TABLE_HEADS>( \SIII-030\SIII-050\SIII-070)
<TABLE_ROW>(Data Channels Supported\2\4\4)
<TABLE_ROW>(I/O Control Processor \A-L35\J19\J19)
<TABLE_ROW>(Drive Unit\ZX-301\ZX-301\ZX-310A)
<TABLE_ROW>(Power Source\TY-100\TY-100A \TY-200)
<TABLE_ROW>(Control Memory\128Kb\128Kb\512Kb)
<TABLE_ROW>(Data Memory\ -- \ -- \512Kb)
<ENDTABLE>

<HEAD1>(Series III Overthruster Functional Description\thruster_func)
<p>
The Series III Overthruster is an intelligent mass thrust device
in accordance with the DS8 architecture.
```

Figure 5–2 and Figure 5–3 show the corresponding output from that SDML file. Comparing these samples may be helpful in understanding how to use these tags to create a manual. Should you wish to create this output yourself, you can obtain file ARTICLE_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Figure 5-2 MANUAL Doctype Output Example, Title Page

Series III Overthruster User Manual

This book describes the Series III Overthruster, the Series III manual-override mode and the Overthruster monitor utilities.

Digital Equipment Corporation

Using the MANUAL Doctype

Figure 5-3 MANUAL Doctype Output Example, Interior Page

Chapter 1

Introduction to the Overthruster

The Series III Overthruster is an intelligent mass thrust device, designed to deliver controlled thrust from a ZX-300 type drive unit. The Series III family provides enhanced data collection and control utilities far superior to those present in the Series II.

1.1 Series III Overthruster Models

The Series III Overthruster family has three models. Table 1-1 lists these models and the salient features of each.

Table 1-1: Comparison of Series III Overthruster Models

Feature	Models		
	SIII-O30	SIII-O50	SIII-O70
Data Channels Supported	2	4	4
I/O Control Processor	A-L35	J19	J19
Drive Unit	ZX-301	ZX-301	ZX-310A
Power Source	TY-100	TY-100A	TY-200
Control Memory	128Kb	128Kb	512Kb
Data Memory	—	—	512Kb

1.2 Series III Overthruster Functional Description

The Series III Overthruster is an intelligent mass thrust device in accordance with the DS8 architecture.

6

Using the MILSPEC Doctype

VAX DOCUMENT has two doctype designs for creating printed military documents, as shown in Figure 6-1, and one design for online documentation:

- MILSPEC.SECURITY

Use MILSPEC.SECURITY to produce documents requiring security classifications or to produce documents that conform to the U.S. Department of Defense standard DOD-STD-2167 or DOD-STD-2167A.

MILSPEC.SECURITY includes tags for security classification, numbering of figures and tables by section number, additional heading levels, 1- to 4-line running titles, single-line running feet, and the ability to use proportionally spaced fonts in code examples.

- MILSPEC.DRAFT

Use MILSPEC.DRAFT to produce double-spaced draft documents.

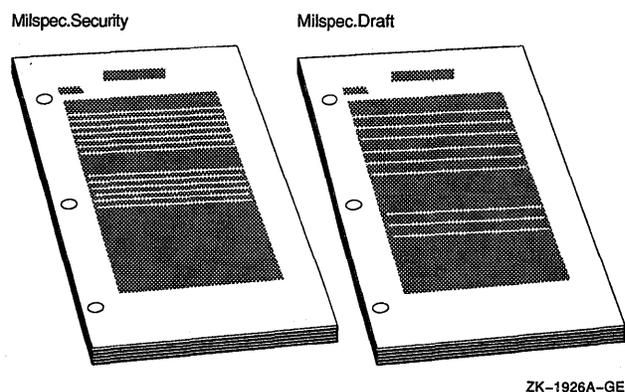
- MILSPEC.ONLINE

Use MILSPEC.ONLINE to produce documents that can be viewed with Bookreader. This doctype accepts all tags that are valid in MILSPEC.SECURITY. Refer to Chapter 7 for information about online documentation.

MILSPEC.ONLINE creates an online document in a 5.9 × 6.6 inch format with numbered headings and ragged right margin.

Table 6-1 lists the page layout of the MILSPEC doctype designs for printed documentation.

Figure 6-1 MILSPEC Doctype Designs



ZK-1926A-GE

Using the MILSPEC Doctype

Table 6–1 Page Layout of the MILSPEC Designs

Page Layout Characteristics	
Running heads	Two-line running heading
Running feet	Sequential using Arabic numerals
Page numbering	Sequential using Arabic numerals
Trim size	8 1/2 x 11 inches
Right margin	Justified

Text Element Characteristics	
Headings	Numbered using Arabic numerals
Paragraphs	Numbered using Arabic numerals
Figures, Examples	Numbered using Arabic numerals
Tables	Numbered using Roman numerals

6.1 MILSPEC Template Files

VAX DOCUMENT provides template files for MIL-STD-490A documents and for DOD-STD-2167 or DOD-STD-2167A Data Item Description documents in the directory DOC\$TEMPLATES. Each template file contains all the headings, titles, and other text elements required by the MIL-STD-490A, DOD-STD-2167 and DOD-STD-2167A specifications for conforming documents. Comments in the template input files guide you in placing your information into the file. See Section 6.3, Section 6.4.1, and Section 6.4.2 for more information on using these templates.

Another source of templates exists. If you have the VAX Language-Sensitive Editor (LSE) Version 2.0 or higher installed on your system, you can access LSE templates for the DOD-STD-2167 or DOD-STD-2167A Data Item Description documents and then expand the appropriate placeholders to create source files for these doctypes. See the *VAX DOCUMENT User's Guide, Volume 1* for more information on using LSE with VAX DOCUMENT.

6.2 MILSPEC Doctype Conformance and Format

The MILSPEC doctype produces documents conforming to United States Department of Defense Military Specification Standard MIL-STD-490A, published June 4, 1985, or documents conforming to United States Department of Defense Standard DOD-STD-2167, also published June 4, 1985.

The MILSPEC doctype provides 24 template SDML files for documents that conform to the Department of Defense standard DOD-STD-2167 for Data Item Descriptions. Data Item Descriptions are MIL-STD-490A documents that are specialized for a particular kind of information. DOD-STD-2167 specifies the exact form of each data item description. See Section 6.4.1 for more information on creating a DOD-STD-2167 document.

Documents produced using the MILSPEC doctype design have the following general format:

- Pages, formal figures, and formal tables are numbered sequentially throughout the document and are not numbered by chapter, section, or appendix.
- Paragraphs are numbered using the global numbered heading tags (<HEAD1> or <HEAD2>) to create the Arabic numerals. A period automatically ends the numbered paragraph headings.
- Formal tables are numbered using Roman numerals.
- The global <PREFACE> tag automatically generates a preface section heading of Foreword rather than Preface. All other VAX DOCUMENT doctypes use the heading Preface.
- The table of contents begins on page ii rather than on page iii. All other VAX DOCUMENT doctypes that support the automatic creation of tables of contents begin the table of contents on page iii.
- Appendixes are numbered using Roman numerals. Sections and paragraphs in an appendix are numbered in Arabic numerals. The Arabic numerals correspond to the appendix number multiplied by 10 if there are fewer than 10 such sections or paragraphs or by 100 if there are 10 or more sections or paragraphs. For example, in Appendix II, the first major paragraph would be paragraph 20.1, and the second paragraph would be 20.2.

The MILSPEC doctype accepts the full range of VAX DOCUMENT global tags, with the exception of the <PART> and <PART_PAGE> tags. Table 6–2 summarizes the tags specific to the MILSPEC doctype, which are also used in the MILSPEC.SECURITY and MILSPEC.DRAFT doctype designs. See Section 6.5 for more information on any of these tags.

Table 6–2 MILSPEC Doctype Tags

Tag Name	Description
<SET_APPENDIX_NUMBER>	Overrides the default Roman numeral VAX DOCUMENT usually assigns to an appendix.
<SIGNATURE_LINE>	Creates up to two rules on a line (one in each signature column) and places a name below each rule; each rule serves as a signatory line for the person listed below it.
<SIGNATURE_LIST>	Begins a 2-column listing of signature lines on the title page and places a heading above each column. You create each row of signature lines using the <SIGNATURE_LINE> tag in the context of the <SIGNATURE_LIST> tag.
<SPECIFICATION_INFO>	Creates a listing of information about the specification document on the title page and creates a 2-line running heading that lists the specification number and date for the rest of the document.
<SPEC_TITLE>	Creates a title with up to seven centered lines on the title page.
<SUBTITLE>	Creates a subtitle with up to seven centered lines on the title page.

Using the MILSPEC Doctype

6.2.1 Example of Using the MILSPEC.SECURITY and MILSPEC.DRAFT Doctypes

This section contains a sample of the first pages of a specification created using the MILSPEC doctype tags. This sample includes the title page of the specification and the first page of text after the title page.

Note that <SET_HEADINGS> is used within the <DOCUMENT_ATTRIBUTES> tag to cause running headings to be centered. Also note that the <SECURITY> and <HIGHEST_SECURITY_CLASS> tags are used to label text elements with security classifications.

The SDML code for the specification is shown first, followed by the output from that SDML code when processed for a POSTSCRIPT destination and the MILSPEC.SECURITY doctype, and then followed by the output when processed for a POSTSCRIPT destination and the MILSPEC.DRAFT doctype.

```
<DOCUMENT_ATTRIBUTES>
<SET_HEADINGS>(CENTERED)
<ENDDOCUMENT_ATTRIBUTES>

<FRONT_MATTER>
<TITLE_PAGE>
<SPECIFICATION_INFO>(12345B\a142-b4\<<DATE>\Part I of Three Parts)

<ONLINE_TITLE>(PDS For the Overthruster Monitor System)

<SPEC_TITLE>(Preliminary Development Specification
\For the Overthruster Monitor System
\Series (Series Configuration Number)
\Order Number (Approved Order Number))

<SUBTITLE>(Submitted Under\Contract A00000--11--A--2222\<<highest_security_class>)

<set_security_class>(C_LEVEL\CL\C_LEVEL\5)
<SIGNATURE_LIST>(Authenticated by:\Approved by:)
<SIGNATURE_LINE>(Procurer\Program Manager)
<SIGNATURE_LINE>(Date\Technical Director)
<SIGNATURE_LINE>(\Consultant)
<ENDSIGNATURE_LIST>
<ENDTITLE_PAGE>
<COMMENT>(\<<endsecurity>)
<CONTENTS_FILE>
<ENDFRONT_MATTER>

<CHAPTER>(Scope\first_sec)
<HEAD1>(Scope\scope_head)
<P>
This document establishes all specifications for the design and
production of the Overthruster Monitor System (USN-122-233x) by our Corporation.
<HEAD1>(Purpose\purpose_sec)
<P>
The purpose of this document is to specify all design and production
dimensions of the Overthruster Monitor System. This will ensure that all
essential requirements are met and that all concerns are addressed.
```

```

<set_security_class>(C_LEVEL\CL\C_LEVEL\5)
<HEAD2>(Primary Purpose\Primary_purpose_sec)
<P>
The primary purpose is to enrich the quality dimension of our product.
<COMMENT><endsecurity>
<HEAD3>(Secondary Purpose\secondary_purpose_sec)
<P>
The secondary purpose is to create a corporate strategy for the product that
encompasses the goals established in <REFERENCE>(primary_purpose_sec\VALUE).
<P>
<ELLIPSIS>
<P>
Production of the Overthruster Monitor System will necessitate a reorganization
of our current production strategy. In order to produce the projected
quantities of the Overthruster Monitor System we will have to make the changes
summarized in <REFERENCE>(OMS_tab).

<set_security_class>(C_LEVEL\CL\C_LEVEL\5)
<TABLE>(Overthruster Monitor System <oparen>OMS<cparen> Production Line Impact\OMS_tab)
<TABLE_ATTRIBUTES>(WIDE)
<TABLE_SETUP>(2\18)
<TABLE_HEADS>(Production<LINE>Line Name\Production<LINE>Line Modification)
<TABLE_ROW>(Alpha<LINE> (System Units)
\100% conversion from Series II OMS production to Series III OMS production.)
<TABLE_ROW>(Beta<LINE> (Unit Stands)
\Increase production 30% and designate 50% of
that production for the Overthruster Monitor System sales.)
<TABLE_ROW>(Gamma<LINE> (Model I Power Supplies)
\Phase out production over 6-month time frame.
<COMMENT><endsecurity>)
<set_security_class>(S_LEVEL\SL\S_LEVEL\6)
Will be modified to produce the new Model IIA Power Supply.
<COMMENT><endsecurity>)
)
<TABLE_ROW>(Omega<LINE> (Model IIA Power Supplies)
\Increase production by 35% until Gamma comes on-line in 6 months.)
<ENDTABLE>

```

Figure 6-2 and Figure 6-3 show the corresponding output from the sample SDML file when processed using the MILSPEC.SECURITY doctype. Comparing these samples may be helpful in understanding how to use these tags to create Milspec documents. Should you wish to create this output yourself, you can obtain file MILSPEC_SECURE_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Figure 6-4, Figure 6-5 and Figure 6-6 show the corresponding output from the same sample SDML file when processed with the MILSPEC.DRAFT doctype. Comparing these samples may be helpful in understanding how to use these tags to create Milspec draft documents. Should you wish to create this output yourself, you can obtain file MILSPEC_DRAFT_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Using the MILSPEC Doctype

Figure 6-2 MILSPEC.SECURITY Doctype Output Example, Title Page

S LEVEL

12345B
a142-b4
21 January 1991
Part I of Three Parts

**Preliminary Development Specification
For the Overthruster Monitor System
Series (Series Configuration Number)
Order Number (Approved Order Number)**

**Submitted Under
Contract A0000-11-A-2222
S LEVEL**

Authenticated by:

Approved by:

Procurer

Program Manager

Date

Technical Director

Consultant

S LEVEL

Figure 6-3 MILSPEC.SECURITY Doctype Output Example, Interior Page

S LEVEL

12345B
21 January 1991

1. SCOPE

1.1 *Scope.* This document establishes all specifications for the design and production of the Overthruster Monitor System (USN-122-233x) by our Corporation.

1.2 *Purpose.* The purpose of this document is to specify all design and production dimensions of the Overthruster Monitor System. This will ensure that all essential requirements are met and that all concerns are addressed.

1.2.1 (CL) *Primary Purpose.* (CL) The primary purpose is to enrich the quality dimension of our product.

1.2.1.1 *Secondary Purpose.* The secondary purpose is to create a corporate strategy for the product that encompasses the goals established in 1.2.1.

.
. .
.

Production of the Overthruster Monitor System will necessitate a reorganization of our current production strategy. In order to produce the projected quantities of the Overthruster Monitor System we will have to make the changes summarized in Table 1.2.1.1-I.

Table 1.2.1.1-I (CL). Overthruster Monitor System (OMS) Production Line Impact

Production Line Name	Production Line Modification
Alpha (System Units)	100% conversion from Series II OMS production to Series III OMS production.
Beta (Unit Stands)	Increase production 30% and designate 50% of that production for the Overthruster Monitor System sales.
Gamma (Model I Power Supplies)	Phase out production over 6-month time frame. (SL) Will be modified to produce the new Model IIA Power Supply.
Omega (Model IIA Power Supplies)	Increase production by 35% until Gamma comes on-line in 6 months.

S LEVEL

1

Using the MILSPEC Doctype

Figure 6-4 MILSPEC.DRAFT Doctype Output Example, Title Page

S LEVEL

12345B

a142-b4

9 January 1991

Part I of Three Parts

**Preliminary Development Specification
For the Overthrust Monitor System
Series (Series Configuration Number)
Order Number (Approved Order Number)**

**Submitted Under
Contract A0000-11-A-2222
S LEVEL**

Authenticated by:

Approved by:

Procurer

Program Manager

Date

Technical Director

Consultant

S LEVEL

Figure 6-5 MILSPEC.DRAFT Doctype Output Example, Interior Page 1

S LEVEL

12345B
9 January 1991

1. SCOPE

1.1 *Scope.* This document establishes all specifications for the design and production of the Overthrustor Monitor System (USN-122-233x) by our Corporation.

1.2 *Purpose.* The purpose of this document is to specify all design and production dimensions of the Overthrustor Monitor System. This will ensure that all essential requirements are met and that all concerns are addressed.

1.2.1 (CL) *Primary Purpose.* (CL) The primary purpose is to enrich the quality dimension of our product.

1.2.1.1 *Secondary Purpose.* The secondary purpose is to create a corporate strategy for the product that encompasses the goals established in 1.2.1.

Production of the Overthrustor Monitor System will necessitate a reorganization of our current production strategy. In order to produce the projected quantities of the Overthrustor Monitor System we will have to make the changes summarized in Table 1.2.1.1-I.

Table 1.2.1.1-I (CL). Overthrustor Monitor System (OMS) Production Line Impact

Production Line Name	Production Line Modification
Alpha (System Units)	100% conversion from Series II OMS production to Series III OMS production.
Beta (Unit Stands)	Increase production 30% and designate 50% of that production for the Overthrustor Monitor System sales.
Gamma (Model I Power Supplies)	Phase out production over 6-month time frame. (SL) Will be modified to produce the new Model IIA Power Supply.

S LEVEL

1

Using the MILSPEC Doctype

Figure 6-6 MILSPEC.DRAFT Doctype Output Example, Interior Page 2

12345B
9 January 1991

C LEVEL

**Table 1.2.1.1-I (CL) (Continued). Overthrust
Monitor System (OMS) Production Line Impact**

Production Line Name	Production Line Modification
Omega (Model IIA Power Supplies)	Increase production by 35% until Gamma comes on-line in 6 months.

6.3 Creating MIL-STD-490A Documents

To create a MIL-STD-490A document, you copy the template file MILSPEC_SAMPLE.SDML from the directory DOC\$TEMPLATES into your work directory, then edit it to insert your text. Alternatively, you can create your own file. If you create a new file, you may still want to look at the template SDML input file as a guide.

To create documents that conform to MIL-STD-490A, you use the MILSPEC doctype tags in the following manner:

- Create the required cover page for the document using the <SPEC_TITLE> and <SPECIFICATION_INFO> tags in the context of the global <TITLE_PAGE> tag.

The <SPEC_TITLE> tag creates a title for the document and the <SPECIFICATION_INFO> tag places additional information on the cover page of the document, and sets the running headings for the rest of the document, including the table of contents.

The <SPEC_TITLE> may be too long for practical use with Bookreader. You may want to use the <ONLINE_TITLE> tag and a shorter title just before the <SPEC_TITLE> tag for use by the Bookreader Library, title bar, and topic bar. This tag is only valid for Bookreader, and is ignored when you process for printed documents. Additionally, you must include the <CONTENTS_FILE> tag if processing for Bookreader. Refer to Chapter 7 for information about online documentation.

- Create major sections (such as Section 1, Scope,) in your document using the global <CHAPTER> tag.

Each <CHAPTER> tag that has paragraph text immediately following it should have a <P> tag placed between the heading and the text for correct formatting, just as in any document. Chapters that contain no pertinent information should be coded with the <CHAPTER> tag, complete with the appropriate title text and the symbol name argument. Such chapters should contain only the following standard disclaimer paragraph as specified in MIL-STD-490A.

This section is not applicable to this specification.

- Create the numbered paragraphs in each major section using the appropriate global numbered heading tags (<HEAD1>, <HEAD2>, and so on). Again, each heading tag that has paragraph text immediately following it should have a <P> tag placed between the heading and the text for correct formatting, just as in any document. The MILSPEC doctype automatically inserts a period at the end of all titled numbered headings.

To cross-reference numbered paragraphs in your document, specify them using the <REFERENCE> tag with the VALUE argument so that the default text Section does not automatically output before the paragraph number.

Refer to *VAX DOCUMENT User's Guide, Volume 1* for information about using the <REFERENCE> tag and its arguments, such as VALUE.

Using the MILSPEC Doctype

If you are processing for Bookreader, you can create hotspots to cross-reference information. Refer to Chapter 7 for information about online documentation.

6.4 Creating Data Item Description Documents

Use VAX DOCUMENT to create a Data Item Description document in accordance with either U.S. Department of Defense standard DOD-STD-2167 or DOD-STD-2167A. DOD-STD-2167A is a Department of Defense military standard published February 29, 1988 that supersedes DOD-STD-2167, published June 4, 1985.

Create a Data Item Description (DID) document in accordance with either DOD-STD-2167 or DOD-STD-2167A in any of the following ways:

- Copy the appropriate template file from the DOC\$TEMPLATES directory into your directory and edit that file (the template files contain comments to guide you in their use).
- Edit a file with LSE and expand the appropriate LSE Data Item Description template.
- Create your own file (if you create your own new file, you may still want to look at the SDML input file templates or the LSE templates as guides).

6.4.1 Creating DOD-STD-2167 Documents

To create a DOD-STD-2167 document, copy and edit one of the 24 Data Item Description (DID) templates listed in Table 6-3, or expand the appropriate LSE templates available in LSE.

Process your finished SDML input file using the MILSPEC.SECURITY or MILSPEC.ONLINE doctypes to have a document format that conforms to DOD-STD-2167.

See Section 6.5 for information on the tags available in the MILSPEC.SECURITY doctype and Chapter 7 for information on the additional tags needed for MILSPEC.ONLINE.

6.4.2 Creating DOD-STD-2167A Documents

To create a DOD-STD-2167A document, copy and edit one of the 17 Data Item Description (DID) templates listed in Table 6-4, or expand the appropriate LSE templates available in LSE.

You create DOD-STD-2167A formatted documents by using the MILSPEC.SECURITY doctype with certain tags enabled in that doctype. To create a document that follows the 2167A format specifications, enter the SDML code shown in Example 6-1 at the top of your SDML file (or include it from a separate file using either the <INCLUDE> tag or the /INCLUDE qualifier) and then process it using the MILSPEC.SECURITY doctype.

Example 6–1 Coding a 2167A-Formatted Document

```
<DOCUMENT_ATTRIBUTES>
<SET_HEADINGS> (CENTERED)
<SET_APPENDIX_ENUMERATION> (ALPHABETIC)
<ENDDOCUMENT_ATTRIBUTES>
```

To create double-spaced draft output of this file, use the same procedure, but process the file using the MILSPEC.DRAFT doctype.

6.4.2.1 Using the Data Item Description Template Files

VAX DOCUMENT contains Data Item Description (DID) template files to make creating DOD-STD-2167 and DOD-STD-2167A documents easier. See Table 6–3 for a list of the DOD-STD-2167 templates; see Table 6–4 for a list of the DOD-STD-2167A templates.

The template files supply a framework for each of the supported DID specifications, so that all you need supply is the text specific to your document. The document template provides all required tags, including section and paragraph headings.

Each template file is a collection of individual element files coded for a specific DID or military document. These files are concatenated into a single file to simplify use and storage. When you use one of these concatenated files, separate it into several files, one file for each major section of your document.

Each section of the template input file has comments with directions for its use. Placing each major section in a separate file makes it easier to maintain your document and lets you use the book-building features of VAX DOCUMENT. When you are ready to create your book, list these files in a profile file using the <ELEMENT> tag and process the profile as a VAX DOCUMENT book build.

Such a profile would appear as follows:

```
<COMMENT> (SDML profile for My Document)
<PROFILE>
<ELEMENT> (frontmatter.sdml)
<ELEMENT> (scopechap.sdml)
<ELEMENT> (secondchap.sdml)
<ELEMENT> (thirdchap.sdml)
<ELEMENT> (fourthchap.sdml)
<ELEMENT> (fifthchap.sdml)
<ELEMENT> (noteschap.sdml)
<ELEMENT> (firstapx.sdml)
<ENDPROFILE>
```

For example, if you were to use the file MILSPEC_DID_80025.SDML, you would place each of the six major sections in a separate file, and also place the front matter section and the symbol definition section in separate files. VAX DOCUMENT would place each of these files in your document as it builds your book.

For more information about creating a profile, refer to the <PROFILE> tag description in *VAX DOCUMENT User's Guide, Volume 1*.

Using the MILSPEC Doctype

To use the DID template files, do the following:

- 1 Select the template you want by referring to either Table 6–3 for DOD-STD-2167 templates or to Table 6–4 for DOD-STD-2167A templates. The names of the template files correspond to their DID document number.

If the template you want is not listed, use a similar template as the basis for creating your own template. You may want to read through the template to make sure it is the one you want.

- 2 Copy the template file you selected or created into your working directory before you modify it. Do not modify the VAX DOCUMENT templates in DOC\$TEMPLATES. Other users of VAX DOCUMENT may also want to use them.
- 3 Separate the template file into separate files, placing each major section (beginning with the <CHAPTER> tag) in a separate file. Place the front matter (beginning with the <FRONT_MATTER> tag) and the symbol definitions (created using the <DEFINE_SYMBOL> and <DEFINE_BOOK_NAME> tags) in separate files.
- 4 Modify the appropriate text arguments to the <DEFINE_SYMBOL> tags so that the proper text (such as your product's name) automatically inserts into the template when the symbol is referenced.

Each template file contains references to symbols created using the <DEFINE_SYMBOL> tag. Comments in the template file identify which symbols are used by all the templates and which symbols are used only in that particular template.

- 5 Create a book-building profile that lists each of the major sections and the front matter file as book elements. These book elements can then be built into a single book by VAX DOCUMENT. You may want to include the <CONTENTS_FILE> and <INDEX_FILE> tags in your profile to automatically include your table of contents and index files into the final book.

Do not include the file that contains the symbol definitions in the profile. Specify the symbol definitions file as an argument to the DOCUMENT /SYMBOLS qualifier when you process your profile, as shown in the following example:

```
$ DOC/LIST/CONTENTS/SYMBOLS=MY_SYMBOLS.SDML
$_MYFILE.SDML MILSPEC LN03
```

The following is a sample book-building profile for a military specification document. The <CONTENTS_FILE> and <INDEX_FILE> tags automatically include the table of contents and index files for the document.

```

<PROFILE>
<ELEMENT>(frontmatter.sdml)
<CONTENTS_FILE>
<ELEMENT>(scopechap.sdml)
<ELEMENT>(secondchap.sdml)
<ELEMENT>(thirdchap.sdml)
<ELEMENT>(fourthchap.sdml)
<ELEMENT>(fifthchap.sdml)
<ELEMENT>(noteschap.sdml)
<ELEMENT>(firstapx.sdml)
<INDEX_FILE>
<ENDPROFILE>

```

If you have a large book element or a book element that contains sections that change a great deal, you may want to separate that book element into several files. You include these separated files, called book *subelements*, into the book element file with the global `<INCLUDE>` tag. The following sample shows the book element file `thirdchap.sdml` that includes several book subelement files.

```

<COMMENT>(File: thirdchap.sdml)
<CHAPTER>(Testing Results\test_results_chap)
<INCLUDE>(testing_intro.sdml)
<INCLUDE>(test1_results.sdml)
<INCLUDE>(test2_results.sdml)
<INCLUDE>(test3_results.sdml)
<INCLUDE>(test4_results.sdml)
<INCLUDE>(test5_results.sdml)
<INCLUDE>(testing_conclusions.sdml)

```

Each of these book subelements can be processed individually and have all cross-references correctly resolved after the entire book has been built. The book-building process creates the XREF cross-reference file that the subelement accesses to resolve the cross-references. See the *VAX DOCUMENT User's Guide, Volume 1* for more information about creating and using book build profiles, symbol definitions files, and processing book subelements.

- 6 Enter the appropriate information for your document into each of the major sections of the template input file.

Table 6–3 MILSPEC Doctype DOD-STD-2167 Data Item Description Templates

Data Item Description Number	Template Description	Template Files Located in the Directory DOC\$TEMPLATES
None.	Sample template for any document conforming to MIL-STD-490A	MILSPEC_SAMPLE.SDML
DI-CMAN-80008 AMSC No. N3584	System/Segment Specification	MILSPEC_DID_80008.SDML
DI-MCCR-80009 AMSC No. N3585	Software Configuration Management Plan	MILSPEC_DID_80009.SDML
DI-MCCR-80010 AMSC No. N3586	Software Quality Evaluation Plan	MILSPEC_DID_80010.SDML
DI-MCCR-80011 AMSC No. N3587	Software Standards and Procedures Manual	MILSPEC_DID_80011.SDML

Using the MILSPEC Doctype

Table 6–3 (Cont.) MILSPEC Doctype DOD-STD-2167 Data Item Description Templates

Data Item Description Number	Template Description	Template Files Located in the Directory DOC\$TEMPLATES
DI-MCCR-80012 AMSC No. N3588	Software Top Level Design Document	MILSPEC_DID_80012.SDML
DI-MCCR-80013 AMSC No. N3589	Version Description Document	MILSPEC_DID_80013.SDML
DI-MCCR-80014 AMSC No. N3590	Software Test Plan	MILSPEC_DID_80014.SDML
DI-MCCR-80015 AMSC No. N3591	Software Test Description	MILSPEC_DID_80015.SDML
DI-MCCR-80016 AMSC No. N3592	Software Test Procedure	MILSPEC_DID_80016.SDML
DI-MCCR-80017 AMSC No. N3593	Software Test Report	MILSPEC_DID_80017.SDML
DI-MCCR-80018 AMSC No. N3594	Computer System Operator's Manual	MILSPEC_DID_80018.SDML
DI-MCCR-80019 AMSC No. N3595	Software User's Manual	MILSPEC_DID_80019.SDML
DI-MCCR-80020 AMSC No. N3596	Computer System Diagnostic Manual	MILSPEC_DID_80020.SDML
DI-MCCR-80021 AMSC No. N3597	Software Programmer's Manual	MILSPEC_DID_80021.SDML
DI-MCCR-80022 AMSC No. N3598	Firmware Support Manual	MILSPEC_DID_80022.SDML
DI-MCCR-80023 AMSC No. N3599	Operational Concept Document	MILSPEC_DID_80023.SDML
DI-MCCR-80024 AMSC No. N3600	Computer Resources Integrated Support Document	MILSPEC_DID_80024.SDML
DI-MCCR-80025 AMSC No. N3601	Software Requirements Specification	MILSPEC_DID_80025.SDML
DI-MCCR-80026 AMSC No. N3602	Interface Requirements Specification	MILSPEC_DID_80026.SDML
DI-MCCR-80027 AMSC No. N3603	Interface Design Document	MILSPEC_DID_80027.SDML
DI-MCCR-80028 AMSC No. N3604	Data Base Design Document	MILSPEC_DID_80028.SDML
DI-MCCR-80029 AMSC No. N3605	Software Product Specification	MILSPEC_DID_80029.SDML
DI-MCCR-80030 AMSC No. N3606	Software Development Plan	MILSPEC_DID_80030.SDML
DI-MCCR-80031 AMSC No. N3607	Software Detail Design Document	MILSPEC_DID_80031.SDML

Table 6-4 MILSPEC.SECURITY Doctype DOD-STD-2167A Data Item Description Templates

Data Item Description Number	Template Description	Template Files Located in the Directory DOC\$TEMPLATES
DI-CMAN-80008A AMSC No. F4328	System/Segment Specification	MILSPEC_DID_80008A.SDML
DI-CMAN-80534 AMSC No. N4329	System/Segment Design Document	MILSPEC_DID_80524.SDML
DI-MCCR-80012A AMSC No. N4330	Software Design Document	MILSPEC_DID_80012A.SDML
DI-MCCR-80013A AMSC No. N4331	Version Description Document	MILSPEC_DID_80013A.SDML
DI-MCCR-80014A AMSC No. N4332	Software Test Plan	MILSPEC_DID_80014A.SDML
DI-MCCR-80015A AMSC No. N4333	Software Test Description	MILSPEC_DID_80015A.SDML
DI-MCCR-80017A AMSC No. N4334	Software Test Report	MILSPEC_DID_80017A.SDML
DI-MCCR-80018A AMSC No. N4335	Computer System Operator's Manual	MILSPEC_DID_80018A.SDML
DI-MCCR-80019A AMSC No. N4336	Software User's Manual	MILSPEC_DID_80019A.SDML
DI-MCCR-80021A AMSC No. N4337	Software Programmer's Manual	MILSPEC_DID_80021A.SDML
DI-MCCR-80022A AMSC No. N4338	Firmware Support Manual	MILSPEC_DID_80022A.SDML
DI-MCCR-80024A AMSC No. N4339	Computer Resources Integrated Support Document	MILSPEC_DID_80024A.SDML
DI-MCCR-80025A AMSC No. N4340	Software Requirements Specification	MILSPEC_DID_80025A.SDML
DI-MCCR-80026A AMSC No. N4341	Interface Requirements Specification	MILSPEC_DID_80026A.SDML
DI-MCCR-80027A AMSC No. N4342	Interface Design Document	MILSPEC_DID_80027A.SDML
DI-MCCR-80029A AMSC No. N4343	Software Product Specification	MILSPEC_DID_80029A.SDML
DI-MCCR-80030A AMSC No. N4344	Software Development Plan	MILSPEC_DID_80030A.SDML

6.5 MILSPEC Doctype Tag Reference

This part of Chapter 6 contains reference information on all the tags available in the MILSPEC doctypes. The MILSPEC doctypes are a full implementation of the United States Military Specification Standard MIL-STD-490A.

MILSPEC Doctype Tag Reference

<CODE_EXAMPLE>

<CODE_EXAMPLE>

Places an example of code in a proportionally spaced font.

SYNTAX

<CODE_EXAMPLE>
code example text

.

.

.

<ENDCODE_EXAMPLE>

ARGUMENTS

code example text

Specifies a code fragment you want to insert into your text.

related tags

- The global <INTERACTIVE> tag
 - The global <LINE_ART> tag
 - The global <VALID_BREAK> tag
-

restrictions

Valid only in the context of the MILSPEC doctypes; all other doctypes use the global <CODE_EXAMPLE> tag, which accepts arguments and has a different format than the MILSPEC <CODE_EXAMPLE> tag.

Do not use indexing tags (<X> and <Y>) in code examples.

Do not use tab characters to format code examples. Use spaces rather than tabs.

Do not use text element tags in <CODE_EXAMPLE> (for example, <P>, <LIST>, or <NOTE>).

required terminator

<ENDCODE_EXAMPLE>

DESCRIPTION

The <CODE_EXAMPLE> tag places an example of code in a proportionally spaced font. It causes the one or more lines of code example text to be indented from the text that surrounds it.

The size of the example, whether it will be indented, and how much it will be indented from the current left margin of text is controlled by the document design.

MILSPEC Doctype Tag Reference

<CODE_EXAMPLE>

Enter the code example text between the <CODE_EXAMPLE> and <ENDCODE_EXAMPLE> tags. Character spaces and blank lines that you enter to format the code will be retained. Also, use the <ELLIPSIS> tag in this context to a vertical ellipsis to indicate you omitted some lines of code. If your code example is longer than a few lines, use the global <VALID_BREAK> tag to indicate the acceptable points for a page break.

Use the global <LINE_ART> or <INTERACTIVE> tags to create monospaced code examples in the MILSPEC doctypes.

EXAMPLE

The following example shows the coding of a proportionally spaced code example.

```
<CODE_EXAMPLE>
<KEYWORD>(type) DURATION <KEYWORD>(is) <KEYWORD>(delta)
    <EMPHASIS>(implementation_defined)
    <KEYWORD>(range) <EMPHASIS>(implementation_defined);
<ENDCODE_EXAMPLE>
```

This example produces the following output:

```
type DURATION is delta
implementation_defined range
implementation_defined;
```

MILSPEC Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

<DOCUMENT_ATTRIBUTES>

Enables doctype-specific tags that override the default design format of the MILSPEC doctypes.

SYNTAX

<DOCUMENT_ATTRIBUTES>

related tags

- <RUNNING_FEET>
 - <RUNNING_TITLE>
-

required terminator

<ENDDOCUMENT_ATTRIBUTES>

DESCRIPTION

The <DOCUMENT_ATTRIBUTES> tag enables doctype-specific tags that override the default design format of the MILSPEC doctypes. The <DOCUMENT_ATTRIBUTES> tag enables a group of tags that let you modify the default format of these doctypes. The default format of these doctypes is:

- Appendixes are numbered by Roman numerals and Arabic numerals.
- Formal tables, figures, and examples are numbered using the current section (paragraph) number.
- Headings are placed, alternately, on the right and left pages.
- Running footers (running headings at the bottom of the page) are placed, alternately, on the right and left pages.

The tags that let you modify the default format are recognized only in the context of the <DOCUMENT_ATTRIBUTES> tag. If you use other SDML tags in this context, they are ignored, as if they had occurred in the context of a <COMMENT> tag.

Use the <DOCUMENT_ATTRIBUTES> tag at the beginning of an input file (or in a file specified using the DOCUMENT /INCLUDE qualifier) to alter the default format of a doctype for the processing of that entire file.

Table 6-5 summarizes the formatting tags enabled by the <DOCUMENT_ATTRIBUTES> tag in these doctypes.

MILSPEC Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

Table 6–5 Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag

Formatting Tags	Description
Tags Enabled in the MILSPEC.SECURITY and MILSPEC.DRAFT Doctypes	
<INDENT_FIRST_LIST>(TRUE) <INDENT_FIRST_LIST>(FALSE)	<p>Specifies whether to indent lists that are not nested in other lists.</p> <ul style="list-style-type: none"> • The TRUE keyword causes lists and their list item designators (such as numbers or bullets) to be indented from the current left text margin. • The FALSE keyword causes lists and their list item designators to be placed flush with the current left text margin. <p>The default keyword is FALSE.</p>
<SET_APPENDIX_ENUMERATION>(ALPHABETIC) <SET_APPENDIX_ENUMERATION>(ROMAN)	<p>Specifies whether appendixes are identified using letters or Roman numerals. By default in the MILSPEC doctypes, appendixes are assigned Roman numerals and Arabic numerals. If you specify <SET_APPENDIX_ENUMERATION>(ALPHABETIC), then the appendixes will be identified using letters.</p>
<SET_FORMAL_ELEMENT_NUMBERING>(BY_SECTION) <SET_FORMAL_ELEMENT_NUMBERING>(SEQUENTIAL)	<p>Specifies how formal tables, figures, and examples are to be numbered in a document processed using the MILSPEC doctypes. By default, formal tables, figures, and examples are numbered in this doctype using the current section (paragraph) number.</p> <p>The SEQUENTIAL keyword indicates that formal elements are to be numbered sequentially throughout the document; for example, the eighth formal table in the document, regardless of what chapter it occurs in, is numbered Table 8.</p>
<SET_HEADERS>(RIGHT) <SET_HEADERS>(LEFT) <SET_HEADERS>(CENTERED) <SET_HEADERS>(CYCLE)	<p>Specifies the positioning of running headings near the top of the page.</p> <ul style="list-style-type: none"> • The RIGHT keyword causes headings to be placed on the right-hand side of the page. • The LEFT keyword causes headings to be placed on the left-hand side of the page. • The CENTERED keyword causes headings to be centered on the page. • The CYCLE keyword indicates that headings should appear on the right-hand side when the page number is odd and on the left-hand side of the page when the page number is even. <p>The default keyword is CYCLE.</p>

MILSPEC Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

Table 6–5 (Cont.) Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag

Formatting Tags	Description
Tags Enabled in the MILSPEC.SECURITY and MILSPEC.DRAFT Doctypes	
<SET_FOOTERS>(RIGHT) <SET_FOOTERS>(LEFT) <SET_FOOTERS>(CYCLE)	<p>Specifies the positioning of running feet near the bottom of the page.</p> <ul style="list-style-type: none">• The RIGHT keyword causes the running footer to be placed on the right-hand side of the page.• The LEFT keyword causes the running footer to be placed on the left-hand side of the page.• The CYCLE keyword causes the running footer to be placed on the right-hand side when the page is even and on the left-hand side of the page when the page number is odd. <p>The default keyword is CYCLE.</p>

EXAMPLE

The following example shows how to use the <DOCUMENT_ATTRIBUTES> tag to create centered headings and to cause appendixes to be ordered using letters.

```
<DOCUMENT_ATTRIBUTES>  
<SET_HEADINGS> (CENTERED)  
<SET_APPENDIX_ENUMERATION> (ALPHABETIC)  
<ENDDOCUMENT_ATTRIBUTES>
```

<HEADn>

Marks a heading of the level specified (1 through 20).

SYNTAX

<HEADn>(heading text [\ symbol name])

ARGUMENTS

heading text

Specifies the text of the heading. If the book design you are using outputs headings that are all capital letters, those headings capitalize regardless of the way you enter them in your input file. However, use uppercase and lowercase letters, according to your local conventions, to obtain the proper capitalization of the heading in the table of contents and in cross-references.

symbol name

This is an optional argument. It specifies the name of the symbol used in all references to this heading.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- The global <CHEAD> tag
- The global <SUBHEAD1> tag
- The global <SUBHEAD2> tag

DESCRIPTION

The <HEADn> tag marks a heading of the level specified (1 through 20). Each of the 20 tags (<HEAD1> through <HEAD20>) does the following:

- Outputs the heading text specified in its first argument
- Automatically numbers the heading
- Resets all lower heading levels (if any)
- Specifies the symbol name with which cross-references to that heading should be made
- Automatically places an entry for the heading in the table of contents

The proper choice of the heading level depends on an understanding of the logical structure of the document you are writing. A <HEAD2> tag will always be logically subordinate to a <HEAD1> tag. The same is true for the relationship between <HEAD3> and <HEAD2>, <HEAD4> and <HEAD3>, and so on.

MILSPEC Doctype Tag Reference

<HEADn>

EXAMPLES

The following tag creates a fourth-level heading.

1 <HEAD4>(SET and SHOW Tasks\set_show_sec)

The following tag creates an eighth-level heading.

2 <HEAD8>(Other Tasks\other_sec)

An example of a heading created by the <HEAD4> tag would be 1.1.1.1 Set. If the heading number created by the tag was 1.1.1.1, then a reference to \set_show_sec would output as Section 1.1.1.1.

Use the <REFERENCE> tag as shown in the following table to modify the output of the cross-reference.

Reference	Output
<REFERENCE>(set_show_sec)	Section 1.1.1.1
<REFERENCE>(set_show_sec\value)	1.1.1.1
<REFERENCE>(set_show_sec\text)	Set and Show Tasks
<REFERENCE>(set_show_sec\full)	Section 1.1.1.1, Set and Show Tasks

<HIGHEST_SECURITY_CLASS>

Prints the classification text associated with the highest security classification encountered in a document.

SYNTAX

<HIGHEST_SECURITY_CLASS>

related tags

- <SECURITY>
- <SET_PAGE_SECURITY>

DESCRIPTION

The <HIGHEST_SECURITY_CLASS> tag prints the classification text associated with the highest security classification encountered in a document. When processing a document, VAX DOCUMENT keeps track of the security levels associated with each <SECURITY> tag you specify. VAX DOCUMENT saves the highest of these security levels and you can place it anywhere in your document using the <HIGHEST_SECURITY_CLASS> tag.

For example, to place the highest security level found in a document on the title page as a subtitle, you would enter the <HIGHEST_SECURITY_CLASS> tag as an argument to the <SUBTITLE> tag.

EXAMPLE

The following example shows how to use the <HIGHEST_SECURITY_CLASS> tag to place the highest security classification found in a document on the title page as a subtitle.

```
<TITLE_PAGE>  
<SPECIFICATION_INFO>(WS-99999\PAGM NNNN D036\<DATE>)  
<SPEC_TITLE>(Program Design Specification  
\For The  
\Machinery Control Program (MCP))  
<subtitle>(<HIGHEST_SECURITY_CLASS>)
```

MILSPEC Doctype Tag Reference

<RUNNING_FEET>

<RUNNING_FEET>

Creates a single-line running footer at the bottom of each page, next to the page number.

SYNTAX

<RUNNING_FEET>(*footer text*)

ARGUMENTS

footer text

Specifies the text of the running footer. The footer appears next to the page number on the bottom of each page.

related tags

- <SET_FOOTERS>
-

DESCRIPTION

The <RUNNING_FEET> tag creates a single-line running footer at the bottom of each page, next to the page number.

Use the <SET_FOOTERS> tag, enabled by the <DOCUMENT_ATTRIBUTES> tag, to alter the position of the page number and title text in the running footer.

This tag is ignored for Bookreader output.

EXAMPLE

The following example creates the running footer, Special Report, at the bottom of the page.

```
<RUNNING_FEET> (Special Report)
```


MILSPEC Doctype Tag Reference

<SECURITY>

<SECURITY>

Assigns a security classification level to a text element.

SYNTAX

<SECURITY> (*classification keyword*)

ARGUMENTS

classification keyword

Is a keyword indicating the classification level. Each keyword has associated text output with a text element, as well as text printed at the top and bottom of each page. The following table shows default classification levels and the keywords associated with them.

Classification Keyword	Abbreviated Text Output	Classification Text for Running Headers and Footers
TOP_SECRET	(TS)	TOP SECRET
SECRET	(S)	SECRET
CONFIDENTIAL	(C)	CONFIDENTIAL
UNCLASSIFIED	(U)	UNCLASSIFIED

You modify classification keywords and associated output text with the <SET_SECURITY_CLASS> tag.

related tags

- <HIGHEST_SECURITY_CLASS>
- <SET_CONTENTS_SECURITY>
- <SET_PAGE_SECURITY>
- <SET_SECURITY_CLASS>

required terminator

<ENDSECURITY>

DESCRIPTION

The <SECURITY> tag assigns a security classification level to a text element. You can apply a security classification to one of the following text elements:

- <CHAPTER> and <APPENDIX> tags

The classification applies to the title of the chapter or appendix.

- <HEAD_n> tags

The classification applies to the heading. If a classification is established before a heading, that classification can be overridden in the heading text argument.

MILSPEC Doctype Tag Reference

<SECURITY>

- <P> tags

The classification of a paragraph created using the <P> tag outputs immediately before the text of the paragraph.

- <TABLE> and <FIGURE> tags.

If a classification is in effect when VAX DOCUMENT processes a <FIGURE> or <TABLE> tag sequence, the classification applies to the entire figure or table, and is placed in the caption. Specify the <SECURITY> tag inside the caption argument to <FIGURE> or <TABLE> tag to apply a security to the caption only. Note that if you place a security classification in the caption of a multi-page figure or table, all pages of that figure or table will have that security applied.

- <TABLE_ROW> tag,

You specify <SECURITY> at the beginning of text you want to classify. Otherwise, security classifications are not automatically applied to table row text or to paragraph text, marked by <P> tags, in table rows.

The following example illustrates how to mark security codes on text elements in a table:

```
<SECURITY>(CONFIDENTIAL)
<TABLE>(Tools Needed to Install the MXXK Internal Circuitry\tools_tab)
<TABLE_SETUP>(2\15)
<TABLE_HEADS>(Tool\Part Number)
<TABLE_ROW>(Voltmeter\RQ-3341-2)
<TABLE_ROW>(<SECURITY>(SECRET)3-Stage Neutralizer<ENDSECURITY>
\The first neutralizer stage uses part number RQ-3321-1.
<SECURITY>(TOP_SECRET)
<P>
Neutralizer stages 2 and 3 use experimental enhanced RQ-3321-1
parts, RXX-3321-A and RXX-3321-B.
<ENDSECURITY>)

<TABLE_ROW>(Screwdriver\RQ-1221-1)
<ENDTABLE>
<ENDSECURITY>
```

When you specify a <SECURITY> tag, the specified security classification is in effect for all subsequent text elements (as described in the preceding list). The abbreviated form of the classification appears in the output text, on the page on which the text is output.

When the text formatter finishes formatting each page of output, it determines the highest classification level applied to any text on that page, and it prints that classification on the top and bottom of the page.

A specific classification stays in effect until you do one of the following:

- Specify the <SECURITY> tag with another classification.
- Specify the <ENDSECURITY> tag, indicating that there is no more classified material.

You can nest <SECURITY> tags to any level, but all levels must be terminated using <ENDSECURITY> tags by the end of the file.

MILSPEC Doctype Tag Reference

<SECURITY>

If you use a <SECURITY> tag anywhere in a document, every page of the document is marked with a security classification. For example, if you assign only one chapter in a multiple-chapter document the Top Secret classification, each page of that chapter is marked Top Secret, and all other pages in the document are automatically marked Unclassified, to indicate that there is no specific classification attached to text on those pages.

EXAMPLE

In the following example, the security applied by the <SECURITY> tag affects the <CHAPTER>, <P>, and <HEAD1> tags shown.

```
<SECURITY>(TOP_SECRET)
<CHAPTER>(MXXK-5 Internal Circuitry)
<p>This document describes the internal circuitry of the MXXK-5.
<head1>(SPECIFICATIONS\specs)
<P>
The following information specifies the internal circuitry of the MXXK-5 unit.
.
.
.
<ENDSECURITY>
```

<SET_APPENDIX_NUMBER>

Overrides the default appendix Roman numeral VAX DOCUMENT assigns to an appendix.

SYNTAX <SET_APPENDIX_NUMBER> (*Roman numeral integer*)

ARGUMENTS *Roman numeral integer*
Specifies an integer equivalent to the Roman numeral for the appendix; for example, the integer 3 is equivalent to the Roman numeral III. This argument must be an integer greater than zero.

related tags

- The global <APPENDIX> tag
- The global <SET_APPENDIX_LETTER> tag
- The global <SET_CHAPTER_NUMBER> tag

restrictions

This tag should not be used in a file that uses a book build profile or that uses the /PROFILE qualifier during processing. If either of these conditions occurs, VAX DOCUMENT issues a warning message and ignores the <SET_APPENDIX_NUMBER> tag for the book build.

DESCRIPTION The <SET_APPENDIX_NUMBER> tag overrides the default appendix Roman numeral VAX DOCUMENT assigns to an appendix.

The <SET_APPENDIX_NUMBER> tag resets the current appendix Roman numeral to the integer you specify as the *Roman numeral integer*. For example, if you specified 4 as the Roman numeral integer, VAX DOCUMENT would set the appendix number to IV.

Place the <SET_APPENDIX_NUMBER> tag in your SDML file before the <APPENDIX> tags you want it to affect. The new appendix number you specify resets the numbering for all following appendixes. For example, if you set the appendix number to 2, the next appendix will be numbered II, the appendix following that appendix will be numbered III, and so on.

The <SET_APPENDIX_NUMBER> can be used multiple times in an SDML file.

MILSPEC Doctype Tag Reference

<SET_APPENDIX_NUMBER>

EXAMPLE

In the following example, the <SET_APPENDIX_NUMBER> tag explicitly sets the appendix to 4. This causes any subsequent appendixes to be numbered beginning with the Roman numeral IV.

```
<SET_APPENDIX_NUMBER>(4)
<APPENDIX>(Run-time Functions\functions_ap)
<p>
The following functions are used at run time:
.
.
.
<APPENDIX>(Run-time Messages\messages)
<p>
The following messages may occur at run time:
.
.
.
```

<SET_CONTENTS_SECURITY>

The <SET_CONTENTS_SECURITY> tag assigns a security classification level to a text element. VAX DOCUMENT applies this security classification to all pages in the table of contents and index, overriding the default security classification.

SYNTAX

<SET_CONTENTS_SECURITY> (*classification keyword*)

ARGUMENTS

classification keyword

Specifies a keyword indicating the classification level. Each keyword has associated text displayed in conjunction with a text element, as well as text printed at the top and bottom of each page. The default classification levels and the classification keywords associated with them are:

Classification Keyword	Abbreviated Text Output	Classification Text for Running Headers and Footers
TOP_SECRET	(TS)	TOP SECRET
SECRET	(S)	SECRET
CONFIDENTIAL	(C)	CONFIDENTIAL
UNCLASSIFIED	(U)	UNCLASSIFIED

Classification keywords and associated output text can be modified with the <SET_SECURITY_CLASS> tag. Any keywords defined using <SET_SECURITY_CLASS> are valid for <SET_CONTENTS_SECURITY>.

related tags

- <SECURITY>
- <SET_CONTENTS_SECURITY>
- <SET_SECURITY_CLASS>

DESCRIPTION

The <SET_CONTENTS_SECURITY> tag assigns a security classification level to a text element. By default, when security classification marking is in effect, the table of contents and index are assigned the highest classification level assigned to the entire document. For the contents and index, use the <SET_CONTENTS_SECURITY> tag to override the security classification that would be applied by default.

MILSPEC Doctype Tag Reference

<SET_CONTENTS_SECURITY>

EXAMPLE

In the following example, the <SET_CONTENTS_SECURITY> tag specifies that the pages in the table of contents and index are to be assigned the UNCLASSIFIED security classification, regardless of the classification of any of the other pages in the document.

```
<SET_CONTENTS_SECURITY>(UNCLASSIFIED)  
<SECURITY>(SECRET)
```

```
.  
. .  
.
```

<SET_PAGE_SECURITY>

Specifies that a security classification be applied to a page to override the default security classification.

SYNTAX

<SET_PAGE_SECURITY> (*classification keyword*)

ARGUMENTS

classification keyword

Specifies a keyword indicating the security classification level. Each keyword has associated text output in conjunction with a text element, as well as text printed at the top and bottom of each page. The following table shows the default classification levels and the keywords associated with them:

Classification Keyword	Abbreviated Text Output	Classification Text for Running Headers and Footers
TOP_SECRET	(TS)	TOP SECRET
SECRET	(S)	SECRET
CONFIDENTIAL	(C)	CONFIDENTIAL
UNCLASSIFIED	(U)	UNCLASSIFIED

related tags

- <SECURITY>
- <SET_CONTENTS_SECURITY>
- <SET_SECURITY_CLASS>

DESCRIPTION

The <SET_PAGE_SECURITY> tag specifies that a security classification be applied to a page to override the default security classification.

Keywords defined using <SET_SECURITY_CLASS> are valid as keywords for the <SET_PAGE_SECURITY> and <SECURITY> tags.

MILSPEC Doctype Tag Reference

<SET_PAGE_SECURITY>

EXAMPLE

In the following example, the <SET_PAGE_SECURITY> tag specifies that the current page is to be assigned the UNCLASSIFIED classification, overriding the highest default classification (SECRET) for that page.

```
<SECURITY>(TOP_SECRET)
<HEAD1>(Installing the MKXX-5 circuit\installing_mkxx_5)
<P>Installation of the MKXX-5 circuit requires 24 steps and the following tools:
<list>(unnumbered)
<le> screwdriver
<le> voltmeter
<endlist>
<ENDSECURITY>
<SECURITY>(SECRET)
<HEAD2>(Beginning the installation\begin_install)
<p>Begin the installation by removing the MKXX-5 circuit from the
shipping materials.
<SET_PAGE_SECURITY>(UNCLASSIFIED)
```

<SET_SECURITY_CLASS>

Establishes a new security classification or overrides the default text associated with an existing classification.

SYNTAX <SET_SECURITY_CLASS >(classification keyword \ id text \ classification text \ priority)

ARGUMENTS

classification keyword

Specifies the keyword to be specified to the <SECURITY> tag to generate the new or modified id text, classification text, or priority. An example might be the keyword TOP_SECRET. User-created keywords may also be used.

id text

Specifies the character string that will be placed in text and heading numbers and figure, example, and table captions, when the security classification outputs. An example might be the text TS for a top secret security classification. User-created text strings may also be used.

classification text

Specifies the text that appears at the top and bottom of the page when the security classifications output. An example would be the text TOP SECRET for a top secret security classification. Other user-created text strings may also be used.

priority

Specifies the numeric priority assigned to the classification, where priority must be an integer in the range 1 to 6, where 6 indicates the highest priority. The following table shows the priorities assigned to the default security classifications:

Class	Priority
TOP_SECRET	4
SECRET	3
CONFIDENTIAL	2
UNCLASSIFIED	1

Priorities 5 and 6 are unassigned by default.

related tags

- <HIGHEST_SECURITY_CLASS>
- <SECURITY>
- <SET_CONTENTS_SECURITY>
- <SET_PAGE_SECURITY>

MILSPEC Doctype Tag Reference

<SET_SECURITY_CLASS>

DESCRIPTION

The <SET_SECURITY_CLASS> tag establishes a new security classification or overrides the default text associated with an existing classification. Use the tag to do the following:

- Modify the text printed in the running header or footer for a given security classification (the classification text argument).
- Modify the code output in the text when a text element is labeled with a security classification (the id text argument).
- Modify the priority of a security classification (the priority argument).
- Create one or two additional security classes and specify the associated text.

The <SET_SECURITY_CLASS> tag requires all four arguments. If you do not want to specify an argument, leave it blank as a null argument.

The <SET_SECURITY_CLASS> tag establishes keywords and text strings to be output on a page for the <SECURITY>, <HIGHEST_SECURITY_CLASS>, <SET_CONTENTS_SECURITY>, and <SET_PAGE_SECURITY> tags.

EXAMPLE

In the following example, the <SET_SECURITY_CLASS> tag modifies the id text associated with the keyword CONFIDENTIAL. When you specify a subsequent <SECURITY>(CONFIDENTIAL) tag, no id text is placed in text or headings, since that argument was omitted when you specified the <SET_SECURITY_CLASS> tag. The running header and footer lines will carry the classification text, WIDGET Company Confidential.

The CONFIDENTIAL keyword has the default priority of 2, as specified in the final argument to the <SET_SECURITY_CLASS> tag.

```
<SET_SECURITY_CLASS>(CONFIDENTIAL\\WIDGET Company Confidential\2)
```

<SIGNATURE_LINE>

Creates up to two rules on a line and places a name below each rule; each rule serves as a signatory line for the person listed below it.

SYNTAX <SIGNATURE_LINE> ([*name-1*][\ *name-2*])

ARGUMENTS *name-n*
This is an optional argument. Specifies the name or title of the person who is to sign on the previous line. These names and lines output in two columns.

-
- related tags**
- <SIGNATURE_LIST>
 - <SPECIFICATION_INFO>
 - <SPEC_TITLE>
 - <SUBTITLE>
 - The global <FRONT_MATTER> tag
 - The global <TITLE_PAGE> tag

restrictions Valid only in the context of a <SIGNATURE_LIST> tag.

DESCRIPTION The <SIGNATURE_LINE> tag creates up to two rules on a line and places a name below each rule; each rule serves as a signatory line for the person listed below it. The tag creates signature lines within a signature list that has a 2-column format; if you omit a name in either column, no name (or rule) outputs in that column.

EXAMPLE The following example shows a signature list with two names in the first row of signatures, a single name in the left column of the second row, and a name in the right column of the third row.

```
<TITLE_PAGE>
.
.
.
<SIGNATURE_LIST>
<SIGNATURE_LINE>(Otto Baloo\T. C. Leeds)
<SIGNATURE_LINE>(Ted Doe\ )
<SIGNATURE_LINE>( \Eunice Smith)
<ENDSIGNATURE_LIST>
```

MILSPEC Doctype Tag Reference

<SIGNATURE_LIST>

<SIGNATURE_LIST>

Begins a 2-column listing of signature lines on the title page of a document and supplies headings for each of those columns.

SYNTAX **<SIGNATURE_LIST>**(*col heading-1 \ col heading-2*)

ARGUMENTS ***col heading-n***
Specifies the heading for the column of signatures. These headings are required.

related tags

- <SIGNATURE_LINE>
- <SPECIFICATION_INFO>
- <SPEC_TITLE>
- <SUBTITLE>
- The global <TITLE_PAGE> tag
- The global <FRONT_MATTER> tag

required terminator <ENDSIGNATURE_LIST>

restrictions Valid only in the context of a global <TITLE_PAGE> tag.

DESCRIPTION The <SIGNATURE_LIST> tag begins a 2-column listing of signature lines on the title page of a document and supplies headings for each of those columns. Each argument places a heading over one of the signature line columns.

This tag enables the <SIGNATURE_LINE> tag to specify each signature line and the name associated with it. You can use as many <SIGNATURE_LINE> tags as you want in the context of the <SIGNATURE_LIST> tag.

MILSPEC Doctype Tag Reference

<SIGNATURE_LIST>

EXAMPLE

The following example shows a sample use of the <SIGNATURE_LIST> tag. In the example, the <SIGNATURE_LIST> tag is used in the context of the global <TITLE_PAGE> tag.

```
<FRONT_MATTER>
<TITLE_PAGE>
.
.
<SUBTITLE>(Submitted Under\Contract A00000--11--A--2222)
<SIGNATURE_LIST>(Authenticated by:\Approved by:)
<SIGNATURE_LINE>(Procuring Activity\Program Manager)
<SIGNATURE_LINE>(Date\Technical Director)
<SIGNATURE_LINE>(\Consultant)
<ENDSIGNATURE_LIST>
.
.
<ENDTITLE_PAGE>
```

MILSPEC Doctype Tag Reference

<SPECIFICATION_INFO>

<SPECIFICATION_INFO>

Creates a listing of information about the specification on the title page and creates a 2-line running heading for the rest of the document.

SYNTAX **<SPECIFICATION_INFO >**(*specification number*
 | *code id number*
 | *specification date*
 [| *additional info*])

ARGUMENTS

specification number

Specifies the number associated with this document. This number formats as the first line in a block of lines in the upper right of the title page. This number also carries as the top line in a 2-line running heading throughout the document.

The running title established by the specification number argument can be overridden in the MILSPEC.SECURITY and MILSPEC.DRAFT doctypes by the text output by the <RUNNING_TITLE> tag.

code id number

Specifies the identification code number for this document. This number formats as the second line in a block of lines in the upper right of the title page.

specification date

Specifies a date for the document. This value can be specified as a text string (for example, 31 October 1986), or it can be specified using the global <DATE> tag, which produces the date at the time the file processes.

This date formats as the third line in a block of lines in the upper right of the title page; it also carries as the bottom line in a 2-line running heading throughout the document.

The running title established by the specification date argument can be overridden in the MILSPEC.SECURITY and MILSPEC.DRAFT doctypes by the running title created by the <RUNNING_TITLE> tag.

additional info

This is an optional tag. Specifies additional information about the document, for example, Part 1 of 3 parts.

This information formats as the fourth line in a block of lines in the upper right of the title page.

related tags

- <RUNNING_TITLE>
- <SIGNATURE_LIST>
- <SIGNATURE_LINE>
- <SPEC_TITLE>

MILSPEC Doctype Tag Reference

<SPECIFICATION_INFO>

- <SUBTITLE>
- The global <DATE> tag
- The global <TITLE_PAGE> tag
- The global <FRONT_MATTER> tag

restrictions

Valid only in the context of a global <TITLE_PAGE> tag.

DESCRIPTION

The <SPECIFICATION_INFO> tag creates a listing of information about the specification on the title page and creates a 2-line running heading for the rest of the document.

The list of information formats on the upper right of the title page. Each line specifies a particular argument as follows:

- Line one lists the *specification number* argument.
- Line two lists the *code id number* argument.
- Line three lists the *specification date* argument.
- Line four is optional and lists the *additional info* argument.

Additionally, the *specification number* and *specification date* arguments output throughout the document as the top and bottom lines, respectively, of a 2-line running heading.

EXAMPLE

The following example shows a sample use of the <SPECIFICATION_INFO> tag.

```
<FRONT_MATTER>
<TITLE_PAGE>
<SPECIFICATION_INFO>(12345B\al42-b4\July 4, 1776\Part I of Three Parts)
<SPEC_TITLE>(Prime Item Development Specification\
For the \ (Approved Title)\of the \Supported \Device)
.
.
<ENDTITLE_PAGE>
```

MILSPEC Doctype Tag Reference

<SPEC_TITLE>

<SPEC_TITLE>

Creates a title with up to seven centered lines on the title page.

SYNTAX

<SPEC_TITLE >*(title text-1 [\ title text-2 . . .
[\ title text-7]])*

ARGUMENTS

title text-n

Specifies text in a title. Each argument centers on a separate line on the title page.

related tags

- <ONLINE_TITLE>
- <SIGNATURE_LINE>
- <SIGNATURE_LIST>
- <SPECIFICATION_INFO>
- <SUBTITLE>
- The global <TITLE_PAGE> tag
- The global <FRONT_MATTER> tag

restrictions

Valid only in the context of a global <TITLE_PAGE> tag.

DESCRIPTION

The <SPEC_TITLE> tag creates a title with up to seven centered lines on the title page. Each title line centers on the page beneath the previous title line.

Use the <SUBTITLE> tag to create a subordinate title for a military specification.

Use the <ONLINE_TITLE> tag to supply an abbreviated title that will be used for the Bookreader library, title bar, and topic bar only. The <ONLINE_TITLE> tag is ignored in printed documents.

EXAMPLE

The following example shows a sample use of the <SPEC_TITLE> tag.

```
<FRONT_MATTER>
<TITLE_PAGE>
<SPECIFICATION_INFO>(12345B\a142-b4\July 4, 1776\Part I of Three Parts)
<SPEC_TITLE>(Prime Item Development Specification\
For the \ (Approved Title)\of
the \Device)
.
.
<ENDTITLE_PAGE>
```

<SUBTITLE>

Creates a subtitle with up to seven centered lines on the title page.

SYNTAX <SUBTITLE>(title text-1 [\ title text-2 . . . \ title text-7])

ARGUMENTS *title text-n*
Specifies a text line in a subtitle. You can specify one to seven arguments. Each argument centers on a separate subtitle line.

related tags

- <SIGNATURE_LIST>
- <SIGNATURE_LINE>
- <SPECIFICATION_INFO>
- <SPEC_TITLE>
- The global <TITLE_PAGE> tag
- The global <FRONT_MATTER> tag

restrictions Valid only in the context of a global <TITLE_PAGE> tag.

DESCRIPTION The <SUBTITLE> tag creates a subtitle with up to seven centered lines on the title page. Each title line centers on the page beneath the previous title line.

Use the <SPEC_TITLE> tag to create a main title for a military specification.

EXAMPLE The following example shows a sample use of the <SUBTITLE> tag.

```
<FRONT_MATTER>
<TITLE_PAGE>
<SPECIFICATION_INFO>(12345B\al42-b4\July 4, 1776\Part I of Three Parts)
<SPEC_TITLE>(Prime Item Development Specification\
For the \ (Approved Title) \ of the \ Device)
<SUBTITLE>(Submitted Under\Contract A00000--11--A--2222)
.
.
.
<ENDTITLE_PAGE>
```


7

Using the ONLINE Doctype

The ONLINE doctype is for creating **Bookreader** documentation.

The ONLINE doctype has a dynamic design, in that it produces documentation that you see, not as printed copy, but as a screen display when you use Bookreader. In general, use the ONLINE **keyword** to process your documents for Bookreader. If you are creating an online manual, military specification, or software reference document, use MANUAL.ONLINE, MILSPEC.ONLINE, or SOFTWARE.ONLINE, respectively. For any of these doctypes, use the same tags described in this chapter. All of the online designs produce documentation in the default style of the specific doctype. For example, SOFTWARE.ONLINE produces documentation that looks like SOFTWARE.REFERENCE.

All online designs display an online document in a 5.9 × 6.6-inch format with numbered headings and ragged right margin. These designs are solely for online display with Bookreader.

Online tags only function if you use Bookreader. Their presence in your documents is ignored when you process printed outputs. However, they are required to produce a Bookreader book.

Coding for online books does not differ much from coding for printed books, but presenting a book online creates a new set of visual considerations. Whether preparing existing SDML files for Bookreader or creating new SDML files, one of your primary concerns is the visual aspect of the finished book. You need to think about how the book will look when viewed with Bookreader.

For these reasons, the VAX DOCUMENT documentation set includes *VAX DOCUMENT Producing Online and Printed Documentation*. That guide is specifically written to help you prepare documents for both printed and online versions.

7.1 ONLINE Doctype Tag Reference

This part of Chapter 7 contains reference information on all the tags available in the ONLINE doctype.

ONLINE Doctype Tag Reference

<BOOK_ONLY>

<BOOK_ONLY>

Labels portions of a template that should not appear in the online help file.

SYNTAX

<BOOK_ONLY>

ARGUMENTS

None.

related tags

- <HELP_ONLY>

restrictions

Used only in a reference template used to generate a help file.

required terminator

<ENDBOOK_ONLY>

DESCRIPTION

The <BOOK_ONLY> tag labels portions of a template that should not appear in the online help file. This tag affects only processing associated with the creation of a help file and does not affect the formatting of the text.

EXAMPLE

The following example shows how to use the <BOOK_ONLY> tag.

```
<OVERVIEW>
```

```
Invokes the Librarian Utility to create, modify, or describe  
an object, macro, help, text, or shareable image library.
```

```
<book_only>
```

```
For a complete description of the Librarian Utility, including  
information about the LIBRARY command and its qualifiers,  
see the <tag>(reference)<(VMS_UTILITIES_REF).</tag>
```

```
<endbook_only
```

```
<tag>(endoverview)>
```

This example shows how to use the <BOOK_ONLY> tag. The presence of the tag has no effect on the processing of the text unless you are producing a help file. If you are producing a help file, the text between the <BOOK_ONLY> and <ENDBOOK_ONLY> tags is deleted. When processed as a text file, this example produces the following output:

```
Invokes the Librarian Utility to create, modify, or describe an object, macro,  
help, text, or shareable image library.
```

```
For a complete description of the Librarian Utility, including information about  
the LIBRARY command and its qualifiers, see the VMS_UTILITIES_REF.
```

<BOOK_REF>

Outputs a BOOK entry for a Bookreader bookshelf file.

SYNTAX <BOOK_REF>(symbol name \ file spec)

ARGUMENTS *symbol name*

Specifies the symbol that is associated with the title of the book.

Symbol names must not exceed 31 characters and must only include alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

Define this symbol in either your source file or a symbol definition file (using the <DEFINE_BOOK_NAME> tag). If you define it in a symbol definition file, include it using the /SYMBOLS qualifier on the DOCUMENT command line.

file spec

Specifies the book file specification that points to the *book_filename.DECW\$BOOK* file. The file type is not required; the default of *.DECW\$BOOK* is assumed.

related tags

- <SHELF_CREATE>
- <SHELF_REF>

restrictions

Valid only in the context of the <SHELF_CREATE> tag.

DESCRIPTION

The <BOOK_REF> tag outputs a BOOK entry for a Bookreader bookshelf file. This entry is a pointer to an existing book file. Do not use the directory or file type information unless you are including books or shelves outside the directory where you are building the customized bookshelf file. This tag has no effect for printed output. See *VAX DOCUMENT Producing Online and Printed Documentation* for more information on building bookshelves.

EXAMPLE

See the example in the discussion of the <SHELF_CREATE> tag.

ONLINE Doctype Tag Reference

<EXTENSION>

2 <EXTENSION>
<P>By using certain tags,
you can highlight an entire paragraph. This feature is
useful for emphasizing a particular piece of information
in an online book.
<ENDEXTENSION>

This example of text produces the following online output:

By using certain tags, you can highlight an entire paragraph. This feature is useful for emphasizing a particular piece of information in an online book.

Be sure to code the <P> tag inside the <EXTENSION> tag or the first character in your paragraph is indented one space.

ONLINE Doctype Tag Reference

<HOTSPOT>

<HOTSPOT>

Changes any text into a hotspot for Bookreader documentation. For printed documentation, this tag generates the text you specify followed by a reference (in parentheses) to the section, chapter, or table, and so on.

SYNTAX

<HOTSPOT>(*symbol name* [*\ text*])

ARGUMENTS

symbol name

Specifies the name of a symbol assigned in a text element tag (for example, <HEADn> or <TABLE>).

The following tags require a symbol name for a book you create for Bookreader:

- <APPENDIX>
- <CHAPTER>
- <CHEAD>
- <EXAMPLE>
- <FIGURE>
- <HEAD>
- <HEADn>
- <PREFACE>
- <PREFACE_SECTION>
- <SUBHEADn>
- <TABLE>
- <COMMAND>
- <ROUTINE>
- <SDML_TAG>
- <SET_TEMPLATE_COMMAND>
- <SET_TEMPLATE_ROUTINE>
- <SET_TEMPLATE_STATEMENT>
- <SET_TEMPLATE_TAG>
- <STATEMENT>
- <SUBCOMMAND>

The first 11 tags are listed in *VAX DOCUMENT Using Global Tags*.

The following tags, listed in *VAX DOCUMENT Using Global Tags*, accept a symbol name, but do not require one, for both printed books and books you create for Bookreader:

- <FRONT_MATTER>
- <GLOSSARY>
- <PART>
- <MATH>

The following tags accept a symbol name for a printed book; the tags are ignored for books you create for Bookreader:

- <REF_NOTE>
- <SECTION>

Symbol names must not exceed 31 characters and must only include alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

text

This is an optional argument. It specifies the text you want to use as the hotspot for Bookreader output. If you do not use this argument, the title of the Bookreader topic you are referring to is used as the hotspot.

For printed output, the text you specify is generated followed by a reference (in parentheses) to the section, chapter, or table, and so on.

related tags

- <REFERENCE>
- <SET_ONLINE_TOPIC>

DESCRIPTION

The <HOTSPOT> tag changes any text into a hotspot for Bookreader documentation. For printed documentation, this tag generates the text you specify followed by a reference (in parentheses) to the section, chapter, or table, and so on.

EXAMPLES

The following example shows how to use the <HOTSPOT> tag with the optional text argument for books displayed by Bookreader:

```
i <HEAD1>(Introduction\intro)
<P>
The section <HOTSPOT>(cat_section\cats) discusses cats.
.
.
.
<HEAD1>(All About Cats\cat_section)
```

This example produces the following Bookreader output:

The section cats discusses cats.

ONLINE Doctype Tag Reference

<HOTSPOT>

The following example shows how to use the <HOTSPOT> tag without the optional text argument for books displayed by Bookreader:

```
2 <HEAD1>(Introduction\intro)
  <P>
  The section <HOTSPOT>(cat_section) discusses cats.
  .
  .
  <HEAD1>(All About Cats\cat_section)
```

This example produces the following Bookreader output:

The section **All About Cats** discusses cats.

The following example shows how to use the <HOTSPOT> tag with the optional text argument for printed books:

```
3 <HEAD1>(Introduction\intro)
  <P>
  The section <HOTSPOT>(cat_section\cats) discusses cats.
  .
  .
  <HEAD1>(All About Cats\cat_section)
```

This example produces the following Bookreader output:

The section (see Section n.n) discusses cats.

The following example shows how to use the <HOTSPOT> tag without the optional text argument for printed books:

```
4 <HEAD1>(Introduction\intro)
  <P>
  The section <HOTSPOT>(cat_section) discusses cats.
  .
  .
  <HEAD1>(All About Cats\cat_section)
```

This example produces the following Bookreader output:

The section **All About Cats** (see Section n.n) discusses cats.

<HELP_ONLY>

Identifies text that you want to include only in Help output and not in printed or Bookreader output.

SYNTAX <HELP_ONLY>

ARGUMENTS *None.*

related tags

- <BOOK_ONLY>
- <SET_HELP_LEVEL>

required terminator <ENDHELP_ONLY>

DESCRIPTION The <HELP_ONLY> tag identifies text that you want to include only in Help output and not in printed or Bookreader output.

EXAMPLE The following example shows how to use the <HELP_ONLY> tag.

```
<HELP_ONLY>
<P>When RSX . . .
<ENDHELP_ONLY>

<P>When the operating system . . .

<HELP_ONLY>
<P>When RSTS . . .
<ENDHELP_ONLY>
```

This example shows how to code a file so that the text between the <HELP_ONLY> and <ENDHELP_ONLY> tags is included only in the Help (.HLP) file and not in printed or Bookreader output. In this case, the paragraphs that begin with “When RSX” and “When RSTS” would be included only in the .HLP file. The following paragraph would appear only in the printed or Bookreader output:

When the operating system . . .

ONLINE Doctype Tag Reference

<KEEP_HELP_LEVEL>

<KEEP_HELP_LEVEL>

Allows you to override the default multi-level Help output and keep the Help output at a single level. This tag affects only the <COMMAND> and <SUBCOMMAND> tags of the SOFTWARE doctype.

SYNTAX <KEEP_HELP_LEVEL>

ARGUMENTS *None.*

- related tags**
- <BOOK_ONLY>
 - <HELP_ONLY>
 - <SET_HELP_LEVEL>

required terminator <ENDKEEP_HELP_LEVEL>

DESCRIPTION The <KEEP_HELP_LEVEL> tag allows you to override the default multi-level Help output and keep the Help output at a single level. This tag affects only the <COMMAND> and <SUBCOMMAND> tags of the SOFTWARE doctype.

Remember that each word in a command is a different Help level, by default. The <KEEP_HELP_LEVEL> tag concatenates all elements of its argument and places the entire argument at a single level. For example, if you have a command called SET TERMINAL and you do not want a Help file with SET at level-1 and TERMINAL at level-2, which is the default, but want both SET and TERMINAL at level-1, use the <KEEP_HELP_LEVEL> and <ENDKEEP_HELP_LEVEL> tags to enclose the command.

EXAMPLE The following example shows how to use the <KEEP_HELP_LEVEL> tag.

```
<COMMAND_SECTION>
<KEEP_HELP_LEVEL>
<COMMAND> (SET TERMINAL)
<ENDKEEP_HELP_LEVEL>
.
.
<COMMAND> (SET QUEUE)
.
.
<COMMAND> (SET PASSWORD)
.
.
<ENDCOMMAND_SECTION>
```

ONLINE Doctype Tag Reference

<KEEP_HELP_LEVEL>

This example shows how to use the <KEEP_HELP_LEVEL> and <ENDKEEP_HELP_LEVEL> tags to cause the enclosed command to be output as level-1 in the Help (.HLP) file. This example produces the following levels in the .HLP file:

```
1 SET_TERMINAL
1 SET
2 QUEUE
2 PASSWORD
```

ONLINE Doctype Tag Reference

<LMF>

<LMF>

For Bookreader documentation, the <LMF> tag marks the beginning of the tag sequence that specifies the License Management Facility (LMF) information for the document.

SYNTAX

<LMF>({ *book symbol name*
multibook license identifier })

ARGUMENTS

book symbol name

Specifies the symbol name for the book, which you must define using the <DEFINE_BOOK_NAME> tag.

multibook license identifier

Refers to a single set of LMF tags, which specify the licensing information for an entire documentation set or group of documents. This argument must match the argument to the <LMF_INFO> tag.

related tags

- <LMF_ALTNAME>
 - <LMF_INFO>
 - <LMF_PRODUCER>
 - <LMF_PRODUCT>
 - <LMF_RELEASE_DATE>
 - <LMF_VERSION_NUMBER>
-

restrictions

You can specify only one set of LMF tags per document.

The <LMF> tag must precede the <LMF_INFO> tag.

required terminator

<ENDLMF>

DESCRIPTION

For Bookreader documentation, the <LMF> tag marks the beginning of the tag sequence that specifies the License Management Facility (LMF) information for the document. You can specify the LMF tags in any order between the <LMF> and <ENDLMF> tags. The <LMF> tag has no effect for printed output.

The exact form of the LMF information is case- and punctuation-dependent. All the LMF information you supply as arguments to the LMF tags must match what exists in the LMF database.

To view books with the Bookreader, you must have installed at least one of the products specified in the <LMF_PRODUCT> and <LMF_ALTNAME> tags. Specify the primary product license in the <LMF_PRODUCT> field. Then, specify one or more alternate licenses for the book by using separate <LMF_ALTNAME> tags for each alternate product license.

You must specify all the LMF tags between the <LMF> and <ENDLMF> tags, except in the following cases:

- You must specify the <LMF_INFO> tag after the <LMF> tag, but the <LMF_INFO> tag does not have to fall within the <LMF> and <ENDLMF> tags.
- You do not have to specify the <LMF_ALTNAME> tag at all, although if you do not, you receive a warning-level message.

Make sure that if you do not have a specific release date and version number, you supply an empty field (that is, zero) to both the <LMF_RELEASE_DATE> and <LMF_VERSION_NUMBER> tags.

You can code the LMF tags in either the front matter file, the profile file, or in a symbols file that you process using the /SYMBOLS qualifier on the DOCUMENT command line. Consider storing the LMF tags in a central symbols file to make obtaining and updating the information easier.

EXAMPLES

The following example shows how to use the <LMF> tag in your front matter file.

```

i <FRONT_MATTER>(front)
    <DEFINE_BOOK_NAME>(book_symbol\User's Guide)
    .
    .
    .
    <LMF>(book_symbol)
    <LMF_PRODUCER>(DEC)
    <LMF_PRODUCT>(VAX-VMS)
    <LMF_RELEASE_DATE>(30-June-1990)
    <LMF_VERSION_NUMBER>(V3.0)
    <LMF_ALTNAME>(FORTRAN)
    <LMF_ALTNAME>(PASCAL)
    <ENDLMF>

    <TITLE_PAGE>
    <LMF_INFO>(book_symbol)
    <TITLE>(User's Guide)
    .
    .
    .
    <ENDTITLE_PAGE>
    <ENDFRONT_MATTER>
  
```

This example shows a complete set of LMF tags for a single document.

ONLINE Doctype Tag Reference

<LMF>

The following example shows how to use the <LMF> tag in a source file or in a symbols file.

2 Code in a symbols file:

```
<DEFINE_BOOK_NAME>(his_book\User's Guide)
<LMF>(many_books)
<LMF_PRODUCER>(DEC)
<LMF_PRODUCT>(FORTRAN)
<LMF_VERSION_NUMBER>(0)
<LMF_RELEASE_DATE>(0)
<LMF_ALTNAME>(ANOTHER_NAME)
<ENDLMF>
```

Code in a source file:

```
<FRONT_MATTER>(front)
<TITLE_PAGE>
<DEFINE_SYMBOL>(product_name\VAX FORTRAN)
<TITLE>(User's Guide)
<LMF_INFO>(many_books)
<ORDER_NUMBER>(AA-12345-BC)
<ellipsis>
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

This example shows, for an entire documentation set or any group of documents, how to use the <LMF> and <LMF_INFO> tags to access the same set of LMF tags for all the documents. Notice that you can use a symbols file to list all the LMF information, and then use the <LMF_INFO> tag in your front matter file to access the licensing information from the symbols file.

<LMF_ALTNAME>

For Bookreader documentation, the <LMF_ALTNAME> tag specifies an alternate product name for the software product.

SYNTAX <LMF_ALTNAME>(alternate name)

ARGUMENTS *alternate name*

Specifies an alternate product name. If you need to specify more than one alternate product name, use a separate <LMF_ALTNAME> tag for each alternate product name.

related tags

- <ENDLMF>
- <LMF>
- <LMF_INFO>
- <LMF_PRODUCER>
- <LMF_PRODUCT>
- <LMF_RELEASE_DATE>
- <LMF_VERSION_NUMBER>

restrictions

Valid only in the context of the <LMF> tag.

DESCRIPTION

For Bookreader documentation, the <LMF_ALTNAME> tag specifies an alternate product name for the software product. This tag has no effect for printed output.

You may need to supply more than one alternate product name for a product. You can supply multiple <LMF_ALTNAME> tags in any order.

You do not have to use the <LMF_ALTNAME> tag; if you do not use it, however, you receive a warning-level message.

EXAMPLE

See the example in the discussion of the <LMF> tag.

ONLINE Doctype Tag Reference

<LMF_INFO>

<LMF_INFO>

For Bookreader documentation, the <LMF_INFO> tag writes licensing information into the .DECW\$BOOK file.

SYNTAX

<LMF_INFO> ({ *book symbol name*
multibook license identifier })

ARGUMENTS

book symbol name

Specifies, for a single book, the *book symbol name* argument that you defined in the <DEFINE_BOOK_NAME> tag, thus matching the argument to the <DEFINE_BOOK_NAME> tag. It must also match the argument to the <LMF> tag.

multibook license identifier

Refers to a single set of LMF tags, which specify the licensing information for an entire documentation set or group of documents. This argument must match the argument to the <LMF> tag.

related tags

- <LMF>
 - <LMF_ALTNAME>
 - <LMF_PRODUCER>
 - <LMF_PRODUCT>
 - <LMF_RELEASE_DATE>
 - <LMF_VERSION_NUMBER>
-

restrictions

Must not precede the <LMF> tag.

DESCRIPTION

For Bookreader documentation, the <LMF_INFO> tag writes licensing information into the .DECW\$BOOK file. For a single book, the argument to the <LMF_INFO> tag must match the *book symbol name* argument you defined in the <DEFINE_BOOK_NAME> tag and must also match the argument to the <LMF> tag. Therefore, a set of LMF tags must exist either in a source file or in a symbols file.

For an entire documentation set or group of documents, use the *multibook license identifier* argument to refer to a single set of LMF tags. The *multibook license identifier* argument must match the argument to the <LMF> tag. This avoids having to specify the same LMF information for every document in the set.

You must specify the <LMF_INFO> tag after the <LMF> tag, but the <LMF_INFO> tag does not have to fall within the <LMF> and <ENDLMF> tags.

EXAMPLES

The following example shows how to use the <LMF_INFO> tag in a source file or in a symbols file for a single document.

1 Code in a source file or a symbols file:

```
<DEFINE_BOOK_NAME>(my_book\VMS Overview)
<LMF>(my_book)
<LMF_PRODUCER>(DEC)
<LMF_PRODUCT>(VAX-VMS)
<LMF_VERSION_NUMBER>(0)
<LMF_RELEASE_DATE>(0)
<LMF_ALTNAME>(ANOTHER_NAME)
<ENDLMF>
```

Code in a source file:

```
<PROFILE>
<LMF_INFO>(my_book)
<ELEMENT>(front_matter.sdml)
<ellipsis>
<ENDPROFILE>
```

This example shows, for a single document, how to use the <LMF_INFO> tag. Notice that the arguments to the <LMF> and the <LMF_INFO> tags match the *symbol name* argument to the <DEFINE_BOOK_NAME> tag.

The following example shows how to use the <LMF_INFO> tag in a source file or in a symbols file for a documentation set.

2 Code in a symbols file:

```
<DEFINE_BOOK_NAME>(his_book\User's Guide)
<LMF>(many_books)
<LMF_PRODUCER>(DEC)
<LMF_PRODUCT>(FORTRAN)
<LMF_VERSION_NUMBER>(0)
<LMF_RELEASE_DATE>(0)
<LMF_ALTNAME>(ANOTHER_NAME)
<ENDLMF>
```

Code in a source file:

```
<FRONT_MATTER>(front)
<TITLE_PAGE>
<DEFINE_SYMBOL>(product_name\VAX FORTRAN)
<TITLE>(User's Guide)
<LMF_INFO>(many_books)
<ORDER_NUMBER>(AA-12345-BC)
<ellipsis>
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

This example shows, for an entire documentation set or any group of documents, how to use the <LMF_INFO> tag to access the same set of LMF tags for all the documents.

ONLINE Doctype Tag Reference

<LMF_PRODUCER>

<LMF_PRODUCER>

For Bookreader documentation, the <LMF_PRODUCER> tag specifies the producer of the software product.

SYNTAX <LMF_PRODUCER>(producer)

ARGUMENTS *producer*
Specifies the name of the software producer, for example, DEC. The spelling of the product name information you supply must match exactly what exists in the LMF database. The name must not exceed 24 characters.

related tags

- <ENDLMF>
- <LMF>
- <LMF_ALTNAME>
- <LMF_INFO>
- <LMF_PRODUCT>
- <LMF_RELEASE_DATE>
- <LMF_VERSION_NUMBER>

restrictions Valid only in the context of the <LMF> tag.

DESCRIPTION For Bookreader documentation, the <LMF_PRODUCER> tag specifies the producer of the software product. This tag has no effect for printed output.

EXAMPLE See the example in the discussion of the <LMF> tag.

<LMF_PRODUCT>

For Bookreader documentation, the <LMF_PRODUCT> tag specifies the software product.

SYNTAX <LMF_PRODUCT>(product name)

ARGUMENTS *product name*
Specifies the name of the software product, for example, VAX-VMS. The spelling of the product name information you supply must match exactly what exists in the LMF database. The name must not exceed 24 characters.

related tags

- <ENDLMF>
- <LMF>
- <LMF_ALTNAME>
- <LMF_INFO>
- <LMF_PRODUCER>
- <LMF_RELEASE_DATE>
- <LMF_VERSION_NUMBER>

restrictions Valid only in the context of the <LMF> tag.

DESCRIPTION For Bookreader documentation, the <LMF_PRODUCT> tag specifies the software product. This tag has no effect for printed output.

EXAMPLE See the example in the discussion of the <LMF> tag.

ONLINE Doctype Tag Reference

<LMF_RELEASE_DATE>

<LMF_RELEASE_DATE>

For Bookreader documentation, the <LMF_RELEASE_DATE> tag specifies the release date of the software product.

SYNTAX <LMF_RELEASE_DATE>(*dd-mmm-yyyy*)

ARGUMENTS *dd-mmm-yyyy*
Specifies the day, month, and year of the release of the software product.

related tags

- <ENDLMF>
- <LMF>
- <LMF_ALTNAME>
- <LMF_INFO>
- <LMF_PRODUCER>
- <LMF_PRODUCT>
- <LMF_VERSION_NUMBER>

restrictions Valid only in the context of the <LMF> tag.

DESCRIPTION For Bookreader documentation, the <LMF_RELEASE_DATE> tag specifies the release date of the software product. This tag has no effect for printed output.

EXAMPLE See the example in the discussion of the <LMF> tag.

<LMF_VERSION_NUMBER>

For Bookreader documentation, the <LMF_VERSION_NUMBER> tag specifies the version number of the software product.

SYNTAX <LMF_VERSION_NUMBER>(version number)

ARGUMENTS *version number*
Specifies the version number of the software product.

related tags

- <ENDLMF>
- <LMF>
- <LMF_ALTNAME>
- <LMF_INFO>
- <LMF_PRODUCER>
- <LMF_PRODUCT>
- <LMF_RELEASE_DATE>

restrictions Valid only in the context of the <LMF> tag.

DESCRIPTION For Bookreader documentation, the <LMF_VERSION_NUMBER> tag specifies the version number of the software product. This tag has no effect for printed output.

EXAMPLE See the example in the discussion of the <LMF> tag.

ONLINE Doctype Tag Reference

<ONLINE_CHUNK>

<ONLINE_CHUNK>

For Bookreader documentation, the <ONLINE_CHUNK> tag breaks lengthy pieces of text into **online chunks** to prevent the text formatter from running out of memory.

SYNTAX <ONLINE_CHUNK>

ARGUMENTS *None.*

restrictions Invalid in **formal tables** and **formal examples**.

related tags

- <BOOK_ONLY>
- <HELP_ONLY>

DESCRIPTION For Bookreader documentation, the <ONLINE_CHUNK> tag breaks lengthy pieces of text into **online chunks** to prevent the text formatter from running out of memory. Using this tag, then, prevents undesirable breaks of information. Use this tag carefully and sparingly.

When the text formatter breaks long code examples, about an inch of white space is sometimes left in the Bookreader output. Inserting an <ONLINE_CHUNK> tag *before* this space removes the extra white space.

This tag has no effect for printed output.

EXAMPLES

The following example shows how to use the <ONLINE_CHUNK> tag.

```
1 <code_example>
SQL> !
SQL> ! You can see from the following display that the CURRENT_INFO
SQL> ! view contains an employee with a last name Toliver and an ID
SQL> ! number 00164.
SQL> !
SQL> SELECT LAST_NAME, FIRST_NAME, ID FROM CURRENT_INFO ORDER BY ID;
LAST_NAME      FIRST_NAME     ID
Toliver        Alvin          00164
Smith          Terry          00165
Dietrich       Rick           00166
Kilpatrick    Janet          00167
Nash           Norman         00168
<ellipsis>
<valid_break>
SQL> !
SQL> SELECT LAST_NAME, FIRST_NAME, EMPLOYEE_ID
cont> FROM CURRENT_JOB
cont> WHERE JOB_START > "AUG-26-1981";
%SQL-F-DATCONERR, Data conversion error
-LIB-F-AMBDATTIM, ambiguous date-time
<tag>(online_chunk)
SQL> !
SQL> ! Now (finally) the correct way to compare a literal to a
SQL> ! value in a column defined as DATE.
SQL> !
SQL> SELECT LAST_NAME, FIRST_NAME, EMPLOYEE_ID
cont> FROM CURRENT_JOB
<ellipsis>
<tag>(endcode_example)
```

This example shows how to use the <ONLINE_CHUNK> tag in a long code example.

ONLINE Doctype Tag Reference

<ONLINE_POPUP>

<ONLINE_POPUP>

Specifies that an informal text element appear (that is, pop up) in a separate window (as do **formal examples**, **formal figures**, and **formal tables**).

SYNTAX

<ONLINE_POPUP>(*text type*)

ARGUMENTS

text type

Specifies a short description of the type of text you want to pop up.

restrictions

A <HEAD_n> tag is invalid in the context of an <ONLINE_POPUP> tag. Begin a pop-up after a <HEAD_n> tag and end it before the next <HEAD_n> tag. This includes template headings, such as *Description* and *Examples*.

Do not nest <ONLINE_POPUP> tags.

Do not use with formal examples, figures, or tables.

required terminator

<ENDONLINE_POPUP>

related tags

- <ONLINE_CHUNK>
- <ONLINE_TITLE>

DESCRIPTION

The <ONLINE_POPUP> tag specifies that an informal text element appear (that is, pop up) in a separate window (as do **formal examples**, **formal figures**, and **formal tables**). Use this tag to display the text elements more clearly and to help keep the **online topics** that contain the pop-ups from becoming unmanageably long for Bookreader navigation. This tag has no effect for printed output.

Note: The text formatter may run out of memory when processing long topics that contain many long, informal tables. Use the <ONLINE_POPUP> tag to pop up the tables in separate windows.

You can use the <ONLINE_POPUP> tag with any segment of text: paragraphs, lists, or notes. However, use the tag sparingly; too many pop-up windows can hinder the usability of your document. Pop-up windows are most useful for displaying graphics or text segments that are somewhat peripheral to the flow of text in the manual. Experiment to determine if a pop-up window is a useful and effective means of presenting peripheral information. Too many pop-up windows create clutter on the screen and confusion for the user.

ONLINE Doctype Tag Reference

<ONLINE_POPUP>

A pop-up window must have an associated text window **hotspot** at which the user can point and click. The <ONLINE_POPUP> tag automatically creates a hotspot and uses the argument you supply. For example, if you supply the argument *table*, the hotspot is the following:

TABLE: Click here to display table.

Do not use the <ONLINE_POPUP> tag with formal examples, figures, or tables. Hotspots for formal elements are created when you use the <REFERENCE> tag to refer to them.

EXAMPLES

The following example shows how to use the <ONLINE_POPUP> tag for an informal table.

```
1 <ONLINE_POPUP>(table)
  <TABLE>
  .
  .
  .
  <ENDTABLE>
  <ENDONLINE_POPUP>
```

This example shows how to use the <ONLINE_POPUP> and <ENDONLINE_POPUP> tags so that an informal table pops up in a separate window.

The following example shows how to use the <ONLINE_POPUP> tag within an example sequence.

```
2 <EXAMPLE_SEQUENCE>
  <ONLINE_POPUP>(Example)
  <EXC>
  .
  .
  .
  <EXTEXT>
  .
  .
  .
  <ENDONLINE_POPUP>
  <ENDEXAMPLE_SEQUENCE>
```

This example shows how to use the <ONLINE_POPUP> and <ENDONLINE_POPUP> tags within an example sequence. You can find the <EXAMPLE_SEQUENCE>, <EXC>, and <EXTEXT> tags in *VAX DOCUMENT Using Doctypes and Related Tags*.

ONLINE Doctype Tag Reference

<ONLINE_TITLE>

<ONLINE_TITLE>

Specifies text that overrides the default section title in the topic label above the text in the text window.

SYNTAX <ONLINE_TITLE>(text)

ARGUMENTS *text*
Specifies the text that you want to appear in the title region. The text should be no longer than 40 characters, because the region where the title appears is narrow. If part of the title is hidden, you must resize the window.

restrictions Do not use tags in the *text* argument.

related tags • <ONLINE_CHUNK>
 • <ONLINE_POPUP>

DESCRIPTION The <ONLINE_TITLE> tag specifies text that overrides the default section title in the topic label above the text in the text window. This tag overrides only the current title; the next topic title is displayed as usual, unless you override it with another <ONLINE_TITLE> tag. This tag has no effect for printed output.

Use the <ONLINE_TITLE> tag on the title page, just before the <TITLE> tag (or for the MILSPEC.ONLINE doctype, just before the <SPEC_TITLE> tag) to specify an abbreviated title that will appear in three places: the title bar, the topic bar, and the Bookreader library. This tag is especially useful for documents that you create with the MILSPEC.ONLINE doctype, because these documents often have very long titles.

EXAMPLES The following example shows how to use the <ONLINE_TITLE> tag.

```
1 <ONLINE_TITLE>(Routine$Name)
   <ROUTINE>(ANY$ROUTINE$NAME)
   .
   .
   .
```

This example specifies that the title be displayed as “Routine\$Name”, rather than the full title, “ANY\$ROUTINE\$NAME”.

The following example shows how to use the <ONLINE_TITLE> tag to produce a Bookreader title that is significantly shorter than the printed title.

ONLINE Doctype Tag Reference

<ONLINE_TITLE>

2 <FRONT_MATTER>(front)
<TITLE_PAGE>
<ONLINE_TITLE>(Short Title)
<TITLE>(A Very Long Title That You Might Want To Abbreviate)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>

This example shows how to use the <ONLINE_TITLE> tag on the title page of a document.

ONLINE Doctype Tag Reference

<SET_HELP_LEVEL>

<SET_HELP_LEVEL>

Allows you to alter the default Help levels in your Help files.

SYNTAX

<SET_HELP_LEVEL>*[(number)]*

ARGUMENTS

number

This is an optional argument. It specifies a positive or negative number that is added to or subtracted from the default value to determine a new Help level. Note that this number is not the Help level number, but a value to be applied to the default Help level.

To reset the default Help levels, specify zero (0) as the *number* argument or do not use an argument. For example, both the <SET_HELP_LEVEL> and <SET_HELP_LEVEL>(0) tags reset the default Help levels.

related tags

- <BOOK_ONLY>
 - <HELP_ONLY>
 - <KEEP_HELP_LEVEL>
-

DESCRIPTION

The <SET_HELP_LEVEL> tag allows you to alter the default Help levels in your Help files. Remember that each word in the command is a different Help level, by default. This tag changes all the default Help levels until you explicitly reset them using the tag again without an argument, or with the zero (0) argument.

For example, by default <HEAD1>, <STATEMENT>, and <COMMAND> tags produce level-1 Help topics. You may want, however, your level-1 “Command” topic to be a level-2 topic, and the “Format”, “Qualifier”, and “Description” sections, which are normally level-2 topics, to be level-3 topics. In this case, use the <SET_HELP_LEVEL>(1) tag before the Help level you want to alter. Using the argument 1 adds one level to the default level-1, thus adding one level to each subsequent Help level.

If you use a negative *number* argument, one level is subtracted from the default Help level. For example, if you want your level-2 “Description” section to be a level-1, use the <SET_HELP_LEVEL>(-1) tag before the <DESCRIPTION> tag. If you want your level-3 “Example” section to be a level-1, use the <SET_HELP_LEVEL>(-2) tag before the <EXAMPLE> tag.

When you want to reset the default Help levels, use the <SET_HELP_LEVEL> tag with or without the zero (0) argument.

EXAMPLE

The following example shows how to use the <SET_HELP_LEVEL> tag.

```
<COMMAND_SECTION>
<COMMAND> (SET TERMINAL)
.
.
.
<SET_HELP_LEVEL> (1)
<COMMAND> (SET QUEUE)
.
.
.
<SET_HELP_LEVEL> (0)
<COMMAND> (SET PASSWORD)
.
.
.
<ENDCOMMAND_SECTION>
```

This example shows how to use the <SET_HELP_LEVEL> tag to alter the default Help levels. One Help level is added to the commands following the <SET_HELP_LEVEL> tag. You reset the default Help levels with another <SET_HELP_LEVEL> tag, with the zero (0) argument or without an argument. This example produces the following levels in the .HLP file:

```
1 SET
2 TERMINAL
2 SET
3 QUEUE
2 PASSWORD
```

ONLINE Doctype Tag Reference

<SET_ONLINE_TOPIC>

<SET_ONLINE_TOPIC>

Specifies the level of topics in your Bookreader document and whether table of contents entries are issued for ranges of messages or glossary items.

SYNTAX

<SET_ONLINE_TOPIC>({
 chapter
 head1
 head2
 head3
 head4
 head5
 head6
 [MASTER]
 [NOMASTER] })

ARGUMENTS

chapter

Specifies that each *entire* chapter is a topic, rather than each <HEAD1> tag in a chapter beginning a new topic, which is the default.

head1 through head6

Specifies the level of heading you want to begin a topic. That heading and all higher-level headings and chapters become topics.

MASTER, NOMASTER

These are optional keyword arguments. For a glossary and a messages appendix, MASTER specifies that table of contents entries are generated for the ranges of entries on each topic of messages or glossary items. NOMASTER suppresses the ranges of entries that are listed in the table of contents for the glossary or messages appendix.

DESCRIPTION

The <SET_ONLINE_TOPIC> tag specifies the level of topics in your Bookreader document and whether table of contents entries are issued for ranges of messages or glossary items. The tag is in effect until you override it with a new <SET_ONLINE_TOPIC> tag. For Bookreader, a topic is equal to a page. Topics are not broken according to the physical dimension of a page; they are broken according to topics of contents in the document. This tag has no effect for printed output.

By default, chapters, first-level headings, appendixes, parts, templates, title pages, copyright pages, prefaces, and glossaries are topics. In addition, because they pop up in separate windows, **formal examples**, **formal figures**, and **formal tables** are topics. However, these defaults may not be appropriate for your manual.

ONLINE Doctype Tag Reference

<SET_ONLINE_TOPIC>

Use the <SET_ONLINE_CHAPTER>(CHAPTER) tag if you have a short chapter that you want to designate as a whole topic. This remedies seeing each level-one heading as a topic, which may be distracting when viewed with Bookreader.

You can use the <SET_ONLINE_TOPIC> tag as many times as necessary. By default, Bookreader treats chapters and first-level headings as topics. If you want various heading levels to specify topics in various chapters or appendixes, specify the <SET_ONLINE_TOPIC> tag several times. The heading level you specify in the argument and all higher-level heading levels become topics. For example, if you specify <SET_ONLINE_TOPIC>(HEAD2), chapter headings, first-level headings, and second-level headings all begin new topics.

If you use the <SET_ONLINE_TOPIC> tag in your profile file, place the tag *after* the <ELEMENT> tag to which it applies.

The <SET_ONLINE_TOPIC> tag also specifies whether table of contents entries are issued for ranges of messages or glossary items. For messages and glossaries, use the MASTER argument to specify that table of contents entries be generated for the ranges of entries on each topic (or page) of messages or glossary items. This can be helpful to the user in very long sections of messages or glossary items.

If you want table of contents entries for messages but not for a glossary, use the NOMASTER argument to suppress the table of contents entries in the glossary. See *VAX DOCUMENT Producing Online and Printed Documentation* for more information on using this tag in a glossary or a messages appendix.

EXAMPLES

The following example shows how to use the <SET_ONLINE_TOPIC> tag to make online topics out of all heads level-2 and higher.

```
1 <SET_ONLINE_TOPIC>(head2)
  <CHAPTER>(Introduction\intro_chap)
  .
  .
```

This example specifies that all <HEAD2> tags, <HEAD1> tags, and <CHAPTER> tags start a new topic.

The following example shows how to use the <SET_ONLINE_TOPIC> tag to make online topics out of chapters.

```
2 <SET_ONLINE_TOPIC>(chapter)
  <CHAPTER>(Introduction\intro_chap)
  .
  .
```

This example specifies that each entire chapter becomes a new topic, instead of each first-level heading within the chapter.

ONLINE Doctype Tag Reference

<SET_ONLINE_TOPIC>

The following example shows how to use the <SET_ONLINE_TOPIC> tag in a profile, first to make online topics out of level-2 heads in one chapter, and then to reset to the default for following chapters.

```
3 <PROFILE>
  <ELEMENT>(mydisk:[mydir]intro_chap.sdml)
  <ELEMENT>(mydisk:[mydir]tools_chap.sdml)
  <SET_ONLINE_TOPIC>(head2)
  <ELEMENT>(mydisk:[mydir]commands_chap.sdml)
  <SET_ONLINE_TOPIC>
  .
  .
  .
<ENDPROFILE>
```

This example specifies that all <HEAD2> tags in TOOLS_CHAP.SDML signify the start of a new topic for Bookreader. For COMMANDS_CHAP.SDML, topics are reset back to the default (<HEAD1>).

The following example shows how to use the <SET_ONLINE_TOPIC> tag to issue table of contents entries for one section of a document, but not another.

```
4 <SET_ONLINE_TOPIC>(MASTER)
  <GLOSSARY>
  <GTERM>(Term)
  <GDEF>(Definition)
  .
  .
  .
  <SET_ONLINE_TOPIC>(NOMASTER)
  <MESSAGES_SECTION>
  <MSG>(Message)
  <MSG_TEXT>(Text.)
  .
  .
  .
```

This example specifies that table of contents entries be issued for the glossary but not for the messages section.

<SHELF_CREATE>

Outputs a **SHELF** entry for the current Bookreader bookshelf file and creates a separate **bookshelf** file.

SYNTAX <SHELF_CREATE> (*symbol name \ file spec*)

ARGUMENTS

symbol

Specifies a symbol name that defines the shelf title. Define this symbol in either your source file or a symbol definition file (using the <DEFINE_SYMBOL> tag). If you define it in a symbol definition file, include it using the \SYMBOLS qualifier on the DOCUMENT command line. If you do not define the bookshelf symbol name, the symbol is used as the title.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

file spec

Specifies the file name of the bookshelf file. The file type is not required; the default of .DECW\$BOOKSHELF is assumed.

Note: If you specify the directory as well as the file name (for example, [PROJECT_A.BOOKS]PROGRAMMING.DECW\$BOOKSHELF), the shelf file is created in that directory. Also, the SHELF entry written to the current bookshelf file includes the directory. If you do not specify a directory, the file is created in the default directory.

When building bookshelves for ULTRIX systems, type file names in lowercase.

related tags

- <BOOK_REF>
- <SHELF_REF>

required terminator

<ENDSHELF_CREATE>

DESCRIPTION

The <SHELF_CREATE> tag outputs a SHELF entry for the current Bookreader bookshelf file and creates a separate bookshelf file. For example, if you specify the *file spec* argument /PROGRAMMING, a PROGRAMMING.DECW\$BOOKSHELF file is created. If you specify a directory as well as the file specification, the new bookshelf file is created in that directory and the SHELF entry written to the current bookshelf file includes that directory. For example, the tag

ONLINE Doctype Tag Reference

<SHELF_CREATE>

```
<SHELF_CREATE> (prog_man\[PROJECT_A.BOOKS]PROGRAMMING)
```

creates a bookshelf file PROGRAMMING.DECW\$BOOKSHELF in the directory [PROJECT_A.BOOKS]. The SHELF entry in the current bookshelf file is:

```
SHELF\[PROJECT_A.BOOKS]PROGRAMMING\Programming With XYZ
```

Note: When you want to refer to an existing shelf or library file, use the <SHELF_REF> tag.

You can nest bookshelves or create them as individual files. Include existing bookshelf source (.SDML) files with the <INCLUDE> tag.

EXAMPLE

The following example shows how to use the <BOOKREF>, <SHELF_CREATE>, and <SHELF_REF> tags.

```
<SHELF_CREATE> (vms_sys_mgmt\LIBRARY)
  <BOOK_REF> (using_bookreader\AA-BOOKS-AA)
  <BOOK_REF> (vax_document\AA-LA95A-TE)
  <SHELF_REF> (vms_base_docset\VMS_BASE)
  <SHELF_CREATE> (vms_decw_usr\VMS_DECW_USER)
    <BOOK_REF> (vms_decw_overview\AA-MG17A-TE)
    <BOOK_REF> (vms_decw_ug\AA-MG18A-TE)
    <BOOK_REF> (vms_decw_desk_app\AA-MG19A-TE)
  <ENDSHELF_CREATE>
  <INCLUDE> (vms_system_management.sdml)
  <SHELF_CREATE> (vms_gen_user\VMS_GENERAL_USER)
    <BOOK_REF> (vms_gloss\AA-LA03A-TE)
    <BOOK_REF> (vms_intro\AA-LA04A-TE)
    .
    .
  <ENDSHELF_CREATE>
<ENDSHELF_CREATE>
```

<SHELF_REF>

Outputs a **SHELF** entry for a Bookreader bookshelf file that points to an existing bookshelf file.

SYNTAX <SHELF_REF>(symbol name \ file spec)

ARGUMENTS

symbol name

Specifies a symbol name that provides the title of the shelf. Define this symbol in either your source file or a symbol definition file (using the <DEFINE_SYMBOL> tag). When you define this symbol in a symbol definition file, include it by using the \SYMBOLS qualifier on the DOCUMENT command line.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

file spec

Specifies the book file specification or symbol name that points to the *shelf_filename*.DECW\$BOOKSHELF file. The file type is not required; the default of .DECW\$BOOKSHELF is assumed.

Note: When building bookshelves for ULTRIX systems, type file names in lowercase.

related tags

- <BOOK_REF>
- <SHELF_CREATE>

restrictions

Valid only in the context of the <SHELF_CREATE> tag.

DESCRIPTION

The <SHELF_REF> tag outputs a **SHELF** entry for a Bookreader bookshelf file that points to an existing bookshelf file. This tag has no effect for printed output. You can use the <INCLUDE> tag in place of the <SHELF_REF> tag; both produce the same results.

Note: If you need to create a new shelf file, you must use the <SHELF_CREATE> tag.

EXAMPLE

See the example in the discussion of the <SHELF_CREATE> tag.

8

Using the OVERHEADS Doctype

The OVERHEADS doctype lets you create pages with large, bold text. This text copies well and is easy to see. Use these doctype designs to create transparent slides for an overhead projector or 35mm photographic mounts for a 35mm slide projector.

The OVERHEADS doctype has two designs, shown in Figure 8-1.

- OVERHEADS

Creates a page with an 8.5 × 11-inch page with a 7 × 8.7-inch image area. Use this design to create slides that fit on overhead projectors, or figures that fit into an 8.5 × 11-inch notebook.

- OVERHEADS.35MM

Creates a page with an 8.5 × 11-inch page with a 6.5 × 4.7-inch image area used to create 35mm slides by photographic reduction.

Figure 8-1 OVERHEADS Doctype Designs

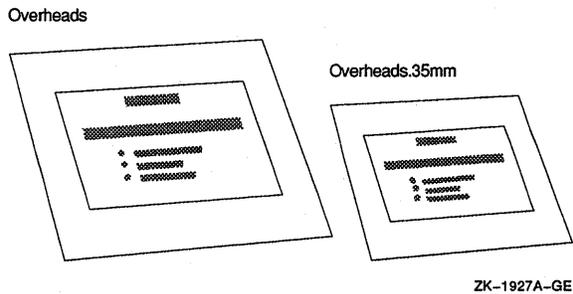


Table 8-1 lists the page layout characteristics of the OVERHEADS doctype design. Table 8-2 lists the page layout characteristics of the OVERHEADS.35MM doctype design.

Table 8-1 Page Layout of the OVERHEADS Design

Page Layout Characteristics	
Running heads	Optional
Running feet	Optional
Page numbering	Optional
Trim size	7 x 8.7 in. centered on 8.5 x 11 in. paper
Gutter width	0
Right margin	Unjustified (Ragged right)

Using the OVERHEADS Doctype

Table 8–1 (Cont.) Page Layout of the OVERHEADS Design

Text Element Characteristics	
Headings	Unnumbered
Paragraphs	Flush left at left margin
Figures, tables, and examples	Numbered

Table 8–2 Page Layout of the OVERHEADS.35MM Design

Page Layout Characteristics	
Running heads	Optional
Running feet	Optional
Page numbering	Optional
Trim size	6.5 x 4.7 in. for reduction to 35 mm slide
Gutter width	0
Right margin	Unjustified (Ragged right)

Text Element Characteristics	
Headings	Unnumbered
Paragraphs	Flush left at left margin
Figures, tables, and examples	Numbered

Table 8–3 briefly describes the tags specific to the OVERHEADS doctype. Section 8.2 contains the reference information on the tags listed in this table.

Table 8–3 Tags Available in the OVERHEADS Doctype

Tag Name	Description
<AUTHOR_INFO>	Specifies from one to four lines of information about the slide presentation. This information outputs on the right side of the slide toward the bottom edge.
<AUTO_NUMBER>	Automatically numbers slides that occur in the same file. Additionally, you can specify an argument to this tag that places a text string at the beginning of each slide number, for example, Vacation-1.
<INTRO_SUBTITLE>	Creates a secondary title of one to four lines on an introductory slide. Typically, titles created using the <INTRO_SUBTITLE> tag occupy a large amount of space on the slide.
<INTRO_TITLE>	Creates a main title of one to four lines on an introductory slide. Typically, titles created using the <INTRO_TITLE> tag occupy a large amount of space on the slide.
<RUNNING_FEET>	Places a heading at the bottom right of the current slide and all subsequent slides. If you use the <RUNNING_FEET> tag with the <AUTO_NUMBER> tag, the slide number outputs on the bottom right, and the text string outputs on the bottom left.
<RUNNING_TITLE>	Places a title at the top of all subsequent slides.

Table 8-3 (Cont.) Tags Available in the OVERHEADS Doctype

Tag Name	Description
<SLIDE>	Begins a new overhead slide. Optionally, use this tag to specify text to be placed at the bottom of the slide.
<SUBTITLE>	Creates a secondary title with one to four title lines for a nonintroductory slide. The <SUBTITLE> tag produces title lines in a smaller type face with less space between lines than the title lines produced by the <INTRO_SUBTITLE> tag.
<TEXT_SIZE>	Modifies the size of the type face used in a topic, table, or list on a single slide.
<TITLE>	Creates a main title with one to four title lines for a nonintroductory slide. The <TITLE> tag produces title lines in the same type face, but with less space between lines than the title lines produced by the <INTRO_TITLE> tag.
<TOPIC>	Specifies a line of topic text for a slide. This text begins at the left margin and is in a smaller type font than the font used by the <TITLE> tag.

Process a file with the OVERHEADS doctype using one of the doctype keywords in the preceding list on the DOCUMENT command line. The following example shows how to process a file named MYGRAPHIC.SDML with the OVERHEADS.35MM doctype to create a 35mm slide mount:

```
$ DOCUMENT mygraphic OVERHEADS.35MM LN03
```

Use the POSTSCRIPT or LN03 destinations when you create your overheads. These destinations provide better quality output than the LINE_PRINTER destination. Then make xerographic copies of your printed VAX DOCUMENT files and use these copies to create the overhead transparencies. Xerographic copies of laser printer output often reproduce better than the original laser printer output.

8.1

A Sample Use of the OVERHEADS Doctype Tags

This section contains an example of two slides produced using OVERHEADS doctype tags. The first slide is an introductory slide for a slide presentation. The second slide is the first that follows the introduction.

The SDML code for the two slides is shown first, followed by the output from that SDML code using the OVERHEADS doctype.

```
<SLIDE>(Presented 3/8/90)
<AUTO_NUMBER>(DMS)
<RUNNING_TITLE>(Pets Are People Too)
<RUNNING_FEET>(Pet selection seminar)
<INTRO_TITLE>(CHOOSING THE\RIGHT PET FOR\YOU)
<INTRO_SUBTITLE>(A Seminar on\Pet Selection)
<AUTHOR_INFO>(D. M. Smith\Veterinarian)
```

Using the OVERHEADS Doctype

```
<SLIDE>
<TITLE>(Heart Rates\In Pets and Wild Mammals)
<SUBTITLE>(Physiological Data \on Selected Mammals)
<TEXT_SIZE>(REGULAR)
<TOPIC>(Nondomesticated Mammal Heart Rates)
<TABLE>
<table_attributes>(KEEP)
<table_setup>(3\19\16)
<table_heads>(Common Name\Weight\Heart Rate)
<table_row>(European hedgehog\500-700 g.\246)
<table_row>(Gray shrew\3-4 g.\782)
<table_row>(Least chipmunk\40 g.\684)
<table_row>(Gray squirrel\500-600 g.\390)
<table_row>(Harbor porpoise\170 kg.\40-110)
<table_row>(Mink\0.7-1.4 kg.\272)
<table_row>(Harbor seal\20-25 kg.\18-25)
<ENDTEXT_SIZE>
<endtable>
```

Figure 8-2 and Figure 8-3 show the corresponding output from the SDML file processed with the OVERHEADS doctype. Comparing these samples may be helpful in understanding how to use these tags to create outputs for overheads. Should you wish to create this output yourself, you can obtain file OVERHEADS_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

When you process the input file with the OVERHEADS doctype, output has a trim size image area of 7 x 8.7 inches. When you process with the OVERHEADS.35MM doctype, output has a trim size image area of 6.5 x 4.7 inches, the proper proportion for photographic reduction to 35 mm slides.

Figure 8-2 OVERHEADS Doctype Output Example, First Slide

CHOOSING THE RIGHT PET FOR YOU

**A Seminar on
Pet Selection**

**D. M. Smith
Veterinarian**

Using the OVERHEADS Doctype

Figure 8-3 OVERHEADS Doctype Output Example, Second Slide

Pets Are People Too

Heart Rates In Pets and Wild Mammals

Physiological Data on Selected Mammals

Nondomesticated Mammal Heart Rates

Common Name	Weight	Heart Rate
European hedgehog	500-700 g.	246
Gray shrew	3-4 g.	782
Least chipmunk	40 g.	684
Gray squirrel	500-600 g.	390
Harbor porpoise	170 kg.	40-110
Mink	0.7-1.4 kg.	272
Harbor seal	20-25 kg.	18-25

8.2 OVERHEADS Doctype Tag Reference

This part of Chapter 8 provides reference information on all the tags specific to the OVERHEADS doctype.

<AUTO_NUMBER>

Specifies that slides be numbered automatically, and that the slide number is at the bottom of every slide. Optionally, specifies text to be placed in front of the slide number on each page.

SYNTAX <AUTO_NUMBER>[(*text*)]

ARGUMENTS *text*
This is an optional argument. It specifies text to be placed in front of the slide number at the foot of every overhead slide. If you do not specify this argument, only the slide number prints.

related tags

- <RUNNING_FEET>
- <SLIDE>

DESCRIPTION The <AUTO_NUMBER> tag specifies that slides be numbered automatically, and that the slide number is at the bottom of every slide. Optionally, it specifies text to be placed in front of the slide number on each page. By default, no page or slide number appears on the bottom of overhead slides. To override this default behavior, do the following:

- Use the <AUTO_NUMBER> tag to request numbering or to optionally specify text to go along with the numbers.
- Use the <RUNNING_FEET> tag to specify text to be placed on the bottom of every overhead slide.
- Use an argument to the <SLIDE> tag. The text of the tag's argument outputs on the bottom of the current slide.

EXAMPLES In the following example, the <AUTO_NUMBER> tag specifies that the slides are to be numbered. The current number outputs in the bottom right corner of every slide.

1 <AUTO_NUMBER>

In the following example, the <AUTO_NUMBER> tag specifies that the slides are to be numbered and that the numbers are to be preceded with the word Earnings and placed in the bottom right corner of every slide.

2 <AUTO_NUMBER>(Earnings)

OVERHEADS Doctype Tag Reference

<RUNNING_FEET>

<RUNNING_FEET>

Specifies text to be placed at the bottom of the next slide and all subsequent slides.

SYNTAX

<RUNNING_FEET> (*running footer text*)

ARGUMENTS

running footer text

Specifies the text to be placed on the bottom left of your slides.

related tags

- <AUTO_NUMBER>
- <SLIDE>

DESCRIPTION

The <RUNNING_FEET> tag specifies text to be placed at the bottom of the next slide and all subsequent slides. This running footer outputs at the bottom left of the slide. To place running feet on all of your slides, place this tag in your SDML file before the first occurrence of a <SLIDE> tag. To disable running feet, place the <RUNNING_FEET> tag in your SDML file with no argument.

By default, no page or slide number outputs on the bottom of overhead slides. To override this default behavior, do the following:

- Use the <AUTO_NUMBER> tag to request numbering and to optionally specify text to go along with the numbers.
- Use the <RUNNING_FEET> tag to specify text to be placed on the bottom of every overhead slide.
- Use an argument to the <SLIDE> tag. The text of the tag's argument outputs on the bottom of the current slide.

If you specify <RUNNING_FEET> in conjunction with <AUTO_NUMBER>, the slide number outputs on the right and the text on the left.

EXAMPLES

In the following example, the <RUNNING_FEET> tag specifies that all slides are to carry the text May 1986 Presentation at the bottom right corner.

```
1 <RUNNING_FEET> (May 1986 Presentation)
  <SLIDE>
  <TITLE> (Base Level 13)
```

OVERHEADS Doctype Tag Reference

<RUNNING_FEET>

In the following example, the <AUTO_NUMBER> tag creates automatic numbering of the slides at the bottom right of the slide, and the <RUNNING_FEET> tag places a running footer on the bottom left of the slide.

2 <RUNNING_FEET> (May 1987)
<AUTO_NUMBER> (Slide)

OVERHEADS Doctype Tag Reference

<RUNNING_TITLE>

<RUNNING_TITLE>

Creates a 1- or 2-line running heading at the top of each slide.

SYNTAX

$$\langle \text{RUNNING_TITLE} \rangle (\left\{ \begin{array}{l} \text{OFF} \\ \text{title-1} [\backslash \text{title-2}] \\ [\backslash \text{FIRST_PAGE}] \end{array} \right\})$$

ARGUMENTS

OFF

Specifies that any existing running titles created using the <RUNNING_TITLE> tag are disabled for the slide on which this tag occurs and on any subsequent slides.

title-1

Specifies the text of a running title. If you specify a 2-line title, this title outputs on the upper title line.

title-2

This is an optional argument. It specifies the bottom line of a running title that has two lines.

FIRST_PAGE

This is an optional argument. It specifies that the running title is to begin output on the first slide. If you do not specify this keyword, the running title begins on the slide after the current slide.

related tags

- <RUNNING_FEET>
- <SLIDE>

DESCRIPTION

The <RUNNING_TITLE> tag creates a 1- or 2-line running heading at the top of each slide. To place a running title on all your slides, place this tag in your SDML file before the first <SLIDE> tag. If you use this tag in the context of the first <SLIDE> tag and you want the running title to begin on that slide, use the FIRST_PAGE argument.

Use the OFF argument to disable any existing running titles created using the <RUNNING_TITLE> tag. These titles will then be disabled for the slide page on which this tag occurs and on any subsequent pages.

Use the <RUNNING_FEET> tag to create a heading that appears at the bottom of the slide page. See the reference description of the <RUNNING_FEET> tag for more information on that tag.

EXAMPLES

The following example shows how each occurrence of a <RUNNING_TITLE> tag creates a running title for the next slide or series of slides.

```
1 <RUNNING_TITLE>(Introduction to SDML)
  <SLIDE>
  <TOPIC>(What is SDML?)
  <SLIDE>
  <TOPIC>(What is a Tag?)

  <RUNNING_TITLE>(Overview of the Tags)
  <SLIDE>
  <TOPIC>(The Basic Tags)
  .
  .
  .
```

The following example shows how to disable a running title by using the OFF argument to the <RUNNING_TITLE> tag.

```
2 <COMMENT>(turn off running titles for the next slide)
  <RUNNING_TITLE>(OFF)
  <SLIDE>
  <TOPIC>(An Example of Output)
```

OVERHEADS Doctype Tag Reference

<SLIDE>

<SLIDE>

Begins a new overhead slide.

SYNTAX

<SLIDE>(*[footer-text]*)

ARGUMENTS

footer-text

This is an optional argument. It specifies text to be placed at the bottom of the slide.

related tags

- <AUTHOR_INFO>
- <AUTO_NUMBER>
- <RUNNING_FEET>
- <RUNNING_TITLE>
- <SUBTITLE>
- <TEXT_SIZE>
- <TITLE>
- <TOPIC>

DESCRIPTION

The <SLIDE> tag begins a new overhead slide. With an argument, this tag specifies text to be placed at the bottom of the slide. An overhead slide can contain the following:

- Major titles and subtitles
Use the <INTRO_TITLE> and <INTRO_SUBTITLE> tags to create title page slides.
- Titles and subtitles followed by topics, lists, tables, or text
Use the <TITLE> and <SUBTITLE> tags to place headings at the top of each new slide. Use the global <LIST>(NUMBERED) and <LIST>(UNNUMBERED) tags for numbered and unnumbered lists.

Use any of the global tags to specify text elements on any slide page, except those associated with the creation of front matter, appendixes, glossaries, indexes, or part pages.

EXAMPLES

The following example specifies a tag sequence that begins a new slide and specifies two main topics.

```
1 <SLIDE>
  <TITLE>(System Components)
  <TOPIC>(Parser)
  <TOPIC>(Interpreter)
```

The following example specifies a slide with a main heading and a subheading. The text Slide 1 outputs at the bottom of this slide.

```
2 <SLIDE>(Slide 1)
  <INTRO_TITLE>(A New \ Production System)
  <INTRO_SUBTITLE>(Introduction and Overview)
```

The following example specifies a slide that uses a numbered list.

```
3 <SLIDE>
  <TITLE>(WORK FLOW)
  <LIST>(NUMBERED)
  <LE>Create the File
  <LE>Run the Checker
  <ENDLIST>
```

OVERHEADS Doctype Tag Reference

<SUBTITLE>

<SUBTITLE>

Specifies a secondary title for a new slide.

SYNTAX

<SUBTITLE> (*title line-1*[\ *title line-2*] . . . [\ *title line-4*])

ARGUMENTS

title line-n

Specifies up to four lines of text for the secondary slide title.

related tags

- <TITLE>
-

DESCRIPTION

The <SUBTITLE> tag specifies a secondary title for a new slide. This subtitle may have up to four separate lines. Each of the subtitle lines is centered on the output page. Use the <TITLE> tag to create a main title for an overhead slide.

EXAMPLE

The following example is for an overhead slide that begins with a main title followed by a subtitle.

```
<SLIDE>
<TITLE>(FILE PROCESSING)
<SUBTITLE>(CONCEPTS AND INSTRUCTIONS)
```

<TEXT_SIZE>

Changes the size of type used in the context of topics, tables, and lists on a single slide.

SYNTAX

<TEXT_SIZE>({ *SMALL*
 REGULAR
 TABLE })

ARGUMENTS

SMALL

Reduces the standard point size of type used in topics and lists.

REGULAR

Increases the size of type in tables to that of standard text.

TABLE

Reduces text to the smallest font size available in the OVERHEADS doctype, which is the standard font size used inside of tables.

related tags

- <SLIDE>

required terminator

<ENDTEXT_SIZE>

DESCRIPTION

The <TEXT_SIZE> tag changes the size of type used in the context of topics, tables, and lists on a single slide. This tag alters the typeface used by text in the following contexts:

- Text specified in a list using the global <LIST> tag
- Text specified as an argument to the <TOPIC> tag
- Text specified in the context of the global <TABLE> tag

The <TEXT_SIZE> tag alters the default size of type only on a single slide. The following slide will have the default type sizes. To alter the type size for more than one slide, use the <TEXT_SIZE> tag on each of the slides. If you use the <TEXT_SIZE> tag in a table, place it right after the <TABLE_SETUP> tag.

```
DECLIT AA VAX JT86B
```

```
VAX DOCUMENT using doctypes  
and related tags
```

OVERHEADS Doctype Tag Reference

<TEXT_SIZE>

EXAMPLE

The following example illustrates each of the various formats available using the <TEXT_SIZE> tag.

```
<SLIDE>
<TOPIC>(ANIMALS THAT MAKE GOOD PETS)
<list>(UNNUMBERED)
<LE>Rabbits---Small, furry, generally best as pets for children
growing up in a non-urban setting.
<p>
<TEXT_SIZE>(small)
Rabbits are typically not good pets in urban settings because of their
extreme sensitivity to noise and because of their love of the outdoors.
<ENDTEXT_SIZE>
<LE>Dogs---Come in assorted shapes and sizes: a general purpose pet.
<p>
<TEXT_SIZE>(TABLE)
Dogs are man's (and woman's) best friend.
<ENDTEXT_SIZE>
<LE>Cats---Cats are ideal pets for apartment dwellers.
<p>
<TEXT_SIZE>(regular)
Cats should not be pets in households that already have tropical fish for pets.
<ENDTEXT_SIZE>
<ENDLIST>
```

<TITLE>

Specifies a title for a new slide.

SYNTAX

<TITLE> (*title line-1*[\ *title line-2*] . . . [\ *title line-4*])

ARGUMENTS

title line-n

Specifies up to four lines of text for a main slide title.

related tags

- <SUBTITLE>
-

DESCRIPTION

The <TITLE> tag specifies a title for a new slide. This title may have up to four separate lines. Each of the title lines centers on the slide. Use the <SUBTITLE> tag to create a subordinate title for an overhead slide.

EXAMPLES

The following example shows a tag sequence that begins a new slide and specifies a single-line title.

```
1 <SLIDE>
  <TITLE>(System Components)
  <TOPIC>(Parser)
  <TOPIC>(Interpreter)
```

This example shows how to have a 3-line title, with each line centered.

```
2 <TITLE>(THE \DEVELOPMENT \CYCLE)
```

OVERHEADS Doctype Tag Reference

<TOPIC>

<TOPIC>

Specifies a line of topic text for a slide.

SYNTAX

<TOPIC> (*topic text*)

ARGUMENTS

topic text
Specifies text for the topic.

related tags

- <SLIDE>
- <TEXT_SIZE>

DESCRIPTION

The <TOPIC> tag specifies a line of topic text for a slide. The text of the <TOPIC> tag begins at the left margin and has a smaller type font than the font used by the <TITLE> tag. Alter the size of the topic text with the <TEXT_SIZE> tag.

EXAMPLE

The following example illustrates a slide with a title, a topic sentence, and an unnumbered list.

```
<SLIDE>
<TITLE>(THE \DEVELOPMENT \CYCLE)
<TOPIC>(WHO)
<LIST>(UNNUMBERED)
<LE>WRITERS
<LE>EDITORS
<LE>COMPOSITORS AND ARTISTS
<ENDLIST>
```

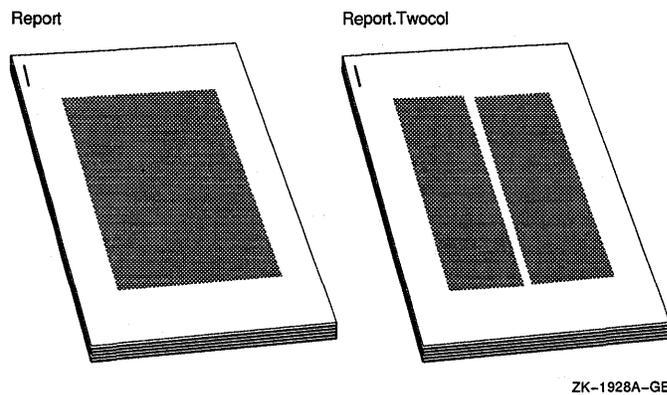
9

Using the REPORT Doctype

The REPORT doctype has two designs, shown in Figure 9–1, and is used for general-purpose documents such as reports and formal outlines.

- **REPORT**
Creates an $8\frac{1}{2} \times 11$ -inch format with unruled numbered and unnumbered headings.
- **REPORT.TWOCOL**
Creates the same format as REPORT, but places the text in two columns.

Figure 9–1 REPORT Doctype Designs



9.1

Characteristics of the REPORT Design

Table 9–1 lists the page layout of the REPORT doctype design. Table 9–2 lists the page layout of the REPORT.TWOCOL doctype designs.

Table 9–1 Page Layout of the REPORT Doctype Design

Page Layout Characteristics	
Running heads	None ¹
Running feet	Page number, title text optional ¹
Page numbering	Sequential ¹

¹You can modify this characteristic

Using the REPORT Doctype

Table 9–1 (Cont.) Page Layout of the REPORT Doctype Design

Page Layout Characteristics	
Trim size	8 1/2 x 11 inches
Right margin	Justified

Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left
Figures, tables, and examples	Numbered

Table 9–2 Page Layout of the REPORT.TWOCOL Doctype Design

Page Layout Characteristics	
Running heads	None ¹
Running feet	Page number, title text optional ¹
Page numbering	Sequential
Trim size	8 1/2 x 11 inches
Right margin	Justified

Text Element Characteristics	
Headings	Unnumbered
Paragraphs	First line indent
Figures, tables, and examples	Numbered

¹You can modify this characteristic

This doctype accepts the full range of VAX DOCUMENT global tags (see the *VAX DOCUMENT Using Global Tags* for more information on the global tags). The REPORT doctype also provides additional tags that let you perform the following functions:

- Create headings or modify the default attributes of the REPORT doctype.
- Create signature lines and list author information in the context of the global <FRONT_MATTER> tag.
- Create formal outlines in the context of the <OUTLINE> tag.

You process a file with the REPORT doctype by using the REPORT or REPORT.TWOCOL doctype keyword on the VAX DOCUMENT command line.

```
$ DOC/LIST/CONTENTS/SYMBOLS=MY_SYMBOLS.SDML-
_$ MyFile.sdml REPORT LN03
```

See Chapter 2 for information on special formatting considerations of a 2-column doctype design, and for suggestions on improving final output.

Table 9–3 summarizes the tags available in the REPORT doctype and provides a brief description of each tag. Section 9.4 contains the reference information on the tags listed in this table.

Table 9–3 Tags Available in the REPORT Doctype

Tag Name	Description
Tags Available in the Front Matter	
<AUTHOR>	Places the name of an author and up to two additional lines of information about the author on the output page.
<BYLINE>	Creates a rule to be used as a signature line, and places the name of the signatory beneath the line.
<SIGNATURES>	Begins a listing of signature lines created by the <BYLINE> tag. Optionally, you can use this tag to begin the listing of signature lines on a new page.
Tags Available Throughout the Document	
<COLUMN>	Specifies that a new column of output should begin in a 2-column doctype.
<DOCUMENT_ATTRIBUTES>	Modifies the numbering of pages and formal elements in the document.
<RUNNING_FEET>	Places a heading at the bottom of each page.
<RUNNING_TITLE>	Places a heading at the top of each page.
<SECTION>	Begins a new page and places an unnumbered heading at the top of the new page on the left margin.
Tags Available to Create Outlines	
<LEVEL>	Specifies an entry in an outline.
<OUTLINE>	Enables the <LEVEL> and <SHOW_LEVELS> tags and specifies a title for the outline.
<SHOW_LEVELS>	Emphasizes text in the outline using either bolding or italics.

9.2 Sample Use of the REPORT Doctype Tags

This section contains an example of the first pages of a report created using the REPORT doctype tags. This report includes a front matter section and an outline in the body of the report. Note how the outline and front matter tags are used in this example. You may find this sample useful in understanding how the tags all fit together to create reports and other general-purpose documents.

The SDML code for the report is shown first, followed by the output from that SDML code.

Using the REPORT Doctype

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(Equipment Usage in this Company)
<RUNNING_TITLE>(Equipment Used)
<RUNNING_FEET>(A Valuable Resource)
<ABSTRACT>
This is an internal report on equipment usage during
the period (May 1989 - November 1989).
<ENDABSTRACT>
<AUTHOR>(Thomas A. Smith\Comptroller\Eastern Division)
<SIGNATURES>
<BYLINE>(T. A. Smith)
<BYLINE>(John Whorfin\Accounting Consultant)
<DATE>(26-November-1989)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
<CHAPTER>(Equipment Usage Summary)
<P>Equipment usage is a very important quantity to monitor.
If equipment is not used, it is a wasted resource. If equipment is over-used,
it tends to break sooner, and means that people must wait to use it. If people
are waiting, they are not being as productive as they might otherwise be.
<P>The following sections summarize equipment usage in various departments.

<HEAD1>(Usage of Official Vehicles\28_UsageofOfficialVehicles)
<P>
Official vehicle usage is listed in a separate report CORP-AUTO-1439u2.
This report is organized as in the following outline. Note that there
are two new categories in the report. These categories are italicized
in the following outline.
<OUTLINE>(Outline of Report\CORP-AUTO-1439u2\Motor Vehicle Usage)
<LEVEL>(1\Four wheeled Vehicles)
<LEVEL>(2\Cars)
<LEVEL>(2\Trucks)
<SHOW_LEVELS>(ITALIC)
<LEVEL>(3\Heavy trucks)
<LEVEL>(3\Light trucks (less than 2 ton))
<SHOW_LEVELS>(OFF)
<LEVEL>(2\Vans)
<ENDOUTLINE>
```

Figure 9-2 and Figure 9-3 show the corresponding output from that SDML file when processed with the REPORT keyword. Comparing these samples may be helpful in understanding how to use these tags to create reports. Should you wish to create this output yourself, you can obtain file REPORT_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Figure 9-2 REPORT Doctype Output Example, Title Page

Equipment Usage in this Company

This is an internal report on equipment usage during the period (May 1989 - November 1989).

Thomas A. Smith
Comptroller
Eastern Division

T. A. Smith

John Whorfin—Accounting Consultant 26-November-1989

Digital Equipment Corporation

Using the REPORT Doctype

Figure 9-3 REPORT Doctype Output Example, Interior Page

CHAPTER 1

EQUIPMENT USAGE SUMMARY

Equipment usage is a very important quantity to monitor. If equipment is not used, it is a wasted resource. If equipment is over-used, it tends to break sooner, and means that people must wait to use it. If people are waiting, they are not being as productive as they might otherwise be.

The following sections summarize equipment usage in various departments.

1.1 Usage of Official Vehicles

Official vehicle usage is listed in a separate report CORP-AUTO-1439u2. This report is organized as in the following outline. Note that there are two new categories in the report. These categories are italicized in the following outline.

Outline of Report
CORP-AUTO-1439u2
Motor Vehicle Usage

- I. Four wheeled Vehicles
 - A. Cars
 - B. Trucks
 - 1. *Heavy trucks*
 - 2. *Light trucks (less than 2 ton)*
 - C. Vans

9.3

A Sample Use of the REPORT.TWOCOL Doctype Tags

This section shows the preceding example modified to show how to use the <COLUMN> tag and the global <CHEAD> tag.

```

<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(Equipment Usage in this Company)
<RUNNING_TITLE>(Equipment Used)
<RUNNING_FEET>(A Valuable Resource)
<ABSTRACT>
This is an internal report on equipment usage during
the period (May 1989 - November 1989).
<ENDABSTRACT>
<AUTHOR>(Thomas A. Smith\Comptroller\Eastern Division)
<SIGNATURES>
<BYLINE>(T. A. Smith)
<BYLINE>(John Whorfin\Accounting Consultant)
<DATE>(26-November-1989)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
<CHAPTER>(Equipment Usage Summary)
<P>Equipment usage is a very important quantity to monitor.
If equipment is not used, it is a wasted resource. If equipment is over-used,
it tends to break sooner, and means that people must wait to use it. If people
are waiting, they are not being as productive as they might otherwise be.
<P>The following sections summarize equipment usage in various departments.

<HEAD1>(Usage of Official Vehicles\28_UsageofOfficialVehicles)
<P>
Official vehicle usage is listed in a separate report CORP-AUTO-1439u2.
This report is organized as in the following outline. Note that there
are two new categories in the report. These categories are italicized
in the following outline.
<OUTLINE>(Outline of Report\CORP-AUTO-1439u2\Motor Vehicle Usage)
<LEVEL>(1\Four wheeled Vehicles)
<LEVEL>(2\Cars)
<LEVEL>(2\Trucks)
<SHOW_LEVELS>(ITALIC)
<LEVEL>(3\Heavy trucks)
<LEVEL>(3\Light trucks (less than 2 ton))
<SHOW_LEVELS>(OFF)
<LEVEL>(2\Vans)
<ENDOUTLINE>
<COLUMN>
<CHEAD>(A Valuable Resource)
<P>
We must all be concerned about the safe handling and preventive
maintenance of all of our vehicles...

```

Figure 9-4 shows the corresponding output from that SDML file when processed with the REPORT.TWOCOL keyword. Comparing these samples may be helpful in understanding how to use these tags to create 2-column reports. Should you wish to create this output yourself, you can obtain file REPORT_TWOCOL_SAMPLE.SDML from directory DOC\$ROOT:[EXAMPLES].

Using the REPORT Doctype

Figure 9-4 REPORT.TWOCOL Doctype Output Example, Title Page

Equipment Usage in this Company

This is an internal report on equipment usage during the period (May 1989 - November 1989).

Thomas A. Smith
Comptroller
Eastern Division

T. A. Smith

John Whorfin—Accounting Consultant 26-November-1989
Digital Equipment Corporation

Figure 9-5 REPORT.TWOCOL Doctype Output Example, Interior Page

CHAPTER 1

EQUIPMENT USAGE SUMMARY

Equipment usage is a very important quantity to monitor. If equipment is not used, it is a wasted resource. If equipment is over-used, it tends to break sooner, and means that people must wait to use it. If people are waiting, they are not being as productive as they might otherwise be.

The following sections summarize equipment usage in various departments.

1.1 USAGE OF OFFICIAL VEHICLES

Official vehicle usage is listed in a separate report CORP-AUTO-1439u2. This report is organized as in the following outline. Note that there are two new categories in the report. These categories are italicized in the following outline.

Outline of Report
CORP-AUTO-1439u2
Motor Vehicle Usage

- I. Four wheeled Vehicles
 - A. Cars
 - B. Trucks
 - 1. *Heavy trucks*
 - 2. *Light trucks (less than 2 ton)*
 - C. Vans

A Valuable Resource

We must all be concerned about the safe handling and preventive maintenance of all of our vehicles...

9.4 REPORT Doctype Tag Reference

This part of Chapter 9 provides reference information on all the tags specific to the REPORT doctype.

<AUTHOR>

Places the name of an author and one or two additional lines of information about the author in the front matter portion of a document.

SYNTAX <AUTHOR>(author name [\ author info-1] [\ author info-2])

ARGUMENTS *author name*
Specifies the name of the author.

author info-n
This is an optional argument. It specifies any additional information about the author below the author's name. Information you specify as *author info-1* outputs above information you specify as *author info-2*.

related tags

- <BYLINE>
- <SIGNATURES>
- The global <FRONT_MATTER> tag

restrictions Valid only in the context of the global <FRONT_MATTER> tag in the REPORT doctype.

DESCRIPTION The <AUTHOR> tag places the name of an author and one or two additional lines of information about the author in the front matter portion of a document. This tag accepts two optional arguments to provide the additional information about the author.

If you want a signatory line for the author in the front matter, use the <SIGNATURES> and <BYLINE> tags. See the descriptions of those tags in this chapter for more information.

REPORT Doctype Tag Reference

<AUTHOR>

EXAMPLE

The following example shows how you can use the <AUTHOR> tag in the front matter of a document. Note how the optional second argument to the <AUTHOR> tag specifies the author's title.

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(The NYUC Simulator Reference Manual)
<ORDER_NUMBER>(AA-Z0000-TE)
<ABSTRACT>
This manual describes the NYUC Simulator.
This program simulates a conversation between three people
by analyzing the syntactic and semantic components of three
related statements, and then synthesizing statements and responses
based upon these original statements.
<ENDABSTRACT>
<REVISION_INFO>(This revision is personally signed.)
<AUTHOR>(Mr. Jones\Research Head, STG Inc.)
<SIGNATURES>
<BYLINE>(Nat Jones\Author)
<DATE>(July 11, 1985)
<PRINT_DATE>(June 1987)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

<BYLINE>

Places a name and other optional information below a ruled line in a signature list.

SYNTAX

<BYLINE>(*name* [\ *additional info*])

ARGUMENTS***name***

Specifies the name of the signatory. This name outputs under the beginning of the signature line on the left side of the page.

additional info

This is an optional argument. It specifies any additional information about the signatory. This information outputs on the same line as the *name* argument with an em dash (—) between the two arguments.

related tags

- <AUTHOR>
- <SIGNATURES>
- The global <FRONT_MATTER> tag

restrictions

Valid only in the context of the global <FRONT_MATTER> tag and after the <SIGNATURES> tag.

DESCRIPTION

The <BYLINE> tag places a name and other optional information below a ruled line in a signature list. You can place additional information about the signer by using the *additional info* argument. Additional information formats to the right of the name of the signer, on the same line, separated by an em dash (—).

Use as many <BYLINE> tags as you want to create approval lines in the front matter of a document, as long as all these tags follow the <SIGNATURES> tag. Use the <SIGNATURES> tag to begin all the approval lines on a separate page of the front matter. See the <SIGNATURES> tag in this chapter for more information.

REPORT Doctype Tag Reference

<BYLINE>

EXAMPLE

The following example shows three occurrences of the <BYLINE> tag. The first two occurrences list the positions of the signers using the optional *additional info* argument. The third occurrence of the <BYLINE> tag omits the optional argument. Note that all three tags follow the <SIGNATURES> tag.

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(The NYUC Simulator Reference Manual)
<REVISION_INFO>(This revision is personally signed.)
<AUTHOR>(Mr. Jones\Research Head, STG Inc.)
<SIGNATURES>
<BYLINE>(Nat Jones\Author)
<BYLINE>(Cecil Mills\Co-author)
<BYLINE>(Matt Smith)
<DATE>(July 11, 1985)
<PRINT_DATE>(June 1987)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

<COLUMN>

In a 2-column doctype, specifies that a new column of output begins.

SYNTAX <COLUMN>

ARGUMENTS *None.*

related tags

- The global <FINAL_CLEANUP> tag

restrictions Valid only in a 2-column doctype.

DESCRIPTION The <COLUMN> tag, in a 2-column doctype, specifies that a new column of output begins. This causes the text immediately following the tag to be started in a new column. If this tag occurs in the left text column, the text immediately following it begins in the right text column. If this tag occurs in the right text column, the text immediately following it begins in the left column of the next page.

Use the <COLUMN> tag when you always want to begin a new column at that point in your text. You can use the COLUMN_BREAK argument to the global <FINAL_CLEANUP> tag to also specify a column break. However, only use it during the final processing of the 2-column document.

See Chapter 2 for more information on improving the formatting of a 2-column doctype such as REPORT.TWOCOL and ARTICLE.

REPORT Doctype Tag Reference

<COLUMN>

EXAMPLE

The following example shows how to use the <COLUMN> tag to begin a new text column. In this example, the writer wants the two descriptions to appear side by side, one in each column.

```
<HEAD>(Woodwind Instruments)
<P>Woodwind instruments have the following
attributes:
<LIST>(UNNUMBERED)
<LE>They are often made of wood, hence their name.
<LE>Musicians create sound using these instruments by causing a reed
to vibrate.
.
.
.
<ENDLIST>
<COLUMN>
<HEAD>(Brass Instruments)
<P>Brass instruments have the following
attributes:
<LIST>(UNNUMBERED)
<LE>They are often made of brass, hence their name.
<LE>Musicians create sound using these instruments by
vibrating (buzzing) their lips into a steel mouthpiece.
.
.
.
<ENDLIST>
```

<DOCUMENT_ATTRIBUTES>

Enables doctype-specific tags that override the default design format of the REPORT doctype.

SYNTAX <DOCUMENT_ATTRIBUTES>

ARGUMENTS *None.*

required terminator <ENDDOCUMENT_ATTRIBUTES>

DESCRIPTION The <DOCUMENT_ATTRIBUTES> tag enables doctype-specific tags that override the default design format of the REPORT doctype. This tag is used in three doctypes:

- ARTICLE
- REPORT
- SOFTWARE

The <DOCUMENT_ATTRIBUTES> tag enables a group of tags in each of these doctypes that allow you to modify the default format of that doctype. VAX DOCUMENT recognizes these tags only in the context of the <DOCUMENT_ATTRIBUTES> tag. If other VAX DOCUMENT tags occur in this context, VAX DOCUMENT ignores them, as if they had occurred in the context of a <COMMENT> tag.

Typically, use the <DOCUMENT_ATTRIBUTES> tag at the beginning of an input file (or in a file processed using the /INCLUDE qualifier on the VAX DOCUMENT command line) to alter the default format of a doctype for the processing of that entire file.

Book builds and **element builds** in this doctype do not save information about attributes, such as page numbers, specified with the <DOCUMENT_ATTRIBUTES> tag. To ensure that the same attributes are specified in both contexts, place <DOCUMENT_ATTRIBUTES> tags in a file that is included in both book and element builds. To do this, either place the <DOCUMENT_ATTRIBUTES> tag at the beginning of every element file, or use the /INCLUDE or /SYMBOLS qualifier to specify a file containing the <DOCUMENT_ATTRIBUTES> tag.

Table 9-4 summarizes the formatting tags enabled by the <DOCUMENT_ATTRIBUTES> tag in the REPORT doctype.

REPORT Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

Table 9-4 Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> tag

Formatting Tags	Description
<SET_HEADINGS>(UNNUMBERED) <SET_HEADINGS>(NUMBERED)	The <SET_HEADINGS> tag specifies whether the heading-level tags produce numbered or unnumbered headings. (<HEAD1>, <HEAD2>, and so on). By default, headings are not numbered in a document processed using the ARTICLE doctype. Use the <SET_HEADINGS>(NUMBERED) tag to specify numbered headings.

EXAMPLE

The following example is of a file that is to be processed under the REPORT doctype. This example shows how you use the <SET_PAGE_NUMBERING> and <SET_FORMAL_ELEMENT_NUMBERING> tags to create page and formal element numbering that is chapter-oriented rather than sequential. Note how the BY_CHAPTER argument is used by both tags to specify that numbering should be by chapter rather than sequential.

```
<DOCUMENT_ATTRIBUTES>  
<SET_PAGE_NUMBERING>(BY_CHAPTER)  
<SET_FORMAL_ELEMENT_NUMBERING>(BY_CHAPTER)  
<ENDDOCUMENT_ATTRIBUTES>
```

<LEVEL>

Specifies an outline entry and the organizational level of that outline entry.

SYNTAX <LEVEL>(level number \ entry text)

ARGUMENTS *level number*
Specifies the organizational level of the entry. This argument can be any whole number from 1 to 6.

entry text
Specifies the text for a particular level.

related tags

- <OUTLINE>
- <SHOW_LEVELS>

restrictions Valid only in the context of an <OUTLINE> tag.

DESCRIPTION The <LEVEL> tag specifies an outline entry and the organizational level of that outline entry. Top-level entries (those specified as <LEVEL>(1)) are marked using uppercase Roman numerals. At the lowest level, level 6, the entries are marked with lowercase letters enclosed in parentheses. The top level formats at the current left margin; each lower level indents from the level above it.

EXAMPLE The following example illustrates an outline created using the <LEVEL> tag in the context of the <OUTLINE> tag. Note how you indent the <LEVEL> tags in the SDML file to make the file easier to read and more maintainable.

```
<OUTLINE>(<EMPHASIS>(Maxillary Taxonomy)\An Enumeration of the
Maxillae\from a Dentition Perspective)
<LEVEL>(1\Historical introduction)
<LEVEL>(1\Dentition in various groups of vertebrates)
  <LEVEL>(2\Reptilia)
    <LEVEL>(3\Histology and development of reptile teeth)
      <LEVEL>(4\Survey of forms)
  .
  .
  .
```

REPORT Doctype Tag Reference

<OUTLINE>

<OUTLINE>

Begins an outline and specifies a title for the outline.

SYNTAX

<OUTLINE>*[(title line-1 [\ title line-2] [\ title line-3])]*

ARGUMENTS

title line-n

This is an optional argument. It specifies a title line. You can specify up to three title lines.

related tags

- <LEVEL>
- <SHOW_LEVELS>

required terminator

<ENDOUTLINE>

DESCRIPTION

The <OUTLINE> tag begins an outline and specifies a title for the outline.

An outline is a hierarchical list of numbered elements in which the hierarchy is conveyed to the reader through the letter or number on each outline entry and the indentation level. The <OUTLINE> tag begins an outline that permits up to six levels in the hierarchy. At the top level, level 1, the outline entries are marked with uppercase Roman numerals. At the lowest level, level 6, the outline entries are marked with lowercase letters enclosed in parentheses. The top level formats at the current left margin; each lower level indents from the level above it.

If you supply one or more title lines as arguments to the <OUTLINE> tag, the lines center above the outline. You may want to add emphasis to some or all of the title lines.

The outline usually aligns at the left margin of the text, but you can embed it in other tags—for example, after an <LE> tag, in which case the outline aligns with other list elements. Likewise, you can embed other tags in the outline, so that text or sublists can be interspersed with the outline entries.

An outline does not affect heading levels established with any of the heading tags (<HEAD1>, <HEAD2>, and so on). Outline entries do not appear in the table of contents.

EXAMPLE

The following example illustrates an outline created using the <OUTLINE> tag and the tags it enables. Note how you indent the <LEVEL> tags in the SDML file to make the file easier to read and more maintainable. This indentation in the SDML file has no effect on the output, which indents automatically according to the level specified in the <LEVEL> tags.

```
<OUTLINE>(<EMPHASIS>(Maxillary Taxonomy)\An Enumeration of the
Maxillae\from a Dentition Perspective)
<LEVEL>(1\Historical introduction)
<LEVEL>(1\Dentition in various groups of vertebrates)
  <LEVEL>(2\Reptilia)
    <LEVEL>(3\Histology and development of reptile teeth)
      <LEVEL>(4\Survey of forms)
  <LEVEL>(2\Mammalia)
    <LEVEL>(3\Histology and development of mammalian teeth)
      <LEVEL>(3\Survey of forms)
        <LEVEL>(4\Primates)
          <LEVEL>(5\Lemuroidea)
            <LEVEL>(5\Anthropoidea)
              <LEVEL>(6\Platyrrhini)
                <LEVEL>(6\Catarrhini)
          <LEVEL>(4\Carnivora)
            <LEVEL>(5\Creodonta)
              <LEVEL>(5\Fissipedia)
                <LEVEL>(6\Aeluroidea)
                  <LEVEL>(6\Arctoidea)
                <LEVEL>(5\Pinnipedia)
          <LEVEL>(4\Etc<hellipsis>)
<ENDOUTLINE>
```

REPORT Doctype Tag Reference

<RUNNING_FEET>

<RUNNING_FEET>

Creates a single line heading at the bottom of each page.

SYNTAX **<RUNNING_FEET>** (*title text*)

ARGUMENTS ***title text***
Specifies the text for the running feet.

related tags

- <CHAPTER>
- <RUNNING_TITLE>
- <SECTION>

DESCRIPTION The <RUNNING_FEET> tag creates a single line heading at the bottom of each page. This heading is called a footer because it appears at the foot of the page. When the same footer runs for several pages, the footers are collectively called running feet.

This tag accepts one argument, which is the text that should appear at the bottom of the page. This text is output exactly as entered, including spacing and capitalization.

Use the <RUNNING_TITLE> tag to create a heading at the top of the page.

Note that you can override headings established by the <RUNNING_FEET> and <RUNNING_TITLE> tags by a subsequent use of the <CHAPTER> tag or <SECTION> tag.

EXAMPLE The following example shows how to use the <RUNNING_FEET> tag to place the footer Getting the Piece of Paper at the bottom of each page. The running footer will be output exactly as entered.

```
<RUNNING_FEET>(Getting the Piece of Paper)
<HEAD>(Getting the Piece of Paper)
<p>
You can buy clean paper in most major supermarkets, department stores,
and hardware stores. You should try to get ruled paper so that
your letter will be neat and easy to read.
```

<RUNNING_TITLE>

Creates a 1- or 2-line running heading at the top of each page.

SYNTAX

$$\langle \text{RUNNING_TITLE} \rangle \left(\begin{array}{l} \text{OFF} \\ \text{title-1} [\backslash \text{title-2}] \\ [\backslash \text{FIRST_PAGE}] \end{array} \right)$$

ARGUMENTS

OFF

Specifies that any existing running titles created using the <RUNNING_TITLE> tag are disabled for the page on which this tag occurs and on any subsequent pages.

title-1

Specifies the text of a running title. If you specify a 2-line title, this title outputs on the upper title line.

title-2

This is an optional argument. It specifies the bottom line of a running title that has two lines.

FIRST_PAGE

This is an optional argument. It specifies that the running title is to be placed on the first output page. If you do not specify this keyword, the running title outputs on the page after the current page.

related tags

- <RUNNING_FEET>

DESCRIPTION

The <RUNNING_TITLE> tag creates a 1- or 2-line running heading at the top of each page. Use the FIRST_PAGE argument to the <RUNNING_TITLE> tag to begin the title lines on the first page of output, rather than on the page after the current page, as is the default.

Use the OFF argument to disable any existing running titles created using the <RUNNING_TITLE> tag. These titles are then disabled for the page on which this tag occurs and on any subsequent pages.

Use the <RUNNING_FEET> tag to create a heading that appears at the bottom of the page. See the <RUNNING_FEET> tag in this chapter for more information.

Override headings established by the <RUNNING_TITLE> and <RUNNING_FEET> tags by a subsequent use of the <CHAPTER> tag.

REPORT Doctype Tag Reference

<RUNNING_TITLE>

EXAMPLES

The following example shows how to use the <RUNNING_TITLE> tag to create the 2-line running title An E. B. Bartz Course: Writing Quality Correspondence. Note that because you use the FIRST_PAGE argument, the 2-line running title appears at the top of the first page.

1 <RUNNING_TITLE>(An E. B. Bartz Course:\Writing Quality Correspondence\FIRST_PAGE)
<HEAD>(How to Write a Letter\32_HowtoWriteaLetter)
<P>
The first thing that you should do in writing a letter is to get a clean piece of paper and a well-sharpened pencil.

The following example shows how you can disable a running title by using the OFF argument to the <RUNNING_TITLE> tag.

2 <COMMENT>(turn off running titles for the following example page)
<RUNNING_TITLE>(OFF)
<HEAD>(An Example of a Letter\33_AnExampleofaLetter)
.
.
.

<SECTION>

Begins a new page and creates a major heading at the left margin of that page.

SYNTAX

<SECTION>(heading text [\ symbol name])

ARGUMENTS

heading text

Specifies the text of the section heading.

symbol name

This is an optional argument. It specifies the name of the symbol used in all references to this heading.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- The global <CHAPTER> tag
- The global <CHEAD> tag
- The global <HEAD> tag
- The global <HEAD1> tag
- The global <REFERENCE> tag

DESCRIPTION

The <SECTION> tag begins a new page and creates a major heading at the left margin of that page. It is one of the three REPORT doctype-specific tags that create unnumbered headings. The other tags that produce unnumbered headings, the global <HEAD> tag and the global <CHEAD> tag, do not begin a new page of output.

EXAMPLE

The following example shows how to use the <SECTION> tag to begin a new page and place an unnumbered heading on that page. Note that this sample omits the *symbol name* argument to the <SECTION> tag, because the writer will not be referencing this section.

```
<SECTION>(Writing Personal Correspondence)
```

```
<P>
```

```
Writing personal correspondence is more fun and less formal than  
writing business correspondence, but many of the same rules apply.
```

```
<HEAD>(How to Write a Letter\34_HowtoWriteaLetter)
```

```
<P>
```

```
The first thing that you should do in writing a letter is to get a  
clean piece of paper and a well-sharpened pencil.
```

REPORT Doctype Tag Reference

<SHOW_LEVELS>

<SHOW_LEVELS>

Emphasizes text in an outline.

SYNTAX

<SHOW_LEVELS>({ *BOLD*
ITALIC
OFF })

ARGUMENTS

BOLD

Specifies that the *entry text* arguments given to all subsequent <LEVEL> tags output in a bold typeface.

ITALIC

Specifies that the *entry text* arguments given to all subsequent <LEVEL> tags output in an italic typeface.

OFF

Specifies that the *entry text* arguments given to all subsequent <LEVEL> tags output in the standard typeface; this is the default.

related tags

- <OUTLINE>
 - <LEVELS>
-

restrictions

Valid only in the context of an <OUTLINE> tag.

DESCRIPTION

The <SHOW_LEVELS> tag emphasizes text in an outline. The text emphasized is the text associated with one or more <LEVEL> tags. Such text outputs in a bold typeface if you use the BOLD argument, or in an italic typeface if you use the ITALIC argument.

The OFF argument disables the bolding or italicizing of the text used as an argument to the <LEVEL> tag. If you do not use the <SHOW_LEVELS> tag, or if you specify the OFF argument, the standard typeface is used in the outline.

See the <LEVEL> tag description in this chapter for more information on the <LEVEL> tag.

EXAMPLE

The following example shows an outline that uses the <SHOW_LEVELS> tag.

The entry text is italicized using the <SHOW_LEVELS>(ITALIC) tag beginning with the second-level entry Mammalia through the fourth-level entry Primates. The italicized text is then turned off using the OFF argument to the <SHOW_LEVELS> tag.

REPORT Doctype Tag Reference

<SHOW_LEVELS>

```
<OUTLINE>(<EMPHASIS>(Maxillary Taxonomy)\An Enumeration of the
Maxillae\from a Dentition Perspective)
<LEVEL>(1\Historical introduction)
<LEVEL>(1\Dentition in various groups of vertebrates)
  <LEVEL>(2\Reptilia)
    <LEVEL>(3\Histology and development of reptile teeth)
      <LEVEL>(4\Survey of forms)
        <COMMENT>(**Italicize the information covered on this weeks test**)
        <SHOW_LEVELS>(ITALIC)
  <LEVEL>(2\Mammalia)
    <LEVEL>(3\Histology and development of mammalian teeth)
      <LEVEL>(3\Survey of forms)
        <LEVEL>(4\Primates)
          <COMMENT>(**turn off italicization**)
          <SHOW_LEVELS>(OFF)
        <LEVEL>(5\Lemuroidea)
        <LEVEL>(5\Anthropoidea)
        <LEVEL>(6\Platyrrhini)
        <LEVEL>(6\Catarrhini)
      <LEVEL>(4\Carnivora)
        <LEVEL>(5\Creodonta)
        <LEVEL>(5\Fissipedia)
        <LEVEL>(6\Aeluroidea)
        <LEVEL>(6\Arctoidea)
        <LEVEL>(5\Pinnipedia)
        <LEVEL>(4\Etc<hellipsis>)
<ENDOUTLINE>
```

REPORT Doctype Tag Reference

<SIGNATURES>

<SIGNATURES>

Begins a list of signatures that appear in the front matter of a document.

SYNTAX

<SIGNATURES>[(NEWPAGE)]

ARGUMENTS

NEWPAGE

This is an optional argument. It specifies that the signature list begins on a new page.

related tags

- <AUTHOR>
- <BYLINE>
- The global <FRONT_MATTER> tag

restrictions

Valid only following the global <FRONT_MATTER> tag.

DESCRIPTION

The <SIGNATURES> tag begins a list of signatures that appear in the front matter of a document. You list each person's name by using the <BYLINE> tag following the <SIGNATURES> tag. The <BYLINE> tag places the name of the person, and additional information about that person (such as their title or affiliation), below a line on which the person is to sign.

See the reference description of the <BYLINE> tag for more information on that tag.

EXAMPLE

The following example shows the <SIGNATURES> tag beginning a list of signature lines. Note how the <BYLINE> tag is used to create each signature line.

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(The NYUC Simulator Reference Manual)
<REVISION_INFO>(This revision is personally signed.)
<AUTHOR>(Dr. Julian Jones\Research Head, STG Inc.)
<SIGNATURES>
<BYLINE>(Julian Jones\Author)
<BYLINE>(Cecil Mills\Co-author)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

10 Using the SOFTWARE Doctype

The SOFTWARE doctype has six designs for printed software documentation, shown in Figure 10–1, and one design for Bookreader documentation.

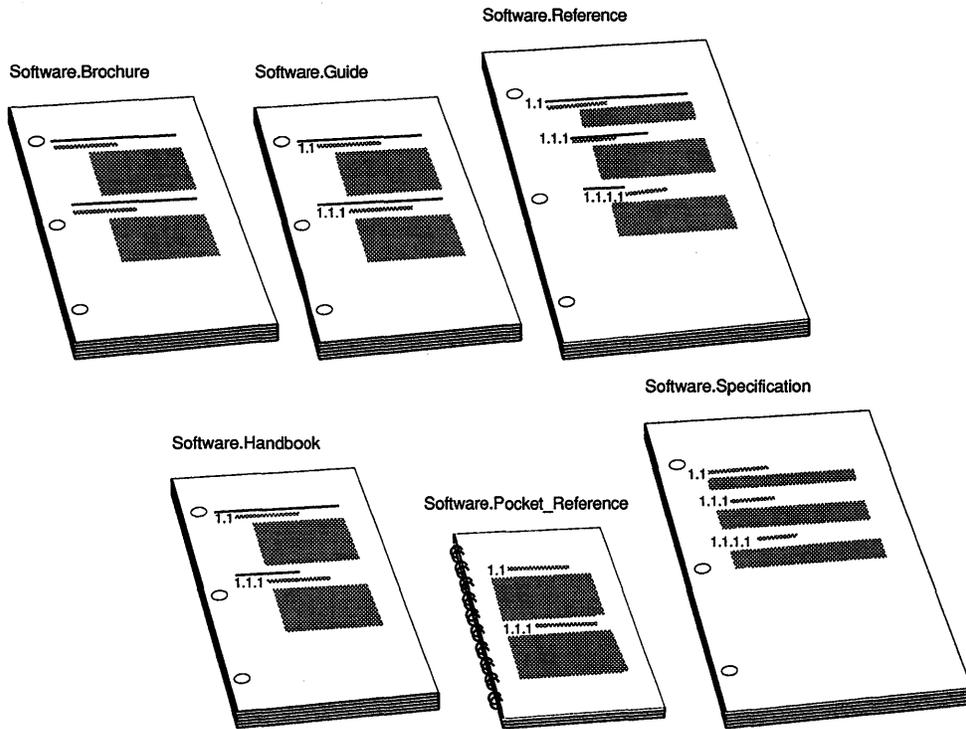
- **SOFTWARE.BROCHURE**
Creates a brochure in a 7 × 9-inch format with unnumbered headings.
- **SOFTWARE.GUIDE**
Creates a users' guide in a 7 × 9-inch format with numbered headings.
- **SOFTWARE.HANDBOOK**
Creates a handbook in a 7 × 9-inch format with numbered headings.
- **SOFTWARE.POCKET_REFERENCE**
Creates a pocket reference in a 5½ × 7-inch format with numbered headings.
- **SOFTWARE.REFERENCE**
Creates a reference manual in an 8½ × 11-inch format with numbered headings.
- **SOFTWARE.SPECIFICATION**
Creates a specification in an 8½ × 11-inch format with numbered headings.
- **SOFTWARE.ONLINE**
Creates an online reference manual in a 5.9 × 6.6-inch format with numbered headings and ragged right margin. This design is solely for Bookreader display. Refer to *VAX DOCUMENT Producing Online and Printed Documentation* for full information about producing Bookreader documentation.

Note: For simplicity, this chapter refers only to the SOFTWARE doctype whenever the discussion is appropriate to all SOFTWARE designs.

The SOFTWARE doctype designs differ primarily in size of page, in margins and rules, and in suitability for documenting tutorial or reference material.

Using the SOFTWARE Doctype

Figure 10-1 SOFTWARE Doctype Designs



ZK-1929A-GE

Using the SOFTWARE Doctype Characteristics of the SOFTWARE Designs

10.1 Characteristics of the SOFTWARE Designs

The following tables list the page layouts of the SOFTWARE doctype designs for printed documentation.

- Table 10–1 is for the SOFTWARE.BROCHURE design.
- Table 10–2 is for the SOFTWARE.GUIDE design.
- Table 10–3 is for the SOFTWARE.HANDBOOK design.
- Table 10–4 is for the SOFTWARE.POCKET_REFERENCE design.
- Table 10–5 is for the SOFTWARE.REFERENCE design.
- Table 10–6 is for the SOFTWARE.SPECIFICATION design.

Table 10–1 Page Layout of the SOFTWARE.BROCHURE Doctype Design

Page Layout Characteristics	
Running heads	None
Running feet	Chapter title text ¹ and page number
Page numbering	Chapter oriented
Trim size	7 x 9
Gutter width	6 picas
Right margin	Unjustified (Ragged right)

Text Element Characteristics	
Headings	Unnumbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

¹If you give the global <TITLE> tag a second argument, title text-2, this second argument replaces the chapter title text as the running footer.

Table 10–2 Page Layout of the SOFTWARE.GUIDE Doctype Design

Page Layout Characteristics	
Running heads	Chapter title text
Running feet	Chapter number and page number
Page numbering	Chapter oriented
Trim size	7 x 9 inches
Gutter width	5 picas
Right margin	Unjustified (Ragged right)

Using the SOFTWARE Doctype

Characteristics of the SOFTWARE Designs

Table 10–2 (Cont.) Page Layout of the SOFTWARE.GUIDE Doctype Design

Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

Table 10–3 Page Layout of the SOFTWARE.HANDBOOK Doctype Design

Page Layout Characteristics	
Running heads	None
Running feet	Chapter title text and page number
Page numbering	Chapter oriented
Trim size	7 x 9 inches
Gutter width	6 picas
Right margin	Unjustified (Ragged right)
Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

Table 10–4 Page Layout of the SOFTWARE.POCKET_REFERENCE Doctype Design

Page Layout Characteristics	
Trim size	5 1/2 x 7 inches
Gutter width	0
Right margin	Unjustified (Ragged right)
Text Element Characteristics	
Headings	Numbered, rule only under <HEAD1>
Paragraphs	Flush left, no first line indent
Figures, tables, and examples	Numbered; table of contents entry

Using the SOFTWARE Doctype Characteristics of the SOFTWARE Designs

Table 10–5 Page Layout of the SOFTWARE.REFERENCE Doctype Design

Page Layout Characteristics	
Running heads	Chapter title text
Running feet	Page number
Page numbering	Chapter oriented
Trim size	8 1/2 x 11 inches
Gutter width	9.5 picas
Right margin	Unjustified (Ragged right)

Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left at gutter width
Figures, tables, and examples	Numbered, table of contents entry

Table 10–6 Page Layout of the SOFTWARE.SPECIFICATION Doctype Design

Page Layout Characteristics	
Trim size	8 1/2 x 11 inches
Gutter width	0
Right margin	Unjustified (Ragged right)

Text Element Characteristics	
Headings	Numbered
Paragraphs	Flush left, no first line indent
Figures, tables, and examples	Numbered; table of contents entry

The following example shows how to process a file named MYSOFTWARE.SDML with the SOFTWARE.GUIDE doctype:

```
$ DOCUMENT mysoftware SOFTWARE.GUIDE LN03
```

Of the six available designs, SOFTWARE.REFERENCE is the default. To use this doctype, you need only enter SOFTWARE (or a truncated form of SOFTWARE) as a keyword on your DOCUMENT command line.

The SOFTWARE doctype provides four templates for documenting reference information. In any of the six SOFTWARE doctype designs, you can use one or more of the following **reference templates**:

- The Command template
- The Routine template

Using the SOFTWARE Doctype

Characteristics of the SOFTWARE Designs

- The Statement template
- The Tag template

SOFTWARE doctype tags fall into these categories:

- The specific tags used for each SOFTWARE doctype

The SOFTWARE doctype-specific tags are available only in the SOFTWARE doctype and let you create basic writing elements you need to describe computer software. See Section 10.19 for detailed information on these tags.

- The groups of tags specific to each of the four reference templates

The reference template tags are available only in the reference templates of the SOFTWARE doctype. With these tags, you can specify the format of a software command, the restrictions on such commands, prompts used in an interactive environment, and so on. See Section 10.8 for detailed information on these tags.

10.2 Common Software Description Tasks

The SOFTWARE doctype-specific tags let you describe software elements in structured reference material and in less structured tutorial material. Use these tags to describe the following, as discussed in following sections:

- Terminal keys and keypads
- Code fragments and their results
- Software messages
- Software arguments, parameters, and qualifiers
- Interactive terminal sessions

10.3 Documenting Terminal Keys and Keypads

The SOFTWARE doctype contains several tags that let you accurately represent terminal keys, keypads, and key names both in text and in examples. These tags fall into two groups:

- Tags that describe keys used throughout the SOFTWARE doctype.
- Tags used to create keypad diagrams used only in the context of the <KEYPAD_SECTION> tag in the SOFTWARE doctype.

Using the SOFTWARE Doctype Documenting Terminal Keys and Keypads

10.3.1 Describing Individual Keys

Use the following tags to label and describe certain keys and key sequences that appear on keyboards and keypads.

<code><CPOS></code>	Creates a typographically distinct marking for the cursor position in an example.
<code><DELETE_KEY></code>	Creates a special character that represents the DELETE key used on most keyboards.
<code><GRAPHIC></code>	Creates a specially formatted character out of two characters specified as arguments.
<code><KEY></code>	Creates a label for a key from the keyboard or keypad.
<code><KEY_NAME></code>	Creates a label for a key name.
<code><KEY_SEQUENCE></code>	Creates a section for depicting a sequence of keys.

Using the `<CPOS>` Tag

Use the `<CPOS>` tag to mark the position of the cursor in a terminal example. The cursor is depicted as the underline character. The following example shows how to use the `<CPOS>` tag.

```
<P>
Correct the directory specification to <QUOTE>([TEXTFILES]) by
moving the cursor to the misspelled letter in the directory
specification, as in the following example:
<DISPLAY>
$ COPY ABC.TXT [T<CPOS>(R)XTFILES]
<ENDDISPLAY>
```

This example produces the following output:

Correct the directory specification to “[TEXTFILES]” by moving the cursor to the misspelled letter in the directory specification, as in the following example:

```
$ COPY ABC.TXT [TRXTFILES]
```

Using the `<DELETE_KEY>` Tag

Use the `<DELETE_KEY>` tag to create the character used on most terminal and typewriter keyboards for the DELETE key. The following example shows how to use the `<DELETE_KEY>` tag.

```
<P>
Press the DELETE key (<DELETE_KEY>) to delete a character.
```

This example produces the following output:

Press the DELETE key (<X>) to delete a character.

Using the `<GRAPHIC>` Tag

Use the `<GRAPHIC>` tag to create special graphic characters that appear on the terminal screen. The `<GRAPHIC>` tag accepts two characters as arguments and formats them next to each other, with the second character formatted slightly below the first character. This tag lets you create representations of the linefeed and formfeed characters, as well as other similarly formatted characters.

Using the SOFTWARE Doctype Documenting Terminal Keys and Keypads

The following example shows how to use the <GRAPHIC> tag.

```
<P>  
Two special characters that will appear in your editing session  
are the linefeed (<GRAPHIC>(L\F)) and the formfeed (<GRAPHIC>(F\F))  
characters.
```

This example produces the following output:

Two special characters that will appear in your editing
session are the linefeed (L_F) and the formfeed (F_F)
characters.

Using the <KEY> Tag

Use the <KEY> tag to represent a terminal key, either in text or in an example. The <KEY> tag accepts a *key label* argument, which specifies the name of the key.

To represent a terminal key in text, use the TEXT keyword argument to the <KEY> tag to place angle brackets before and after the key label. To represent a terminal key in an example, use the BOX keyword argument to the <KEY> tag to place the key label in a box that resembles a key. If you specify neither the TEXT nor the BOX keyword, the default format is BOX.

Note that the <KEY> tag is used in the context of the <KEY_SEQUENCE> tag. See the description of the <KEY_SEQUENCE> tag in this section for more information on that tag.

The following example shows how to code the <KEY> tag, as it would appear in text using the TEXT keyword argument, and as it would appear in an example using the BOX keyword argument.

```
<P>  
You should now press the <KEY>(RETURN\TEXT) key to insert a line.  
<KEY_SEQUENCE>  
<KEY>(RETURN\BOX)  
<ENDKEY_SEQUENCE>
```

This example produces the following output:

You should now press the <RETURN> key to insert a line.

RETURN

Using the <KEY_NAME> Tag

Use the <KEY_NAME> tag to differentiate the name of a key from the other output in an example or in text.

The following example shows how to use the <KEY_NAME> tag.

```
<P>  
Press the <KEY_NAME>(HELP) or the <KEY_NAME>(DO) key for more information.
```

This example produces the following output:

Press the HELP or the DO key for more information.

Using the SOFTWARE Doctype Documenting Terminal Keys and Keypads

Using the <KEY_SEQUENCE> Tag

Use the <KEY_SEQUENCE> tag to give an example of a sequence of keys. For example, you might want to describe the sequence of keys needed to exit a text editor.

The <KEY_SEQUENCE> tag enables the <KEY_PLUS> and the <KEY_TYPE> tags to make creating such key sequences easier. The <KEY_PLUS> tag creates a plus sign (+) between two keys, and the <KEY_TYPE> tag lets you associate a key sequence with some textual information, such as a terminal type. In addition to these two tags, you can use the <KEY> tag in the context of the <KEY_SEQUENCE> tag.

When you use the <KEY> tag in a key sequence, it accepts an additional argument, which lets you place two key labels rather than one inside a box or angle brackets. When you use two key labels, they are stacked together with the first argument placed on the top. This extra argument makes it possible to specify keys that use two stacked words as their label, such as the Next Screen key.

The following code fragment contains a series of key examples in a <KEY_SEQUENCE> section. Note that the <KEY> tag is used in two contexts in this example. In the context of the <KEY_SEQUENCE> tag, the <KEY> tag accepts two key label arguments. Outside the context of the <KEY_SEQUENCE> tag, the <KEY> tag accepts only a single-key label argument.

Note also that the first <KEY> tag is specified with no keyword argument, so that the default BOX is used. The second <KEY> tag explicitly uses the BOX keyword to specify BOX formatting. The third <KEY> tag specifies the TEXT keyword argument.

```
<P>
You would use the following sequence of keys:
<KEY_SEQUENCE>
<KEY>(Next\Screen) <KEY_PLUS> <KEY>(PF3\BOX)
<ENDKEY_SEQUENCE>
<P>
These keys are not associated with the <key>(WHITE\TEXT) keys.
```

This example produces the following output:

You would use the following sequence of keys:

Next
Screen

 +

PF3

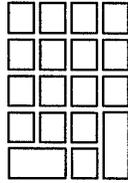
These keys are not associated with the <WHITE> keys.

10.3.2 Describing Keypads and Keypad Keys

A keypad is a group of keys separate from those on the typing keyboard. These keys are typically used for special applications, such as editing, data

Using the SOFTWARE Doctype Documenting Terminal Keys and Keypads

entry, or cursor movement. The following diagram shows such a keypad.



Showing keypads and keypad keys is often difficult because it involves preparing artwork by hand. Using the SOFTWARE doctype, you can prepare your own keypad and keypad key diagrams by using the `<KEYPAD_SECTION>` tag and the tags it enables.

The `<KEYPAD_SECTION>` tag begins a section in which keypad diagrams can be created and described. This section terminates with the `<ENDKEYPAD_SECTION>` tag. Each keypad or portion of a keypad is created using the following tags:

- `<KEYPAD>`

Begins a single illustration of a keypad or a portion of a keypad, and optionally allows a title to be placed on each illustration. A keypad cannot be more than four columns wide or more than five rows long. This tag is terminated by the `<ENDKEYPAD>` tag. You cannot create more than one keypad in a keypad section using the `<KEYPAD>` and `<ENDKEYPAD>` tags.

- `<KEYPAD_ROW>`

Creates a 4-column keypad row for all but the bottom row of the keypad.

- `<KEYPAD_ENDROW>`

Creates a special 3-column keypad row for the larger keys on the bottom row of the keypad.

If you use the `DISPLAY` keyword argument with the `<KEYPAD>` tag, you can specify arguments to the `<KEYPAD_ROW>` and `<KEYPAD_ENDROW>` tags that place text on the appropriate key in the keypad diagram.

If you do not use the `DISPLAY` keyword, you can specify only one of three keywords as arguments to the `<KEYPAD_ROW>` and `<KEYPAD_ENDROW>` tags.

The following keywords are accepted by the `<KEYPAD_ROW>` and `<KEYPAD_ENDROW>` tags:

- `OPEN`—Specifies that the key in that column is to be left blank. `OPEN` is the default.
- `CLOSED`—Specifies that the key in that column is to be shaded in.
- `NONE`—Specifies that no key should be drawn in that column.

The `<KEYPAD_ROW>` and `<KEYPAD_ENDROW>` tags accept the same keyword arguments, which let you specify whether a key should be drawn, and whether a key that is drawn should be shaded in. If you specify no keyword argument for a particular keypad column, the key is drawn and it is left open (not shaded in).

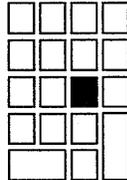
Using the SOFTWARE Doctype Documenting Terminal Keys and Keypads

The following examples show how to use the keypad section tags to create various keypad diagrams. The first example shows a complete keypad with one key shaded in.

```
<KEYPAD_SECTION>
<KEYPAD>(A Complete Keypad Diagram)
<KEYPAD_ROW>( \ \ \ )
<KEYPAD_ROW>( \ \ \ )
<KEYPAD_ROW>( \ \ \CLOSED\ )
<KEYPAD_ROW>( \ \ \NONE )
<KEYPAD_ENDROW>( \ \ )
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

A Complete Keypad Diagram



The following example shows a single line from the previous keypad diagram:

```
<KEYPAD_SECTION>
<KEYPAD>(A Single Keypad Line)
<KEYPAD_ROW>( \ \ \CLOSED\ )
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

A Single Keypad Line

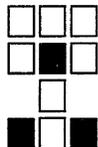


The following example shows the complete keypad with some keys eliminated from the diagram. You can use the NONE keyword argument to eliminate keys from the keypad diagram.

```
<KEYPAD_SECTION>
<KEYPAD>(An Irregular Keypad)
<KEYPAD_ROW>( \ \ \NONE)
<KEYPAD_ROW>( \CLOSED\ \NONE)
<KEYPAD_ROW>(NONE\ \NONE\NONE)
<KEYPAD_ROW>(CLOSED\ \CLOSED\NONE)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

An Irregular Keypad



Using the SOFTWARE Doctype Documenting Terminal Keys and Keypads

The following examples show how the DISPLAY keyword argument to the <KEYPAD> tag lets you specify text in some keys and shade other keys:

```
<KEYPAD_SECTION>
<KEYPAD>(The Editing Keypad with Key Labels\DISPLAY)
<KEYPAD_ROW>(PF1\PF2\PF3\PF4)
<KEYPAD_ROW>(7\8\9\-)
<KEYPAD_ROW>(4\5\6\,)
<KEYPAD_ROW>(1\2\3\ )
<KEYPAD_ENDROW>(0\.\ENTER)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

The Editing Keypad with Key Labels

PF1	PF2	PF3	PF4
7	8	9	-
4	5	6	,
1	2	3	ENTER
0	.		

```
<KEYPAD_SECTION>
<KEYPAD>(The Editing Keypad with Key Labels and Shaded Keys\DISPLAY)
<KEYPAD_ROW>(CLOSED\PF2\PF3\PF4)
<KEYPAD_ROW>(7\8\9\-)
<KEYPAD_ROW>(CLOSED\5\6\,)
<KEYPAD_ROW>(1\2\3\ )
<KEYPAD_ENDROW>(0\.\ENTER)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

The Editing Keypad with Key Labels and Shaded Keys

	PF2	PF3	PF4
7	8	9	-
	5	6	,
1	2	3	ENTER
0	.		

10.4 Documenting Code Fragments

There are three tags available in the SOFTWARE doctype that let you describe software code fragments. These tags let you create syntax statements, emphasize arguments, and create samples of screen displays.

- <SYNTAX> Formats text on the page exactly as entered to allow correct positioning of code or syntax statements.
- <ARGUMENT> Emphasizes arguments to functions, procedures, or tags in text using an altered text font (such as bold face or italic).
- <DISPLAY> Simulates a screen display.

Using the <DISPLAY> tag

The <DISPLAY> tag lets you create an example that simulates a screen display. You terminate this example with the <ENDDISPLAY> tag. The <DISPLAY> tag accepts one of two keyword arguments.

The WIDE keyword argument extends the display format into the left margin if the example is too wide for normal formatting. The KEEP keyword argument specifies that the entire display example is placed on the next page if it does not fit on the existing page. You use this argument to prevent unnecessary page breaks in display examples.

The following example shows how to use the <DISPLAY> tag to simulate a screen display. This example is coded using the WIDE argument due to the width of the example. Note that all the spacing in the display is retained as it was entered.

```
<DISPLAY>(WIDE)
VAX/VMS V4.4 on node XXXXXX 6-NOV-1986 17:43:18.03 Uptime 0 02:08:56
  Pid  Process Name  State Pri  I/O      CPU      Page flts Ph.Mem
20400080 NULL             COM    0    0  0 00:15:46.03  0  0
20400081 SWAPPER      HIB   16    0  0 00:00:19.74  0  0
20400085 ERREMT       HIB    7   157  0 00:00:01.32  67 104
20400086 CACHE_SERVER HIB   16    6  0 00:00:00.12  57  92
20400087 CLUSTER_SERVER HIB   10   14  0 00:00:00.61 109 257
20400088 OPCOM        LEF    7  126  0 00:00:01.06 332  77
20400089 JOB_CONTROL   HIB    8 2081  0 00:00:22.95 190 322
2040008A CONFIGURE   HIB    9   19  0 00:00:00.18  99 136
20400092 SYMBIONT_0001  HIB    6  312  0 00:00:06.34 2886 63
<ENDDISPLAY>
```

This example produces the following output:

```
VAX/VMS V4.4 on node XXXXXX 6-NOV-1986 17:43:18.03 Uptime 0 02:08:56
  Pid  Process Name  State Pri  I/O      CPU      Page flts Ph.Mem
20400080 NULL             COM    0    0  0 00:15:46.03  0  0
20400081 SWAPPER      HIB   16    0  0 00:00:19.74  0  0
20400085 ERREMT       HIB    7   157  0 00:00:01.32  67 104
20400086 CACHE_SERVER HIB   16    6  0 00:00:00.12  57  92
20400087 CLUSTER_SERVER HIB   10   14  0 00:00:00.61 109 257
20400088 OPCOM        LEF    7  126  0 00:00:01.06 332  77
20400089 JOB_CONTROL   HIB    8 2081  0 00:00:22.95 190 322
2040008A CONFIGURE   HIB    9   19  0 00:00:00.18  99 136
20400092 SYMBIONT_0001  HIB    6  312  0 00:00:06.34 2886 63
```

Using the SOFTWARE Doctype Documenting Code Fragments

Using the <SYNTAX> tag

The <SYNTAX> tag lets you distinguish the syntax of a programming language statement from regular text. The <SYNTAX> tag distinguishes the syntax statement by making the typeface of the statement different from the typeface used in the surrounding text. You terminate the <SYNTAX> tag with the <ENDSYNTAX> tag.

The <SYNTAX> tag accepts two optional arguments. The **WIDE** keyword argument allows more width for the syntax statement by letting the syntax statement extend into the left margin. The *alternate heading* argument lets you specify a heading for the syntax statement.

The following example shows how to use the <SYNTAX> tag to separate a syntax statement from surrounding text. Note that all the spacing in the statement is retained as it was entered.

```
<SYNTAX>
IF condition THEN
    statement list
[ELSE
    statement list];
<ENDSYNTAX>
```

This example produces the following output:

```
IF condition THEN
    statement list
[ELSE
    statement list];
```

Using the <ARGUMENT> tag

The <ARGUMENT> tag lets you label an argument by displaying that argument in a typeface that differs from that of surrounding text (for example, in some doctypes it may cause the argument to be displayed in a bold typeface). This tag accepts only the argument name as an argument.

The following example shows how to use the <ARGUMENT> tag to separate the argument from the surrounding text.

The CHR\$ function converts the <ARGUMENT>(char data) argument into a numeric value.

This example produces the following output:

The CHR\$ function converts the **char data** argument into a numeric value.

10.5 Documenting Software Messages

You can describe the messages issued by software programs by using the <MESSAGE_SECTION> tag and the tags it enables.

Tags enabled by the <MESSAGE_SECTION> tag are summarized in the following list:

Using the SOFTWARE Doctype Documenting Software Messages

<code><MESSAGE_TYPE></code>	Specifies the type of message being described using the <code><MSG></code> or <code><MSGS></code> tag.
<code><MSG></code>	Labels and formats up to two lines of message text in the message description section.
<code><MSGS></code>	Labels and formats up to nine lines of message text in the message description section.
<code><MSG_TEXT></code>	Specifies the explanatory text associated with the messages described with the <code><MSG></code> or the <code><MSGS></code> tag.

Messages generally come in one of three forms:

- A message text string only. An example of this type of string is System Resources Unavailable.
- A message text string preceded by a text string identification code. An example of this type of string is `%DIRECT-W-NOFILES, no files found`.
- A message text string preceded by a numeric string identification code. An example of this type of string is `%1288374 file lookup failed`.

The `<MESSAGE_SECTION>` tag begins the message description section. You terminate this tag by the `<ENDMESSAGE_SECTION>` tag. Select the tags you need to use in the message description section based upon the following criteria:

- The format your messages most closely follows
- The number of lines each message occupies on the terminal display

Using the `<MESSAGE_TYPE>` tag

Use the `<MESSAGE_TYPE>` tag to specify the type of message labeled by the `<MSG>` or `<MSGS>` tags. The `<MESSAGE_TYPE>` tag accepts one of three keyword arguments to determine the type of messages being described:

- **NOIDENT**
Specifies that the `<MSG>` and `<MSGS>` tags in the message section will not be given a numeric or text identification string as the first argument. **NOIDENT** is the default keyword. Specifying **NOIDENT** as the keyword argument has the same effect as not specifying the `<MESSAGE_TYPE>` tag at all.
- **TEXTIDENT**
Specifies that the `<MSG>` and `<MSGS>` tags in the message section will be given a text identification string as an argument.
You specify text identification strings and message text arguments as pairs. The **TEXTIDENT** keyword creates a comma (,) between text identification strings and message text arguments.
- **NUMIDENT**
Specifies that the `<MSG>` and `<MSGS>` tags in the message section will be given a numeric identification string as an argument.

Using the SOFTWARE Doctype Documenting Software Messages

You specify numeric identification strings and message text arguments as pairs. The NUMIDENT keyword assumes that the numeric identification string specified will not exceed a total length of 6 picas (approximately 10 characters).

The following is a sample use of the <MESSAGE_TYPE> tag. Note how the NUMIDENT keyword indicates that the message being described has a numeric identification code.

```
<MESSAGE_SECTION>  
<MESSAGE_TYPE> (NUMIDENT)  
<MSG>(%1288374 \file lookup failed.)  
.  
.  
.
```

Using the <MSG> tag

Use the <MSG> tag for messages that occupy only one or two lines when they are output. The arguments accepted by the <MSG> tag are based upon the type of message being described. The message type is determined by the keyword argument used with the <MESSAGE_TYPE> tags, as follows:

- NOIDENT keyword

The <MSG> tag requires a message text argument and accepts an optional second message text argument.

- TEXTIDENT keyword

The <MSG> tag requires a text message identification string and a message text argument. It also accepts a second line of message text as an optional third argument. The TEXTIDENT keyword creates a comma (,) between text identification strings and message text arguments. The optional second message text argument is stacked under the first message text.

- NUMIDENT keyword

The <MSG> tag requires a numeric message identification string and a message text argument. It also accepts a second line of message text as an optional third argument. The numeric message identification string must be no longer than 10 characters (6 picas). The optional second message text argument is stacked under the first message text.

The following is a sample use of the <MSG> tag. Note how the optional third argument is used for the additional message text.

```
<MESSAGE_SECTION>  
<MESSAGE_TYPE> (NUMIDENT)  
<MSG>(%1244374\file lookup failed\directory not found)  
.  
.  
.  
<ENDMESSAGE_SECTION>
```

Using the SOFTWARE Doctype

Documenting Software Messages

Using the <MSG> tag

Use the <MSG> tag for messages that occupy from one to nine lines when they are output. The arguments accepted by the <MSG> tag are based upon the type of message being described. The message type is determined by the keyword argument used with the <MESSAGE_TYPE> tag.

- NOIDENT keyword

The <MSG> tag requires one message text argument and accepts up to a total of nine message text arguments.

- TEXTIDENT keyword

The <MSG> tag requires text message identification strings as the first and optionally as the third, fifth, and seventh arguments. It also requires a message text argument as the second argument and optionally as the fourth, sixth, and eighth arguments. The TEXTIDENT keyword creates a comma (,) between text identification strings and message text arguments. The optional arguments are stacked under the required arguments.

- NUMIDENT keyword

The <MSG> tag requires numeric message identification strings as the first and optionally as the third, fifth, and seventh arguments. It also requires a message text argument as the second argument and optionally as the fourth, sixth, and eighth arguments. The text message identification strings must be no longer than 10 characters (approximately 6 picas). The optional arguments are stacked under the required arguments.

The following is a sample use of the <MSG> tag. Note how the numeric identification codes and the text strings associated with them are specified in pairs.

```
<MESSAGE_SECTION>
<MESSAGE_TYPE> (NUMIDENT)
<MSG> (%133455\read write error\%0000221\disk not available
\%6644544\file not found)
.
.
.
```

Using the <MSG_TEXT> tag

The <MSG_TEXT> tag labels the text that describes the messages listed by the <MSG> or <MSG> tags. By default, the <MSG_TEXT> tag has a default heading of Explanation:. If you want the heading for your text to be more descriptive, you can specify an alternate heading for this tag (for example, User Action). Note that any alternate heading you specify for the <MSG_TEXT> tag will have a colon (:) appended to the end of the heading when it is output.

The following example shows how to use the tags enabled by the <MESSAGE_SECTION> tag to create a message description section. Note how using the <MESSAGE_TYPE> tag with the TEXTIDENT keyword results in a comma (,) being placed after each message identification string.

Using the SOFTWARE Doctype Documenting Software Messages

```
<MESSAGE_SECTION>
<MESSAGE_TYPE>(TEXTIDENT)
<MSG>(BACKLINK\Incorrect directory back link)
<MSG_TEXT>(Facility) VERIFY, Verify Utility
<MSG_TEXT>(Severity) BACKLINK-F-BADLINK
<MSG_TEXT>
The Verify Utility could not process your command, please check the
syntax of your statement.
<MSG>(UAF-E-NAOFIL\Unable to open file SYSUAF.DAT\-RMS-E-FNF\file not found)
<MSG_TEXT>(User Action)
Check the syntax and reenter the command.
<MSG_TEXT>
This is some explanatory text for the previous message.
<ENDMESSAGE_SECTION>
```

This example produces the following output:

BACKLINK, Incorrect directory back link

Facility: VERIFY, Verify Utility

BACKLINK-F-BADLINK: The Verify Utility could not process your command, please check the syntax of your statement.

**UAF-E-NAOFIL, Unable to open file SYSUAF.DAT
-RMS-E-FNF, file not found**

User Action: Check the syntax and reenter the command.

BACKLINK-F-BADLINK: This is some explanatory text for the previous message.

10.6 Documenting Arguments, Parameters, and Qualifiers

There are four sets of tags available in the SOFTWARE doctype for describing arguments, parameters, and qualifiers in a list.

<ARGDEFLIST>	Creates a definition list of arguments
<PARAMDEFLIST>	Creates a definition list of parameters
<QUALDEFLIST>	Creates a definition list of qualifiers
<QUAL_LIST>	Creates a summary list of qualifiers

The tags used to create these SOFTWARE doctype definition lists and the form of these lists are very similar to those used by the global <DEFINITION_LIST> tag. The qualifier summary list differs from the definition lists in that it is designed only as a summary list of qualifiers and contains no provision for definition text.

These SOFTWARE doctype-specific tags are used to label lists of arguments, parameters, or qualifiers inside or outside the context of the reference templates.

Using the SOFTWARE Definition List Tags

The following list describes the software definition list tag sets. Note that tags that function in the same manner in each of the definition lists are described together.

Using the SOFTWARE Doctype

Documenting Arguments, Parameters, and Qualifiers

Software Definition List Tags

<ARGDEFLIST>
<PARAMDEFLIST>
<QUALDEFLIST>

Begins each type of list and enables the tags that follow. Each of these tags allows an optional heading to be specified as an argument. Use the NONE keyword argument if you want to indicate that the list contains no items.

If you use these tags inside a reference template, you may define default headings.

<ARGITEM>
<PARAMITEM>
<QUALITEM>

Labels the argument, parameter, or qualifier to be listed. Each of these tags requires one argument and accepts up to a total of seven arguments to list any related arguments, parameters, or qualifiers.

<ARGDEF>
<PARAMDEF>
<QUALDEF>

Labels the text string that describes the appropriate argument, parameter, or qualifier.

<ENDARGDEFLIST>
<ENDPARAMDEFLIST>
<ENDQUALDEFLIST>

Terminates each type of list and disables the contained tags.

The following examples show how to create a parameter definition list, an argument definition list, and a qualifier definition list.

The first example shows a parameter definition list in the Command template. This coding of the <PARAMDEFLIST> tag uses the NOHEAD keyword to suppress the output of a default heading (in this case, the heading parameters). If this definition list were coded outside the context of a reference template, no default heading would be defined and the NOHEAD argument would not be needed.

```
<P>The system maintains logical names and their
associated equivalence strings in two types of tables:
<PARAMDEFLIST> (NOHEAD)
<PARAMITEM> (process-private)
<PARAMDEF>These tables contain logical names that are available only
to your process.
<PARAMITEM> (shareable)
<PARAMDEF>These tables contain logical names that are available to other
processes on the system.
<ENDPARAMDEFLIST>
```

This example produces the following output:

The system maintains logical names and their associated equivalence strings in two types of tables:

Using the SOFTWARE Doctype

Documenting Arguments, Parameters, and Qualifiers

process-private

These tables contain logical names that are available only to your process.

shareable

These tables contain logical names that are available to other processes on the system.

The following example shows how to use the <ALIGN_AFTER> tag for additional formatting flexibility in an argument definition list outside a reference template.

```
<ARGDEFLIST>
<ARGITEM>(STATUS:arg\
<ALIGN_AFTER>(STATUS:)COMMAND\
<ALIGN_AFTER>(STATUS:)TASK)
<ARGDEF>Specifies whether exit status is to be returned
from the RUN command.
<ENDARGDEFLIST>
```

This example produces the following output:

```
STATUS:arg
COMMAND
TASK
```

Specifies whether exit status is to be returned from the RUN command.

The following example shows a qualifier definition list in the Command template. Note how the related qualifiers are stacked.

```
<COMMAND_SECTION>
<QUALDEFLIST>(Qualifiers)
<QUALITEM>(/HERE\ /THERE)
<QUALDEF>
The /HERE qualifier specifies the location of your workplace;
the /THERE qualifier specifies the location of your home;
/THERE is the default.
<QUALITEM>(/TIME\ /NOTIME)
<QUALDEF>
Specifies the amount of free time you have available.
If the /TIME qualifier is used, an amount must be given;
/NOTIME is the default.
<ENDQUALDEFLIST>
<ENDCOMMAND_SECTION>
```

This example produces the following output:

Qualifiers

```
/HERE
/THERE
```

The /HERE qualifier specifies the location of your workplace; the /THERE qualifier specifies the location of your home; /THERE is the default.

```
/TIME
/NOTIME
```

Specifies the amount of free time you have available. If the /TIME qualifier is used, an amount must be given; /NOTIME is the default.

Using the SOFTWARE Doctype Documenting Arguments, Parameters, and Qualifiers

Using the Software Qualifier Summary List Tags

VAX DOCUMENT provides several tags to create a summary list of command qualifiers. The list generated by these tags creates two default headings, and places one heading over each qualifier listed by the <QPAIR> tag. Although qualifier summary lists are most often used following the <FORMAT> tag in the Command template, they may be used either inside or outside the context of the reference templates.

Use the following tags to create a qualifier summary list:

- <QUAL_LIST>
Begins the list and enables the <QUAL_LIST_HEADS>, <QUAL_LIST_DEFAULT_HEADS>, <QPAIR>, and <ENDQUAL_LIST> tags. This tag places two default headings on the page. The heading Command Qualifiers is placed over the first list column and the heading Defaults is placed over the second list column. You may override these headings by using the <QUAL_LIST_HEADS> or <QUAL_LIST_DEFAULT_HEADS> tags.
Use the NONE keyword argument to indicate there are no qualifiers. If you use the NONE keyword, do not use the <ENDQUAL_LIST> tag. Otherwise, terminate this tag with the <ENDQUAL_LIST> tag.
- <QUAL_LIST_HEADS>
Creates alternate headings for a single qualifier summary list. This tag requires two arguments that specify the alternate headings. If either argument is null, the associated heading is not output.
- <QUAL_LIST_DEFAULT_HEADS>
Creates new default headings for all subsequent qualifier summary lists. This tag requires two arguments that specify the new default headings. If either argument is null, the associated heading is not output.
- <QPAIR>
Specifies the pair of qualifiers to be listed. This tag requires two qualifier arguments; the first argument is listed under the default heading Command Qualifiers, and the second argument is listed under the default heading Defaults.

The following example shows a qualifier summary list. Note how no description text is used. Note also how each list has a default heading.

```
<QUAL_LIST>  
<QPAIR> (/HERE\ /THERE)  
<QPAIR> (/TIME\ /NOTIME)  
<ENDQUAL_LIST>
```

This example produces the following output:

Command Qualifiers	Defaults
/HERE	/THERE
/TIME	/NOTIME

Using the SOFTWARE Doctype

Creating a Series of Interactive or Code Examples

10.7 Creating a Series of Interactive or Code Examples

Often, it is useful to explain concepts through the use of examples. You can create a series of numbered interactive and code examples by using the `<EXAMPLE_SEQUENCE>` tag.

Interactive and code examples in an example sequence have the same form and accept the same tags as examples created using the global `<INTERACTIVE>` and `<CODE_EXAMPLE>` tags.

You can use the following tags to construct a sequence of examples:

- `<EXAMPLE_SEQUENCE>` tag

Begins a sequence of numbered, informal examples. This tag accepts two optional arguments, used to alter the heading and suppress the numbering of examples.

The *heading info* argument alters the example sequence heading. This argument can be one of the following:

- An alternate heading for the example sequence.
- The `NOHEAD` keyword, which suppresses the output of a heading for the example sequence.
- The `EXAMPLE` keyword, which suppresses numbering of the examples in the sequence and outputs the heading `Example`; this keyword should be used when you have a single informal example rather than a series of examples.

The `NONUMBER` keyword argument suppresses the numbering of examples in an example sequence. When you use the keyword `EXAMPLE` as the *heading info* argument, the `NONUMBER` keyword argument is unnecessary.

The `<EXAMPLE_SEQUENCE>` tag enables the `<EXAMPLES_INTRO>`, `<EXC>`, `<EXI>`, and `<EXTTEXT>` tags. You terminate this tag by the `<ENDEXAMPLE_SEQUENCE>` tag.

- `<EXAMPLES_INTRO>`

Specifies introductory text for the example sequence.

- `<EXC>`

Specifies the beginning of a code example in an example sequence. You format this code example exactly the same as you would if using the global `<CODE_EXAMPLE>` tag. You terminate the code example by the `<EXTTEXT>` tag.

- `<EXI>`

Specifies the beginning of an interactive example in an example sequence. This tag uses the global `<S>` and `<U>` tags in the example in the same manner as the global `<INTERACTIVE>` tag. You terminate the interactive example by the `<EXTTEXT>` tag.

Using the SOFTWARE Doctype Creating a Series of Interactive or Code Examples

- `<EXTEXT>`

Specifies text that explains the previous example in the sequence. You must use this tag to terminate examples begun using the `<EXC>` and `<EXI>` tags.

The following example shows how to use the example sequence tags. Note the alternate heading specified as an argument to the `<EXAMPLE_SEQUENCE>` tag.

```
<EXAMPLE_SEQUENCE>(An Interactive Example and a Code Example)
<EXAMPLES_INTRO>
This is introductory text for the sample examples.
<exi><S>($ )<U>(SET WORK/NOTIME)
<EXTEXT>
This command sets the /NOTIME qualifier to the SET WORK command.
<EXC>This is a code example,                ined, exactly as entered.
        note how f                a
                o                t
                r                e
                matting is r

<EXTEXT>
This shows the flexibility available in a code example in an example
sequence.
<ENDEXAMPLE_SEQUENCE>
```

This example produces the following output:

An Interactive Example and a Code Example

This is introductory text for the sample examples.

```
1  $ SET WORK/NOTIME
```

This command sets the /NOTIME qualifier to the SET WORK command.

```
2  This is a code example,                ined, exactly as entered.
        note how f                a
                o                t
                r                e
                matting is r
```

This shows the flexibility available in a code example in an example sequence.

10.8 Using the Reference Templates

The SOFTWARE doctype contains four templates called **reference templates** that help you create software reference documentation. A reference template is a set of tags intended for some specialized documentation purpose, such as creating an outline, composing the front matter of a book, or documenting a set of software commands.

Every reference template has a beginning and an end. These template boundaries are set by a pair of tags:

- The **template-enabling tag** begins the template and sets up all the template-specific formats and tag definitions.
- The **template-ending tag** ends the template and disables the template-specific formats and tag definitions.

Using the SOFTWARE Doctype

Using the Reference Templates

If template-specific tags are encountered outside the template in which they are defined, VAX DOCUMENT treats these tags as undefined and issues a warning message.

There is one rule about using templates:

- 1 You must end one template before you begin another template; you may not nest one template inside another.

Reference templates are especially useful in creating and maintaining reference documentation. They make the coding of reference information easier in several ways:

- The structured nature of reference templates makes it easier to consistently code, format, and order reference information.
- Text coded into a reference template is more modular and structured than the text in a nontemplated SDML file, which makes it easier to modify and maintain reference information.
- Using a template for reference information allows the writer to fill in the blanks, and helps ensure that no essential information, such as restrictions or the lack of restrictions on a command, is omitted.
- Using a tag template for reference information lets you use default formats, headings, and tags. This helps guarantee that even when several writers work on different portions of the same book, all of these portions look alike.

Using the SOFTWARE Doctype Reference Templates

There are four reference templates available in the SOFTWARE doctype:

- **Command template**
Describes commands and their various components. Examples of command descriptions are the descriptions of the DCL commands used in the VMS operating system.
- **Routine template**
Describes software routines and their various components. Examples of routines descriptions are the descriptions of the VMS run-time library routines.
- **Statement template**
Describes programming language constructs (such as statements and functions) and their various components. An example of a statement description is the CASE statement available in the VAX Pascal programming language.
- **Tag template**
Describes VAX DOCUMENT tags and their components. Examples of tag descriptions can be found in all chapters in this manual.

Using the SOFTWARE Doctype Using the Reference Templates

All reference templates are similar in design because all are intended to be used to create similarly formatted reference material. This similarity means that certain tags are common to all four templates. However, because each template is customized for the presentation of a particular kind of information, several tags are available only in specific templates. For example, all four templates have a description section that describes the current command, routine, statement, or tag. However, only the Routine template has a section for describing in detail the values returned by a routine.

Like all tag templates, the software reference templates are begun by a template-enabling tag (for example, the `<COMMAND_SECTION>` tag), and terminated by a similarly named template-ending tag (for example, the `<ENDCOMMAND_SECTION>` tag). However, because the reference templates are designed to create reference-oriented documentation, they also must contain one or more **reference element** tags.

A reference element is a single element in a reference section, such as the description of a single command in a command reference section, or the description of a single routine in a routine reference section. The collection of these single elements into a group creates a reference section.

The default reference element tag names match the templates they are used in. The `<COMMAND>` tag occurs in the Command template, the `<ROUTINE>` tag occurs in the Routine template, and so on. When these reference element tags occur in a reference template, they signal the beginning of a new reference element description.

By default, each new reference element description begins on a new output page, with the name of the current reference element output at the top of the page.

Reference element tags are not terminated by matching ending tags, as are most template tags. Instead, a reference element description is terminated either by the next reference element tag, by the end of the reference template, or by the end of the SDML file.

Note that of all the tags used in the various reference templates, only the template-enabling tags and the reference element tags are required; all other tags used in the templates are optional.

Using the SOFTWARE Doctype

Using the Reference Templates

The following list shows the template-enabling tag and the reference element tag for each template.

Template Name	Template-Enabling Tag	Reference Element Tag
Command	<COMMAND_SECTION>	<COMMAND>
Routine	<ROUTINE_SECTION>	<ROUTINE>
Statement	<STATEMENT_SECTION>	{ <STATEMENT> ¹ } { <FUNCTION> }
Tag	<TAG_SECTION>	<SDML_TAG>

¹The use of braces indicates that the enclosed tag names are used interchangeably. VAX DOCUMENT provides two such names to improve the readability of your SDML file.

The Default Format of the Reference Templates

The software reference templates all have the same default output format. When a reference template is begun by specifying a template-enabling tag, the template has the following format:

- A new reference element description is begun and placed at the beginning of a new output page.

For example, each time a <COMMAND> tag is encountered in the command template, a new reference element description is begun and placed at the beginning of the next output page.

- The name of the current reference element is placed on each output page. The placement of this name depends on the doctype used. In most doctypes, this name is placed at the top of the page.

For example, if a command called SET TIME was being described and had been specified as <COMMAND>(SET TIME) in the SOFTWARE.REFERENCE doctype, then the single line running heading SET TIME would be placed at the top of the output page above the reference description.

- Each page is numbered using the last major page number prefix.

For example, if Chapter 2 was the immediately preceding chapter, then 2 would be the prefix associated with the page numbers in the reference section. Similarly, A would be the prefix if appendix A had preceded the reference section.

- Default headings are defined by the reference template for those tags that have such headings.

For example, the <RESTRICTIONS> tag has associated with it the default heading text Restrictions in the templates in which it is available.

Using the SOFTWARE Doctype Using the Reference Templates

Using the NONE Keyword Argument to Template Tags

Most of the reference template tags place a heading on the output page. Almost all of these tags accept the keyword NONE as an argument. This keyword indicates that the heading should be placed on the output page followed by the text None.

By using the NONE keyword, you ensure that a particular kind of information is explicitly declared as nonexistent or not applicable in a reference element description. Use the NONE keyword rather than typing the word None under the heading because inconsistencies in formatting and spelling of the word may otherwise be introduced.

When you use the NONE keyword as an argument to a template tag, the tag that terminates that template tag must be omitted. For example, if you specify the NONE argument to the <PARAMDEFLIST> tag, the <ENDPARAMDEFLIST> tag must be omitted.

Of all the tags that output a heading in the reference templates, only the <FORMAT>, <STATEMENT_FORMAT>, and <DESCRIPTION> tags do not accept a keyword argument of NONE, because placing the text None beneath the headings output by any of these tags would be inappropriate. If there are no format or description sections, these tags should be omitted.

Using Samples of the Reference Templates

VAX DOCUMENT provides the following sample reference template SDML files in the directory DOC\$TEMPLATES:

Command Template: DOC\$TEMPLATES:COMMAND.SDML
Routine Template: DOC\$TEMPLATES:ROUTINE.SDML
Statement Template: DOC\$TEMPLATES:STATEMENT.SDML
Tag Template: DOC\$TEMPLATES:TAG.SDML

You can also enter an editing session with LSE and expand the appropriate template tags into a reference template SDML file. See *VAX DOCUMENT User's Guide, Volume 1* for more information on using LSE with VAX DOCUMENT.

See the template samples at the end of this chapter for examples of the SDML code and output from each of the four reference templates.

10.9 Creating Your Own Reference Template Tags

VAX DOCUMENT provides the following tags for the creation of specialized reference template tags:

- <SET_TEMPLATE_LIST>
Creates a user-defined set of tags for creating your own headed list in the template.
- <SET_TEMPLATE_PARA>
Creates a user-defined set of tags for creating your own headed paragraph section in the template.

Using the SOFTWARE Doctype

Creating Your Own Reference Template Tags

- `<SET_TEMPLATE_TABLE>`
Creates a user-defined set of tags for creating your own headed table in the template.

Each of these tags defines a template tag and its terminator for use in a template section. You can specify the following arguments to these tags:

- Name by which a template tag is to be invoked
- Default heading for the tag when it is invoked
- Name of tags defined in the context of the template tag

For example, if you specify MYLIST as the first argument to the `<SET_TEMPLATE_LIST>` tag, you would define a list that is begun by the `<MYLIST>` tag and terminated by the `<ENDMYLIST>` tag. If you specify My List as the second argument to the `<SET_TEMPLATE_LIST>` tag, you would define a default heading for this list of My List. And if you specify MY_ITEM as the third argument to the `<SET_TEMPLATE_LIST>` tag, you would define the `<MY_ITEM>` tag as a tag to be used in the context of the `<MYLIST>` tag.

Each of the tags defined using these tags accepts an alternate heading argument and the NONE keyword argument. Note that these user-defined template tags have the same behavior as standard template tags. If the NONE keyword argument is specified, the terminating tag must not be used.

10.10 Creating Your Own Template Tables

The `<SET_TEMPLATE_TABLE>` tag lets you create your own set of tags for making a table with optional headings in a template. Tables created using this tag can have either two or three columns. This tag requires five arguments and accepts the optional *table column headings* argument. This tag has the following syntax:

Syntax

```
<SET_TEMPLATE_TABLE>(table tag name  
                    \default table heading  
                    \table row tag name  
                    \column count  
                    \column widths  
                    [table column headings])
```

The following list summarizes the arguments allowed by this tag in the order in which you must specify them:

table tag name

Specifies the user-defined name of the tag that begins the user-defined table.

default table heading

Specifies the default text heading to be output over the entire user-defined table. You override this heading by an alternate heading specified as an argument to the table tag name tag.

Using the SOFTWARE Doctype

Creating Your Own Template Tables

table row tag name

Specifies the name of the tag to be used to indicate individual table rows in the table being defined. For example, if the *table row tag name* argument is specified as SAMP_ROW, the individual table row tag will be <SAMP_ROW>.

The tag you create by this argument is similar to the global <TABLE_ROW> tag.

column count

Specifies the number of columns in the user-defined table. The accepted arguments are:

- 2 — Specifies that the table is to have two columns.
- 3 — Specifies that the table is to have three columns.

table column widths

Specifies the approximate widths of the table columns. The width of the last table column is determined by VAX DOCUMENT. So, if you specify a 2-column list, you must specify only one column-width argument, as shown in the following code example.

```
<SET_TEMPLATE_TABLE>(KEYVALS\Keyword Values\KEYVAL\2\10\Keyword\Value)
```

If you specify a 3-column list, you must specify two column-width arguments, as shown in the following code example.

```
<SET_TEMPLATE_TABLE>(KEYVAL_LIST\Keyword Ranges\KEYVAL\3\10\10  
\Keyword\High\Lower)
```

table column headings

Specifies optional default headings for each column in the user-defined table. If you specified a 2-column list, you may specify up to two heading arguments. If you specified a 3-column list, you may specify up to three heading arguments.

The following examples show a definition of the user-defined <RECORDTABLE> table tags. In this example, the <RECORDTABLE> tag is defined as a 2-column list with column headings, and with a default table heading of Best Songs.

The first use of these table tags in the following example sets the text supplied to the two <45RPM> tags in the table, and then terminates the table. The second use shows how the NONE keyword argument can be used.

```
<SET_TEMPLATE_TABLE>(RECORDTABLE\Best Songs\45RPM\2\12\Performer\Song Title)  
<RECORDTABLE>  
<45RPM>(Sinatra\Strangers in the Night)  
<45RPM>(Moody Blues\Nights in White Satin)  
<ENDRECORDTABLE>  
<RECORDTABLE>(NONE)
```

This example produces the following output:

Using the SOFTWARE Doctype

Creating Your Own Template Tables

best songs

Performer	Song Title
Sinatra	Strangers in the Night
Moody Blues	Nights in White Satin

best songs

None.

10.11 Modifying the Reference Templates

In most cases, you will not find it necessary to modify the reference templates. However, if you do want to modify these formats, you can do so by using one of the following tags:

- `<SET_TEMPLATE_HEADING>` tag
Creates new default headings for tags used in the reference templates.
- Template-enabling tags
Modifies the format of the entire template. This includes creating running headings, creating page number prefixes, and setting whether the reference template (and any reference elements in it) begins on a new page of output.
- `<SET_TEMPLATE_templatename>` tags
Modifies the format of the reference element tags used in each template. This includes specifying the current reference element name as a secondary running heading, specifying that the reference element heading has additional information stacked beneath it, and specifying whether the reference element tag should begin on a new page of output.

The modifications made by these tags only affect those tags that follow the modifying tag in the SDML file in the current reference template.

10.12 Modifying Default Headings in a Template

Each reference template enables several tags that place default headings on the output page. In most cases you will not want to change these default headings, because of the possibility of introducing typographical errors into the text of the heading. Changing the default headings can also cause your new headings to be incompatible with other headings in the current template or with headings in other reference sections.

However, if you do want to modify these headings, you can do so by using the `<SET_TEMPLATE_HEADING>` tag, which lets you specify a new default heading for a template tag in one of the reference templates. This tag has the following syntax:

Using the SOFTWARE Doctype

Modifying Default Headings in a Template

Syntax

<SET_TEMPLATE_HEADING> (*element keyword\default heading*)

The <SET_TEMPLATE_HEADING> tag accepts two arguments: the name of the template tag to receive the new default heading, and the text of the new default heading. The new default heading will then be used by all subsequent uses of the template tag named as an argument to the <SET_TEMPLATE_HEADING> tag in the current template. This heading overrides any previously defined default heading for that template tag in the current template.

A default heading created using the <SET_TEMPLATE_HEADING> tag will be in effect until that heading is reset in the current template by another <SET_TEMPLATE_HEADING> tag. Note also that the <SET_TEMPLATE_HEADING> tag has no effect on templates other than the one in which it is used.

The following example shows how you would create a new default heading. In this example, the default heading for the <RSDEFLIST> tag is set to be Conditions Signalled.

```
<ROUTINE_SECTION>
<SET_TEMPLATE_HEADING>(RSDEFLIST\Conditions Signalled)
<RSDEFLIST>
<RSITEM>(SS$_NORMAL\Service successfully completed.)
<RSITEM>(SS$_ACCVIO\Access violation.)
<ENDRSDEFLIST>
```

This example produces the following output:

CONDITIONS SIGNALLED

```
SS$_NORMAL           Service successfully completed.
SS$_ACCVIO           Access violation.
```

Table 10–7 summarizes the default headings assigned to the standard template tags.

Table 10–7 Default Headings of Reference Template Tags

Template Tag	Default Heading
Command Template Tags	
<FORMAT>	Format
<PARAMDEFLIST>	Parameters
<QUALDEFLIST>	Qualifiers
<DESCRIPTION>	Description
<RESTRICTIONS>	Restrictions
<PROMPTS>	Prompts

Using the SOFTWARE Doctype

Modifying Default Headings in a Template

Table 10–7 (Cont.) Default Headings of Reference Template Tags

Template Tag	Default Heading
Routine Template Tags	
<FORMAT>	Format
<ARGDEFLIST>	Arguments
<DESCRIPTION>	Description
<RSDEFLIST>	Return Values
Tag Template	
<FORMAT>	Format
<PARAMDEFLIST>	Arguments
<RELATED_TAGS>	Related Tags
<TERMINATING_TAG>	Required Terminator
<DESCRIPTION>	Description
Statement Template	
<STATEMENT_FORMAT>	Format
<DESCRIPTION>	Description
<FORMAT>	Format

10.13 Using the Template-Enabling Tags

You use template-enabling tags to begin a reference template, and may optionally use them to alter the default format of that template. The following table lists the template-enabling tags for each template.

Template Name	Template-enabling Tag
Command Template	<COMMAND_SECTION>
Routine Template	<ROUTINE_SECTION>
Statement Template	<STATEMENT_SECTION>
Tag Template	<TAG_SECTION>

All the template-enabling tags have the same syntax and perform the same functions. The syntax for the Command template form of the template-enabling tags is as follows:

```
<COMMAND_SECTION>[[running title][\number prefix][\NEWPAGE]]
.
.
.
<ENDCOMMAND_SECTION>
```

Arguments to the template-enabling tags are optional. However, if you use arguments, you must specify them in the order shown in the previous syntax description.

Using the SOFTWARE Doctype Using the Template-Enabling Tags

If you decide to modify a template using the template-enabling tag, these modifications will be in effect only in that particular template. The three arguments accepted by the template-enabling tags are given in the following list.

running title

Sets the second running heading for the page to be the text specified as the *running title* argument. This argument is valid only when double running heads are being used in the template.

number prefix

Sets the numbering prefix to be used to construct page numbers and formal figure, table, and example numbers. Such numbers might be DCL-12, STAT-5, or Table STAT-3.

NEWPAGE

Causes the initial text for the reference section to start on a new page. This argument also causes any template reference element (such as the <COMMAND> tag) to begin on a new page.

10.13.1 Template-Enabling Tag Behavior in the SOFTWARE.SPECIFICATION Doctype

When you use the SOFTWARE.SPECIFICATION doctype, an SDML file that contains both reference templates and chapters has the following output format:

- The doctype uses the current chapter title as a running footer on the right of the page bottom when the page number is odd and on the left of the page bottom when the page number is even. The chapter title overrides the running footer created by a template-enabling tag, such as <COMMAND>_SECTION> or <ROUTINE_SECTION>.
- The doctype uses the name of the current reference element, such as a <COMMAND> or <ROUTINE> tag, as a running title on the right of the page top when the page number is odd and on the left of the page top when the page number is even.

The doctype also uses the name of the current reference element as a running footer on the left of the page bottom when the page number is odd and on the right of the page bottom when the page number is even.

- A <RUNNING_TITLE> tag, if used, overrides the running title created by the reference element at the top of the page for the current reference template. A running title created using the <RUNNING_TITLE> tag exists only for the current reference template and ends with the template-ending tag, such as <ENDCOMMAND_SECTION> or <ENDROUTINE_SECTION>.

When you use the SOFTWARE.SPECIFICATION doctype, an SDML file that contains *reference templates but no chapters* has the following output format:

Using the SOFTWARE Doctype Using the Template-Enabling Tags

- The doctype uses the current template-enabling tag, such as `<COMMAND_SECTION>` or `<ROUTINE_SECTION>`, as a running footer on the right of the page bottom when the page number is odd and on the left of the page bottom when the page number is even.
- The doctype uses the name of the current reference element, such as `<COMMAND>` or `<ROUTINE>`, as a running title at the top of the page.
- A `<RUNNING_TITLE>` tag, if used, overrides the running title created by the reference element for the current reference template. A running title created using the `<RUNNING_TITLE>` tag exists only for the current reference template and ends with the template-ending tag, such as `<ENDCOMMAND_SECTION>` or `<ENDROUTINE_SECTION>`.

When you use a reference template, you typically use it in one of the following contexts:

- In a part begun using the global `<PART>` tag in a large document. Generally, the part follows one or more chapters that are numbered using the chapter number as the prefix for page numbers and for formal figure, table, and example numbers.
- In a chapter begun using the global `<CHAPTER>` tag in a book with chapter-oriented page numbers.
- In an appendix begun using the global `<APPENDIX>` tag that contains reference information intended to stand alone, that can be pulled out of the book in which it appears and placed in a binder with other system commands or routines.

In any of these situations you can use multiple reference templates, so long as each is terminated before the next reference template begins.

Note: VAX DOCUMENT does not allow you to nest reference templates inside one another and will signal an error if this occurs.

Using Reference Templates in Chapters

When a set of command or routine sections occur in a chapter, page and formal element numbering remain the same. That is, if the current chapter is 2, page numbering will continue to be 2-n.

You can let the chapter-heading text be carried at the top of each page (if that is the current doctype style), with the command name as the second-level heading. In a doctype in which the chapter heading text is carried at the bottom of the page, the running feet will remain unchanged.

By specifying the following tags and arguments:

```
<CHAPTER> (debug_chap)
.
.
.
<COMMAND_SECTION> (Debugger Commands)
<SET_TEMPLATE_COMMAND> (DBG_COMMAND\DOUBLERUNNINGHEADS)
<DBG_COMMAND> (ALLOCATE)
```

Using the SOFTWARE Doctype Using the Template-Enabling Tags

the page numbering will use the current number as the page number prefix, for example 3–n, with formal elements being numbered in the same way. In the command reference descriptions, the top-level title Debugger Commands will appear at the top of each page. The name of the current command will appear just below this heading.

When a document consists of chapters, each of which contains related reference elements, the chapter's running title will almost always be used as the running title. However, you can override this heading with the reference section enabling tags:

```
<CHAPTER>(aci_routines_chap)
<ROUTINE_SECTION>(ACI Routines)
```

When you specify a character-string prefix following a <CHAPTER> tag, the prefix will be used for:

- Page numbering
- Formal element (figure, table, and example) numbering

For example:

```
<CHAPTER>(FDL_routines_chap)
<COMMAND_SECTION>(\FDL)
```

In this example, the <COMMAND_SECTION> tag sets the folio using the string FDL. The header levels in this chapter will continue to be numbered using the chapter number (that is, 2.1, 2.1.1, and so on) but the pages and formal figures, tables, and examples will be numbered FDL–1, FDL–2, and so on.

Using Reference Templates in the Appendix Document Zone

If you want to document routines or commands or other reference elements in pull-out sections suitable for placing in a binder with other system commands or routines, and want these sections to be distinctly labeled, you can specify a character-string prefix to be used throughout your pullout reference section.

10.14 Using the <SET_TEMPLATE_templatename> Tags

The <SET_TEMPLATE_templatename> tag is not a VAX DOCUMENT tag. It is a generic name given to a set of four tags that have the same syntax, and that perform the same functions. These tags differ only in their names and in the templates in which they are available:

Command Template	<SET_TEMPLATE_COMMAND>
Routine Template	<SET_TEMPLATE_ROUTINE>
Statement Template	<SET_TEMPLATE_STATEMENT>
Tag Template	<SET_TEMPLATE_TAG>

These tags have the following syntax (illustrated by the <SET_TEMPLATE_COMMAND> tag in this case):

Using the SOFTWARE Doctype

Using the `<SET_TEMPLATE_templatename>` Tags

Syntax

`<SET_TEMPLATE_COMMAND>` (*tag name*[\i>attribute] [\i>attribute][\i>attribute])

These tags let you override the default formatting attributes for reference elements used in each of the reference templates. These tags require a first argument that is the name of the reference element tag in the template. You can specify the name of the default reference element tag (such as the `<ROUTINE>` tag in the Routine template), or you can specify a new tag name (such as `<MY_ROUTINE>`).

If you specify a new reference element tag name, you may not use the old tag name. It is no longer valid. The new tag name replaces the previous reference element tag. All subsequent uses of a reference element tag should use the last established tag name.

You can then specify one or more of the following keywords that are accepted by these tags (note that these keyword arguments must be separated by backslashes):

NONEWPAGE

Specifies that reference descriptions are not to start on new pages. By default, the tag defined by the first argument of the `<SET_TEMPLATE_templatename>` tag begins a command description on a new page.

DOUBLERUNNINGHEADS

Specifies that the command descriptions have two running titles at the top of every page. You set the top running title by the `<COMMAND_SECTION>` tag or by the heading of the most recent `<CHAPTER>` tag. By default, if a doctype design option does not call for running top titles, only the current command name is placed at the top of each page.

STACK

Specifies that when there are multiple arguments for the tag defined by the first argument to the `<SET_TEMPLATE_templatename>` tag, the arguments are stacked at the beginning of the page.

By default, when you specify multiple arguments, the second and third arguments are assumed to be optional descriptive information and output on the same line as the command name.

The following code example shows a sample use of the `<SET_TEMPLATE_COMMAND>` tag. In this example, the tag `<XYZ_COMMAND>` is defined and used. The keyword argument `NONEWPAGE` is specified to the `<COMMAND_SECTION>` tag so that the first description will not start on a new page. The keyword argument `STACK` is specified so that the two arguments `FIND_FIRST` and `FF` are stacked with the first argument on top.

```
<COMMAND_SECTION>(XYZ Commands\\NEWPAGE)
<SET_TEMPLATE_COMMAND>(XYZ_COMMAND\\NONEWPAGE\\STACK)
<XYZ_COMMAND>(FIND_FIRST\\FF)
```

This example produces the following output:

FIND_FIRST FF

10.15 Using the Command Template

Table 10–8 summarizes the tags available in the Command template, the default headings associated with them, and how they should be used. The table presents the tags in the same order as in the template in directory DOC\$TEMPLATES. Use these tags to create a command reference section. In most manuals, a command section is an encyclopedic reference section that describes each command the software system offers: its format, restrictions, prompts, parameters, functional description, command qualifiers, and positional qualifiers, plus examples of its use.

This section also contains a sample input and output file using the Command template. You may find these sample files useful in understanding how the Command template tags fit together.

- Section 10.15.1 contains the SDML file of a sample use of the Command template tags.
- Section 10.15.2 contains the output file created using the sample SDML file.

These samples describe the APPEND command. They are intended only as samples, and should not be used as a source of reference for this command.

Table 10–8 Command Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<COMMAND_SECTION>	None	Begins a Command section
<SET_TEMPLATE_COMMAND>	None	Alters the default format of a Command section
<SET_TEMPLATE_HEADING>	None	Alters the default headings in a reference section
<SET_TEMPLATE_PARA>	None	Creates user-defined tags for a specially formatted paragraph in a reference section
<SET_TEMPLATE_LIST>	None	Creates user-defined tags for a specially formatted list in a reference section
<SET_TEMPLATE_TABLE>	None	Creates user-defined tags for a specially formatted table in a reference section
<COMMAND>	None	Begins each new element to be referenced in the template; this is the default reference element tag in the Command reference template
<OVERVIEW>	None	Labels an overview of the reference element
<FORMAT>	Format	Labels the format of the reference element's syntax
<PARAMDEFLIST>	Parameters	Begins a definition list of the parameters or arguments associated with the reference element

Using the SOFTWARE Doctype

Using the Command Template

Table 10–8 (Cont.) Command Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<RESTRICTIONS>	Restrictions	Begins a list of zero or more restrictions on the reference element's use
<PROMPTS>	Prompts	Begins a list of the prompts associated with the reference element
<DESCRIPTION>	Description	Labels a reference element description section
<QUALDEFLIST>	Qualifiers	Begins a definition list of zero or more qualifiers associated with the reference element
<EXAMPLE_SEQUENCE>	Examples	Begins a sequence of one or more examples
<SUBCOMMAND_SECTION>	None	Begins a section of subcommands in a Command section
<SUBCOMMAND_SECTION_HEAD>	None	Specifies the heading for introductory text that precedes a subcommand section
<SET_TEMPLATE_SUBCOMMAND>	None	Alters the name of the <SUBCOMMAND> tag, and optionally lets you specify that each use of that tag should not begin output on a new page
<SUBCOMMAND>	None	Begins a new subcommand element to be described in a subcommand section

10.15.1 Sample SDML File of the Command Template

The following is an extended code example showing a VAX DOCUMENT SDML file that uses the Command template:

```
<COMMAND_SECTION>(Using the SOFTWARE Doctype\\NEWPAGE)
<SET_TEMPLATE_COMMAND>(DCL_COMMAND)

<DCL_COMMAND>(APPEND)
<OVERVIEW>
Adds the contents of one or more specified input files to the end of the
specified output file.
<ENDOVERVIEW>

<format>(Syntax)
<fCMD>(APPEND) <FPARMS>(input file spec[,<hellipsis>] output file spec)

<QUAL_LIST>(Command Qualifiers)
<QPAIR>(/BACKUP\ /CREATED)
<QPAIR>(/BEFORE[=time] \ /BEFORE=TODAY)
<ENDQUAL_LIST>

<QUAL_LIST>(Positional Qualifiers)
<QPAIR>(/ALLOCATION=n \See text.)
<QPAIR>(/[NO]CONTIGUOUS\None.)
<ENDQUAL_LIST>
<ENDFORMAT>

<RESTRICTIONS>(NONE)

<PROMPTS>
<PROMPT>(From:\input file spec[,<hellipsis>])
<PROMPT>(To:\output file spec)
<ENDPROMPTS>
<PARAMDEFLIST>
<PARAMITEM>(input file spec[,<hellipsis>])
<PARAMDEF>Specifies the names of one or more input files to be appended.
<P>
If you specify more than one input file, separate the specifications with
either commas (,) or plus signs (+).
Commas and plus signs are equivalent. All input files
are appended, in the order specified, to the end of the output file.
<P>
You can use wildcard characters in the file specification(s).

<PARAMITEM>(output file spec)
<PARAMDEF>
Specifies the name of the file to which the input files will be appended.
<P>
You must include at least one field in the output file specification. If you
do not specify a device and/or directory, the APPEND command uses the current
default device and directory. For other fields that you do not specify, the
APPEND command uses the corresponding field of the input file specification.
<P>
If you use the asterisk wildcard character in any field(s) of the
output file specification, the APPEND command uses the corresponding field of
the input file specification. If you are appending more than one
input file, APPEND uses the corresponding fields from the first input file.
<ENDPARAMDEFLIST>

<DESCRIPTION>
The APPEND command is similar in syntax and function to the COPY command.
Normally, the APPEND command adds the contents of one or more files
to the end of an existing file without incrementing the version number.
The /NEW_VERSION qualifier causes the APPEND command to create a new output
file if no file with that name exists.
<ENDDescription>
```

Using the SOFTWARE Doctype Using the Command Template

<QUALDEFLIST>(Command Qualifiers)
<QUALITEM>(/BACKUP)
<QUALDEF>
Selects files according to the dates of their most recent backup.
This qualifier is relevant only
when used with the /BEFORE or /SINCE qualifier. Use of the
/BACKUP qualifier is incompatible with /CREATED, /EXPIRED, and /MODIFIED.
The default is /CREATED.
<QUALITEM>(/BEFORE[=time])
<QUALDEF>
Selects only those files that are dated before the specified time.
<P>
You can specify
either an absolute time or a combination of absolute and delta times.
<comment>
See Section <reference>(time_sec) for complete information on
specifying time values.
<endcomment>
You can also use the
keywords TODAY, TOMORROW, and YESTERDAY. If no time is specified,
TODAY is assumed.
<ENDQUALDEFLIST>

<QUALDEFLIST>(Positional Qualifiers)
<QUALITEM>(/ALLOCATION=n\
<QUALDEF>
Forces the initial allocation of the output file to the number of 512-byte
blocks specified as n.
<P>
This qualifier is valid in conjunction with the /NEW_VERSION qualifier.
The allocation size is
applied only if a new file is actually created. If you create a new
file and you do not specify /ALLOCATION, the initial allocation of the
output file is determined by the size of the input file(s).
<QUALITEM>(/CONTIGUOUS\ /NOCONTIGUOUS)
<QUALDEF>
Indicates whether the output file is contiguous, that is, whether the file
must occupy consecutive physical disk blocks.
<P>
By default, the APPEND command creates an output file in the same format as
the corresponding input file. If an input file is contiguous, the APPEND
command attempts to create a contiguous output file, but does not
report an error if there is not enough space. If you append multiple
input files of different formats, the output file might or might not
be contiguous. Use the /CONTIGUOUS qualifier to ensure that the
output file is contiguous.
<ENDQUALDEFLIST>

<EXAMPLE_SEQUENCE>
<EXI><S>(\$) <U>(APPEND TEST.DAT NEWTEST.DAT)
<EXTTEXT>
The APPEND command appends the contents of the file TEST.DAT from the default
disk and directory to the file NEWTEST.DAT also located on the default disk
and directory.
<EXI>(WIDE)<S>(\$) <U>(APPEND /NEW_VERSION /LOG *.TXT T.SUM)
<S>(%APPEND-I-CREATED, D1\$:[MAL]T.SUM;1 created)
<S>(%APPEND-S-COPIED, D1\$:[MAL]A.TXT;2 copied to D1\$:[MAL]T.SUM;1 (1 block))
<S>(%APPEND-S-APPENDED, D1\$:[MAL]B.TXT;3 appended to D1\$:[MAL]T.SUM;1 (3 records))
<S>(%APPEND-S-APPENDED, D1\$:[MAL]G.TXT;7 appended to D1\$:[MAL]T.SUM;1 (51 records))
<S>(%APPEND-S-NEWFILES, 1 file created)
<EXTTEXT>
The APPEND command appends all files with file types of TXT to a file named
T.SUM. The /LOG qualifier requests a display of the specifications of each
input file appended. If the file T.SUM does not exist, the APPEND command
creates it, as the output shows. The number of blocks or records shown in the
output refers to the SDML file and not to the target file total.
<ENDEXAMPLE_SEQUENCE>
<ENDCOMMAND_SECTION>

10.15.2 Sample Output File of the Command Template

The following is the output from the extended code example in Section 10.15.1, produced using the SOFTWARE.REFERENCE doctype design. Note that your own output may vary, depending on the SOFTWARE design under which you process the SDML file.

Using the SOFTWARE Doctype

Command Template Output Example

APPEND

Adds the contents of one or more specified input files to the end of the specified output file.

SYNTAX

APPEND *input file spec[, . . .] output file spec*

Command Qualifiers	Defaults
--------------------	----------

/BACKUP

/CREATED

/BEFORE[=time]

/BEFORE=TODAY

Positional Qualifiers	Defaults
-----------------------	----------

*/ALLOCATION=*n**

See text.

/[NO]CONTIGUOUS

None.

restrictions

None.

prompts

From: *input file spec[, . . .]*

To: *output file spec*

PARAMETERS

input file spec[, . . .]

Specifies the names of one or more input files to be appended.

If you specify more than one input file, separate the specifications with either commas (,) or plus signs (+). Commas and plus signs are equivalent. All input files are appended, in the order specified, to the end of the output file.

You can use wildcard characters in the file specification(s).

output file spec

Specifies the name of the file to which the input files will be appended.

You must include at least one field in the output file specification. If you do not specify a device and/or directory, the APPEND command uses the current default device and directory. For other fields that you do not specify, the APPEND command uses the corresponding field of the input file specification.

If you use the asterisk wildcard character in any field(s) of the output file specification, the APPEND command uses the corresponding field of the input file specification. If you are appending more than one input file, APPEND uses the corresponding fields from the first input file.

Using the SOFTWARE Doctype Command Template Output Example

DESCRIPTION

The APPEND command is similar in syntax and function to the COPY command. Normally, the APPEND command adds the contents of one or more files to the end of an existing file without incrementing the version number. The /NEW_VERSION qualifier causes the APPEND command to create a new output file if no file with that name exists.

COMMAND QUALIFIERS

/BACKUP

Selects files according to the dates of their most recent backup. This qualifier is relevant only when used with the /BEFORE or /SINCE qualifier. Use of the /BACKUP qualifier is incompatible with /CREATED, /EXPIRED, and /MODIFIED. The default is /CREATED.

/BEFORE[=time]

Selects only those files that are dated before the specified time.

You can specify either an absolute time or a combination of absolute and delta times. You can also use the keywords TODAY, TOMORROW, and YESTERDAY. If no time is specified, TODAY is assumed.

POSITIONAL QUALIFIERS

/ALLOCATION=n

Forces the initial allocation of the output file to the number of 512-byte blocks specified as n.

This qualifier is valid in conjunction with the /NEW_VERSION qualifier. The allocation size is applied only if a new file is actually created. If you create a new file and you do not specify /ALLOCATION, the initial allocation of the output file is determined by the size of the input file(s).

/CONTIGUOUS

/NOCONTIGUOUS

Indicates whether the output file is contiguous, that is, whether the file must occupy consecutive physical disk blocks.

By default, the APPEND command creates an output file in the same format as the corresponding input file. If an input file is contiguous, the APPEND command attempts to create a contiguous output file, but does not report an error if there is not enough space. If you append multiple input files of different formats, the output file might or might not be contiguous. Use the /CONTIGUOUS qualifier to ensure that the output file is contiguous.

EXAMPLES

```
❏ $ APPEND TEST.DAT NEWTEST.DAT
```

The APPEND command appends the contents of the file TEST.DAT from the default disk and directory to the file NEWTEST.DAT also located on the default disk and directory.

Using the SOFTWARE Doctype Command Template Output Example

```
2 $ APPEND/NEW_VERSION/LOG *.TXT T.SUM
%APPEND-I-CREATED, D1$:[MAL]T.SUM;1 created
%APPEND-S-COPIED, D1$:[MAL]A.TXT;2 copied to D1$:[MAL]T.SUM;1 (1 block)
%APPEND-S-APPENDED, D1$:[MAL]B.TXT;3 appended to D1$:[MAL]T.SUM;1 (3 records)
%APPEND-S-APPENDED, D1$:[MAL]G.TXT;7 appended to D1$:[MAL]T.SUM;1 (51 records)
%APPEND-S-NEWFILES, 1 file created
```

The APPEND command appends all files with file types of TXT to a file named T.SUM. The /LOG qualifier requests a display of the specifications of each input file appended. If the file T.SUM does not exist, the APPEND command creates it, as the output shows. The number of blocks or records shown in the output refers to the SDML file and not to the target file total.

10.16 Using the Routine Template

Table 10–9 summarizes the tags available in the Routine template, the default headings associated with them, and how they should be used. The table presents the tags in the same order as in the template in directory DOC\$TEMPLATES. Use these tags to create a routine reference section. In most manuals, a routine reference section describes each routine the software offers: the routine’s format, returns, arguments, full routine description (perhaps with figures and tables), return values, and shows examples of its use.

This section also contains a sample input and output file using the Routine template. You may find these sample files useful in understanding how the Routine template tags fit together.

- Section 10.16.1 contains the SDML file of a sample use of the Routine template tags.
- Section 10.16.2 contains the output file created using the sample SDML file.

These samples describe the \$ENQ and MTH\$xSORT routines. They are intended only as samples, and should not be used as a source of reference for these routines.

Table 10–9 Routine Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<ROUTINE_SECTION>	None	Begins a Routine section
<SET_TEMPLATE_ROUTINE>	None	Alters the default format of a Routine section
<SET_TEMPLATE_HEADING>	None	Alters the default headings in a reference section
<SET_TEMPLATE_PARA>	None	Creates user-defined tags for a specially formatted paragraph in a reference section
<SET_TEMPLATE_LIST>	None	Creates user-defined tags for a specially formatted list in a reference section
<SET_TEMPLATE_TABLE>	None	Creates user-defined tags for a specially formatted table in a reference section
<ROUTINE>	None	Begins each new element to be referenced in the template; this is the default reference element tag in the Routine reference template
<OVERVIEW>	None	Labels an overview of the reference element
<FORMAT>	Format	Labels the format of the reference element’s syntax
<RETURNS>	Returns	Provides specific information about the attributes of the value returned by the routine
<RETTEXT>	None	Provides general information about the attributes of the value returned by the routine
<ARGDEFLIST>	Arguments	Begins a definition list of the arguments associated with the reference element
<DESCRIPTION>	Description	Labels a reference element description section

Using the SOFTWARE Doctype Using the Routine Template

Table 10–9 (Cont.) Routine Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<RSDEFLIST>	Return Values	Begins a definition list of zero or more routine return status codes and their meanings
<EXAMPLE_SEQUENCE>	Examples	Begins a sequence of one or more examples

10.16.1 Sample SDML File of the Routine Template

The following is an extended code example showing a VAX DOCUMENT SDML file that uses the Routine template:

```

<routine_section>(Using the SOFTWARE Doctype\\NEWPAGE)
<set_template_routine>(VMS_ROUTINE\doublerunningheads)
<VMS_ROUTINE>($ENQ\Enqueue Resource)

<OVERVIEW>This is the brief description section. It contains one or
two sentences describing what the routine does.
<ENDOVERRIDEVIEW>

<format>(Syntax)
<frtn>($ENQ) <fargs>([efn], lkmode, lksb, itmlst)
<ENDFORMAT>

<RETURNS>(cond_value\longword integer\read only\by value in R0)
<RETTEXT>At times, it may be necessary to include a sentence or two
here to further describe the nature of the information returned.
<ENDRETTEXT>

<ARGDEFLIST>
<ARGITEM>(efn\cond_value\longword integer\read only\by value)
<ARGDEF>Number of the event flag that is to be set when access is
granted to the specified resource. If not specified, the default is
event flag number 0.
<ARGITEM>(lkmode\cond_value\longword integer\read only\by descriptor\varying string
array descriptor)
<ARGDEF>Name of lock mode requested. May be one of the following:
<TABLE>
<TABLE_SETUP>(2\17)
<TABLE_HEADS>(Name of Lock Mode\Description)
<TABLE_ROW>(LCK$K_NLMODE\Null lock mode)
<TABLE_ROW>(LCK$K_CRMODE\Concurrent read mode)
<TABLE_ROW>(LCK$K_CWMODE\Concurrent write mode)
<ENDTABLE>
<ARGITEM>(lksb\cond_value\longword integer\write only\by value)
<ARGDEF>Address of the lock status block. The lock status block
receives the final completion status and lock I.D., and optionally
contains a lock value block.
<ARGITEM>(itmlst)
<ARGDEF>
Item list specifying the lock information that $GETLKI is to return. The
<variable>(itmlst) argument is the address of a list of item descriptors, each
of which describes an item of information. The list of item descriptors is
terminated by a longword of 0.
<line_art>
      31                15                0
      +-----+-----+-----+
      |      item code      |      buffer length      |
      +-----+-----+-----+
      |              buffer address              |
      +-----+-----+-----+
      |              return length address              |
      +-----+-----+-----+
<endline art>
<endargdeflist>

<DESCRIPTION>
This section contains the full, detailed description of the routine.
It may contain tables and figures. There is no fixed size for this
description section.
<ENDDDESCRIPTION>

```

Using the SOFTWARE Doctype Using the Routine Template

```
<RSDEFLIST>
<RSITEM>(SS$ NORMAL\Indicates successful completion)
<RSITEM>(SS$ ABORT\This description contains a full explanation of some of
the possible causes for the abortion)
<RSITEM>(SS$ DEADLOCK\This description contains a full explanation of
some possible causes for the deadlock situation.)
<ENDRSDEFLIST>

<EXAMPLE_SEQUENCE>
<examples_intro>
This section contains an example of the use of the routine.
This section can also contain figures and tables.
<ENDEXAMPLE_SEQUENCE>

<VMS_ROUTINE>(MTH$XSQRT)
<OVERVIEW>The square root procedure returns the square root of the
input parameter. The input parameter may have one of four data types:
F_Floating, D_Floating, G_Floating, and H_Floating.
<ENDOVERRIDEVIEW>

<format>(Syntax)
<ffunc>(MTH$SQRT\ (x))
<ffunc>(MTH$DSQRT\ (x))
<ffunc>(MTH$GSQRT\ (x))
<ffunc>(MTH$HSQRT\ (x))
<ENDFORMAT>

<RETURNS>(cond_value\F_Floating, D_Floating, or G_Floating
\write only\by value in R0)
<RETURNS>(headonly)
<RETTEXT>The square roots of F_Floating, D_Floating, and G_Floating
input parameters are returned by immediate value in R0 and R1. The
square root of an H_Floating parameter is returned by reference
in the output parameter <VARIABLE>(sqrt).
<ENDRETTEXT>

<ARGDEFLIST>(Argument)
<ARGITEM>(x\cond_value\F_Floating,D_Floating, G_Floating, or H_Floating
\read only\by reference)
<ARGDEF>The number for which the square root is desired.
<ARGITEM>(sqrt\cond_value\H_Floating\write only\by reference)
<ARGDEF>The square root of the H_Floating parameter.
<ENDARGDEFLIST>

<DESCRIPTION>
This is a description of the MTH$XSQRT function.
<ENDDescription>
<ENDROUTINE_SECTION>
```

10.16.2 Sample Output File of the Routine Template

The following is the output from the extended code example in Section 10.16.1, produced using the SOFTWARE.REFERENCE doctype design. Note that your own output may vary, depending on the SOFTWARE design under which you process the SDML file.

Using the SOFTWARE Doctype Routine Template Output Example

\$ENQ Enqueue Resource

This is the brief description section. It contains one or two sentences describing what the routine does.

SYNTAX **\$ENQ** [*efn*], *lkmode*, *lksb*, *itmlst*

RETURNS VMS Usage: **cond_value**
 type: **longword integer**
 access: **read only**
 mechanism: **by value in R0**

At times, it may be necessary to include a sentence or two here to further describe the nature of the information returned.

ARGUMENTS *efn*

VMS Usage: **cond_value**
type: **longword integer**
access: **read only**
mechanism: **by value**

Number of the event flag that is to be set when access is granted to the specified resource. If not specified, the default is event flag number 0.

lkmode

VMS Usage: **cond_value**
type: **longword integer**
access: **read only**
mechanism: **by descriptor—varying string array descriptor**
Name of lock mode requested. May be one of the following:

Name of Lock Mode	Description
LCK\$K_NLMODE	Null lock mode
LCK\$K_CRMODE	Concurrent read mode
LCK\$K_CWMODE	Concurrent write mode

lksb

VMS Usage: **cond_value**
type: **longword integer**
access: **write only**
mechanism: **by value**

Address of the lock status block. The lock status block receives the final completion status and lock I.D., and optionally contains a lock value block.

Using the SOFTWARE Doctype Routine Template Output Example

itmlst

Item list specifying the lock information that \$GETLKI is to return. The *itmlst* argument is the address of a list of item descriptors, each of which describes an item of information. The list of item descriptors is terminated by a longword of 0.

31	-----	15	-----	0
	item code		buffer length	
	buffer address			
	return length address			

DESCRIPTION

This section contains the full, detailed description of the routine. It may contain tables and figures. There is no fixed size for this description section.

RETURN VALUES

SS\$_NORMAL	Indicates successful completion.
SS\$_ABORT	This description contains a full explanation of some of the possible causes for the abortion.
SS\$_DEADLOCK	This description contains a full explanation of some possible causes for the deadlock situation.

EXAMPLES

This section contains an example of the use of the routine. This section can also contain figures and tables.

Using the SOFTWARE Doctype Routine Template Output Example

MTH\$xSQRT

The square root procedure returns the square root of the input parameter. The input parameter may have one of four data types: F_Floating, D_Floating, G_Floating, and H_Floating.

SYNTAX **MTH\$SQRT**(*x*)
 MTH\$DSQRT(*x*)
 MTH\$GSQRT(*x*)
 MTH\$HSQRT(*x*)

RETURNS VMS Usage: **cond_value**
 type: **F_Floating, D_Floating, or G_Floating**
 access: **write only**
 mechanism: **by value in R0**

RETURNS The square roots of F_Floating, D_Floating, and G_Floating input parameters are returned by immediate value in R0 and R1. The square root of an H_Floating parameter is returned by reference in the output parameter *sqrt*.

ARGUMENT **X**
 VMS Usage: **cond_value**
 type: **F_Floating, D_Floating, G_Floating, or H_Floating**
 access: **read only**
 mechanism: **by reference**
 The number for which the square root is desired.

sqrt

VMS Usage: **cond_value**
type: **H_Floating**
access: **write only**
mechanism: **by reference**
The square root of the H_Floating parameter.

DESCRIPTION This is a description of the MTH\$xSQRT function.

10.17 Using the Statement Template

Table 10–10 summarizes the tags available in the Statement template, the default headings associated with them, and how they should be used. The table presents the tags in the same order as in the template in directory DOC\$TEMPLATES.

Use these tags to create a statement reference section. In most manuals, a statement reference section describes each statement the software offers: its purpose and detailed format, plus examples of its use.

This section also contains a sample input and output file using the Statement template. You may find these sample files useful in understanding how the Statement template tags fit together.

- Section 10.17.1 contains the SDML file of a sample use of the Statement template tags.
- Section 10.17.2 contains the output file created using the sample SDML file.

These samples describe the RECORD statement and the MID\$ function. They are intended only as samples, and should not be used as a source of reference for these statements and functions.

Table 10–10 Statement Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<STATEMENT_SECTION>	None	Begins a Command section
<SET_TEMPLATE_STATEMENT>	None	Alters the default format of a Statement section
<SET_TEMPLATE_HEADING>	None	Alters the default headings in a reference section
<SET_TEMPLATE_PARA>	None	Creates user-defined tags for the creation of a specially formatted paragraph in a reference section
<SET_TEMPLATE_LIST>	None	Creates user-defined tags for a specially formatted list in a reference section
<SET_TEMPLATE_TABLE>	None	Creates user-defined tags for a specially formatted table in a reference section
{ <STATEMENT> ¹ }	None	Begins each new element to be referenced in the template; this is the default reference element tag in the Statement reference template
{ <FUNCTION> }		
<OVERVIEW>	None	Labels an overview of the reference element
<STATEMENT_FORMAT>	Format	Labels the format of the reference element's syntax
<FORMAT>	Format	Labels the format of the reference element's syntax
<DESCRIPTION>	Description	Labels a reference element description section
<EXAMPLE_SEQUENCE>	Examples	Begins a sequence of one or more examples

¹The use of braces indicates that the enclosed tag names are used interchangeably. VAX DOCUMENT provides two such names to improve the readability of your SDML file.

Using the SOFTWARE Doctype

Using the Statement Template

10.17.1 Sample SDML File of the Statement Template

The following is an extended code example showing a VAX DOCUMENT SDML file that uses the Statement template:

```
<STATEMENT_SECTION>(Using the SOFTWARE Doctype\\NEWPAGE)
<STATEMENT>(RECORD)

<OVERVIEW>
The RECORD statement lets you name and define data structures in a
BASIC program and provides the BASIC interface to the VAX Common Data
Dictionary (CDD). You can use the defined RECORD name anywhere a
BASIC data-type keyword is valid.
<ENDOVERRIDE>

<STATEMENT_FORMAT>
<FCMD>(RECORD)
<FPARMS>(rec nam)
<STATEMENT_LINE>(rec component)
<ELLIPSIS>
<FCMD>(END RECORD)
<FPARMS>([ rec nam ])
<construct_list>(rec component:)
<construct>(rec component:<list>(stacked\braces)
    <le>data type rec item [ , [ data type ] rec item ]
    <le>group clause
    <le>variant clause<endlist>
<construct>(rec item:<list>(stacked\braces)
    <le>unsubs vbl [ = int const ]
    <le>array ( int const,...) [ = int const ]
    <le><keyword>(FILL) [ ( int const ) ] [ = int const ]<endlist>
<construct>(group clause:)
    <keyword>(GROUP) group nam [ ( int const,... ) ]
    <statement_line>(rec component\indent)
    <ellipsis>
    <statement_line>(<keyword>(END GROUP) [ group nam ])
<construct>(variant clause:)
    <keyword>(VARIANT)
    <statement_line>(case clause\indent)
    <ellipsis>
    <statement_line>(<keyword>(END VARIANT))
<construct>(case clause:)
    <keyword>(CASE)
    <statement_line>([ rec component ]\indent)
<endconstruct_list>

<ENDSTATEMENT_FORMAT>

<function>(MID$)

<OVERVIEW>
The MID$ function extracts a specified substring from the middle of a
string, leaving the main string unchanged.
<ENDOVERRIDE>

<STATEMENT_FORMAT>
<fcmd>()<fparms>(str vbl =<list>(stacked\braces)
    <le>MID
    <le>MID$ <endlist> <keyword>((str exp, int expl, int exp2)))
<ENDSTATEMENT_FORMAT>

<ENDSTATEMENT_SECTION>
```

10.17.2 Sample Output File of the Statement Template

The following is the output from the extended code example in Section 10.17.1, produced using the `SOFTWARE.REFERENCE` doctype design. Note that your own output may vary, depending on the `SOFTWARE` design under which you process the SDML file.

Using the SOFTWARE Doctype Statement Template Output Example

RECORD

The RECORD statement lets you name and define data structures in a BASIC program and provides the BASIC interface to the VAX Common Data Dictionary (CDD). You can use the defined RECORD name anywhere a BASIC data-type keyword is valid.

Format

RECORD *rec nam*
rec component

·
·
·

END RECORD [*rec nam*]

rec component: { *data type rec item* [, [*data type*] *rec item*] }
group clause
variant clause

rec item: { *unsubs vbl* [= *int const*] }
array (*int const*, . . .) [= *int const*]
FILL [(*int const*)] [= *int const*]

group clause: **GROUP** *group nam* [(*int const*, . . .)]
rec component

·
·
·

END GROUP [*group nam*]

variant clause: **VARIANT**
case clause

·
·

END VARIANT

case clause: **CASE**
[*rec component*]

MID\$

The MID\$ function extracts a specified substring from the middle of a string, leaving the main string unchanged.

Format

$$str\ vbl = \left\{ \begin{array}{l} MID \\ MID\$ \end{array} \right\} (\text{str exp}, \text{int exp1}, \text{int exp2})$$

Using the SOFTWARE Doctype

Using the Tag Template

10.18 Using the Tag Template

Table 10–11 summarizes the tags available in the Tag template, the default headings associated with them, and how they should be used. The table presents the tags in the same order as in the template in directory DOC\$TEMPLATES.

Use these tags to create a tag reference section, such as the one at the end of this chapter. In most manuals, a tag reference section describes each tag the software offers: each tag's format, syntax, arguments, related tags, restrictions, required terminators, and functional description, plus examples of its use.

This section also contains a sample input and output file using the Tag template. You may find these sample files useful in understanding how the Tag template tags fit together.

- Section 10.18.1 contains the SDML file of a sample use of the Tag template tags.
- Section 10.18.2 contains the output file created using the sample SDML file.

This sample describes the <SYNTAX> tag. It is intended only as a sample, and should not be used as a source of reference for this tag.

Table 10–11 Tag Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<TAG_SECTION>	None	Begins a Tag section
<SET_TEMPLATE_TAG>	None	Alters the default format of a Tag section
<SET_TEMPLATE_HEADING>	None	Alters the default headings in a reference section
<SET_TEMPLATE_PARA>	None	Creates user-defined tags for a specially formatted paragraph in a reference section
<SET_TEMPLATE_LIST>	None	Creates user-defined tags for a specially formatted list in a reference section
<SET_TEMPLATE_TABLE>	None	Creates user-defined tags for a specially formatted table in a reference section
<SDML_TAG>	None	Begins each new element to be referenced in the template; this is the default reference element tag in the Tag reference template
<OVERVIEW>	None	Labels an overview of the reference element
<FORMAT>	Format	Labels the format of the reference element's syntax
<PARAMDEFLIST>	Arguments	Begins a definition list of the arguments associated with the reference element
<RELATED_TAGS>	Related Tags	Begins a list of zero or more tags related to the tag being described

Using the SOFTWARE Doctype Using the Tag Template

Table 10–11 (Cont.) Tag Template Tags as Available from DOC\$TEMPLATES

Tag Name	Default Heading	Template Usage
<TERMINATING_TAG>	Required Terminator	Labels the tag that terminates the tag being described
<RESTRICTIONS>	Restrictions	Begins a list of zero or more restrictions on the reference element's use
<DESCRIPTION>	Description	Labels a reference element description section
<EXAMPLE_SEQUENCE>	Examples	Begins a sequence of one or more examples

Using the SOFTWARE Doctype

Using the Tag Template

10.18.1 Sample SDML File of the Tag Template

The following is an extended code example showing a VAX DOCUMENT SDML file that uses the Tag template:

```
<TAG_SECTION>(Using the SOFTWARE Doctype\\NEWPAGE)
<SDML_TAG>(SYNTAX)

<OVERVIEW>
Lets you use special characters to describe language
syntaxes.
<ENDOVERRIDEVIEW>

<format>(Syntax)
<FTAG>(SYNTAX\\<LIST>(STACKED\\braces)
      <LE>heading text [<ARG_SEP>WIDE]
      <LE>WIDE<ENDLIST>\\OPTIONAL)
<ENDFORMAT>

<PARAMDEFLIST>
  <PARAMITEM>(heading text)
  <PARAMDEF>Specifies a heading. The doctype controls the font used to
display the heading. By default, this tag has no heading. You may want to create
a heading using the <TAG>(SYNTAX_DEFAULT_HEAD) tag.

    <PARAMITEM>(WIDE)
    <PARAMDEF>Specifies that the syntax statement may exceed the normal right
margin of the text. If you are using doctype designs that indent
the text body, a wide syntax example will extend into the left margin.
<ENDPARAMDEFLIST>

<RELATED_TAGS>
<RELATED_TAG>(display)
<RELATED_TAG>(syntax_default_head)
<RELATED_ITEM>The global <TAG>(CODE_EXAMPLE) tag
<RELATED_ITEM>The global <TAG>(FORMAT) tag
<ENDRELATED_TAGS>

<RESTRICTIONS>
You cannot use tab characters,
index tags (such as the <TAG>(x) and <TAG>(Y) tags),
or text element tags (such as <TAG>(p), <TAG>(list), or <TAG>(note))
in this type of example.
<ENDRESTRICTIONS>

<TERMINATING_TAG>(ENDSYNTAX)

<DESCRIPTION>
The <TAG>(SYNTAX) tag lets you accurately describe language
syntax. Languages can include programming languages, command
languages, application defined languages, and so forth. This tag also
separates the syntax example from the remaining text, retains blank
spaces and open lines, and labels the example (if you specified one)
using a doctype-specific font different from the current text font.
<ENDDescription>

<EXAMPLE_SEQUENCE>(EXAMPLE)
<EXC><LITERAL><P>The COPY command has the following syntax:
<SYNTAX>
      COPY input_file output_file
<ENDSYNTAX>

<ENDLITERAL>
<EXTTEXT>
This example produces the following output:
<P>
The COPY command has the following syntax:
<SYNTAX>
      COPY input_file output_file
<ENDSYNTAX>
<ENDEXAMPLE_SEQUENCE>
```

10.18.2 Sample Output File of the Tag Template

The following is the output from the extended code example in Section 10.18.1, produced using the SOFTWARE.REFERENCE doctype design. Note that your own output may vary, depending on the SOFTWARE design under which you process the SDML file.

Using the SOFTWARE Doctype Tag Template Output Example

<SYNTAX>

Lets you use special characters to describe language syntaxes.

SYNTAX

<SYNTAX>[({ *heading text* [\ *WIDE*] })]
WIDE

ARGUMENTS

heading text

Specifies a heading. The doctype controls the font used to display the heading. By default, this tag has no heading. You may want to create a heading using the <SYNTAX_DEFAULT_HEAD> tag.

WIDE

Specifies that the syntax statement may exceed the normal right margin of the text. If you are using doctype designs that indent the text body, a wide syntax example will extend into the left margin.

related tags

- <DISPLAY>
- <SYNTAX_DEFAULT_HEAD>
- The global <CODE_EXAMPLE> tag
- The global <FORMAT> tag

restrictions

You cannot use tab characters, index tags (such as the <X> and <Y> tags, or text element tags (such as <P>, <LIST>, or <NOTE>) in this type of example.

required terminator

<ENDSYNTAX>

DESCRIPTION

The <SYNTAX> tag lets you accurately describe language syntax. Languages can include programming languages, command languages, application defined languages, and so forth. This tag also separates the syntax example from the remaining text, retains blank spaces and open lines, and labels the example (if you specified one) using a doctype-specific font different from the current text font.

Using the SOFTWARE Doctype Tag Template Output Example

EXAMPLE

```
<P>The COPY command has the following syntax:  
<SYNTAX>  
    COPY input_file output_file  
<ENDSYNTAX>
```

This example produces the following output:

The COPY command has the following syntax:

```
COPY input_file output_file
```

Using the SOFTWARE Doctype

The SOFTWARE Doctype Tags

10.19 The SOFTWARE Doctype Tags

This part of Chapter 10 provides reference information on the SOFTWARE doctype tags and templates.

<ARGDEF>

Begins the text that defines an item in an argument definition list.

SYNTAX <ARGDEF>

ARGUMENTS *None.*

related tags

- <ARGDEFLIST>
- <ARGITEM>

restrictions Valid only in the context of the <ARGDEFLIST> tag.

DESCRIPTION The <ARGDEF> tag begins the text that defines an item in an argument definition list. This text describes the item listed by the previous <ARGITEM> tag. Terminate the text begun by the <ARGDEF> tag with the next <ARGITEM> or <ENDARGDEFLIST> tag.

EXAMPLE The following example shows an argument definition list used outside the routine template.

```
<ARGDEFLIST>(Arguments)
<ARGITEM>(data-1\data-2)
<ARGDEF>Specifies the arguments through which data is given to the routine.
<ENDARGDEFLIST>
```

This example produces the following output:

ARGUMENTS *data-1*
data-2
Specifies the arguments through which data is given to the routine.

SOFTWARE Doctype Tag Reference

<ARGDEFLIST>

<ARGDEFLIST>

Begins a definition list of arguments.

SYNTAX

<ARGDEFLIST>[({ *alternate heading*
NOHEAD
NONE })]

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading. The default heading provided by VAX DOCUMENT for the <ARGDEFLIST> tag can vary. See the DESCRIPTION section for more information on default argument definition list headings.

NOHEAD

This is an optional keyword argument. It suppresses the output of the default heading for the <ARGDEFLIST> tag.

NONE

This is an optional keyword argument. It causes the text None to be written to indicate that no arguments are available. Note that when you use the NONE keyword, do not use the <ENDARGDEFLIST> tag.

related tags

- <ARGDEF>
 - <ARGITEM>
 - <ARGTEXT>
 - <PARAMDEFLIST>
 - <QUALDEFLIST>
 - <SET_TEMPLATE_ARGITEM>
 - <SET_TEMPLATE_HEADING>
 - The global <DEFINITION_LIST> tag
-

required terminator

<ENDARGDEFLIST>

Required unless you specify the NONE keyword as an argument to the <ARGDEFLIST> tag.

DESCRIPTION

The <ARGDEFLIST> tag begins a definition list of arguments. This tag is similar in format and syntax to the global <DEFINITION_LIST> tag. See VAX DOCUMENT *Using Global Tags* for more information on the <DEFINITION_LIST> tag.

SOFTWARE Doctype Tag Reference

<ARGDEFLIST>

The <ARGDEFLIST> tag enables two tags to create an argument definition list. The <ARGITEM> tag labels the list item being defined, and the <ARGDEF> tag begins the definition of the list item.

When you use the <ARGDEFLIST> tag in the Routine template, the arguments accepted by the <ARGITEM> tag are changed, and the <ARGTEXT> tag is enabled. The change in the arguments accepted by the <ARGITEM> tag and the addition of the <ARGTEXT> tag give you a more structured environment in which to create Routine template argument definition lists. See the descriptions of these tags in this chapter for more information.

When you use the <ARGDEFLIST> tag in the templates, a default heading is provided. Modify this heading for that single argument definition list using the alternate heading argument. A heading specified in this way overrides any existing default heading.

Use the <SET_TEMPLATE_HEADING> tag to create your own default headings in a reference template. Using this tag modifies the default headings for all subsequent <ARGDEFLIST> tags used in that reference template. See the reference description of the <SET_TEMPLATE_HEADING> tag for more information on that tag.

When you use the <ARGDEFLIST> tag outside a template, you define no default heading. Create your own heading for a single argument definition list by specifying that heading as the *alternate heading* argument.

The following informal table lists the default headings for the <ARGDEFLIST> by their context:

Context	Default Heading
Command Template	Arguments
Routine Template	Arguments
Statement Template	No default heading
Tag Template	Arguments
Outside a Template	No default heading

EXAMPLES

The following examples show various uses of the <ARGDEFLIST> tag. The following example shows an argument definition list used outside the context of the Routine template. Note the syntax used for the <ARGITEM> tag.

```
❏ <ARGDEFLIST>(Arguments)
   <ARGITEM>(data-1\data-2)
   <ARGDEF>Specifies the arguments through which data is given to the routine.
   <ENDARGDEFLIST>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<ARGDEFLIST>

ARGUMENTS

data-1

data-2

Specifies the arguments through which data is given to the routine.

The following example shows two argument definition lists used in the Routine template. The first list is coded using the Routine template-specific <ARGITEM> tag syntax. Note the headings produced by the <ARGITEM> tag in the output of this example.

The second argument definition list illustrates a use of the <ARGTEXT> tag. Typically, the <ARGTEXT> tag is used instead of the <ARGITEM> or <ARGDEF> tags.

```
2 <ROUTINE_SECTION>
.
.
.
<ARGDEFLIST>
<ARGITEM>(x\floating_point\F_Floating, D_Floating, G_Floating, or H_Floating
\read only\by reference\may also be given by value)
<ARGDEF>The number for which the square root is desired.
<ENDARGDEFLIST>
<ARGDEFLIST>
<ARGTEXT>
The arguments to the SYS$NONE routine are identical to those used
by the SYS$NULL routine. See the description of the SYS$NULL routine for
more information on these arguments.
<ENDARGTEXT>
<ENDARGDEFLIST>
.
.
<ENDROUTINE_SECTION>
```

This example produces the following output:

ARGUMENTS

X

VMS Usage: **floating_point**

type: **F_Floating, D_Floating, G_Floating, or H_Floating**

access: **read only**

mechanism: **by reference—may also be given by value**

The number for which the square root is desired.

ARGUMENTS

The arguments to the SYS\$NONE routine are identical to those used by the SYS\$NULL routine. See the description of the SYS\$NULL routine for more information on these arguments.

<ARGITEM>

Labels one to seven routine argument items to be defined in an argument definition list outside the Routine template, or a single routine argument and its attributes in the Routine template.

SYNTAX <ARGITEM>(item-1 [\ item-2 . . . [\ item-7]])

<ARGITEM>(arg name[\ usage information
 \ data type \ access
 \ mechanism[\ mechanism info]])

ARGUMENTS

item-n

Specifies the item in the argument list to be defined. This tag accepts a minimum of one *item-n* argument and a maximum of seven. When you specify more than one *item-n* argument, each subsequent *item-n* argument after the initial argument formats under the first argument.

arg name

Specifies the descriptive name assigned to the argument for reference purposes.

usage information

This is an optional argument. It specifies a keyword indicating the category of data to which the argument's value belongs. These keywords are system dependent, and are specified by agreed-upon conventions.

data type

This is an optional argument. It specifies the data type of the argument; for example, longword, byte, G_floating, and so on.

access

This is an optional argument. It specifies the access applied to the argument; for example, read-only, write-only, and so on.

mechanism

This is an optional argument. It specifies the mechanism by which the argument is passed; for example, by descriptor, by reference, or by value.

mechanism info

This is an optional argument. It specifies additional information you may append to the *mechanism* argument output.

related tags

- <ARGDEF>
- <ARGDEFLIST>
- <ARGTEXT>

SOFTWARE Doctype Tag Reference

<ARGITEM>

restrictions

The first syntax listed in the format section is valid in an argument definition list outside the Routine template. The second syntax listed in the format section is valid only in an argument definition list inside the Routine template.

required terminator

<ARGDEF>

DESCRIPTION

The <ARGITEM> tag labels one to seven routine argument items to be defined in an argument definition list outside the Routine template, or a single routine argument and its attributes in the Routine template. This tag accepts one of two sets of arguments, depending on whether or not the <ARGITEM> tag is used in the Routine reference template.

Outside the Routine template, the <ARGITEM> tag accepts the *item-n* argument, which may be repeated up to seven times, as specified in the first syntax listed in the Format section. This form of the <ARGITEM> tag lets you label one to seven routine argument items inside an argument definition list.

In the Routine template, the <ARGITEM> tag uses the arguments listed in the second syntax listed in the Format section. This form of the <ARGITEM> tag lets you label a single routine argument and its attributes.

EXAMPLE

See the example in the <ARGDEFLIST> tag description.

<ARGTEXT>

Labels definition text in an argument definition list that replaces the information contained in a pair of <ARGITEM> and <ARGDEF> tags.

SYNTAX

<ARGTEXT>

related tags

- <ARGDEF>
- <ARGDEFLIST>
- <ARGITEM>

restrictions

Valid only in the context of the <ARGDEFLIST> tag in the Routine template.

required terminator

<ENDARGTEXT>

DESCRIPTION

The <ARGTEXT> tag labels definition text in an argument definition list that replaces the information contained in a pair of <ARGITEM> and <ARGDEF> tags. This tag is most useful when the arguments to be described are already listed elsewhere in the reference documentation.

EXAMPLE

The following example shows how to use the <ARGTEXT> tag to replace a pair of <ARGDEF> and <ARGITEM> tags. Note that this tag is restricted to argument definition lists used in the context of the Routine template.

```
<ROUTINE_SECTION>
.
.
.
<argdeflist>
<argtext>
The arguments to the SYS$NONE routine are identical to those used
by the SYS$NULL routine. See the description of the SYS$NULL routine for
more information on these arguments.
<endargtext>
<endargdeflist>
.
.
.
<endroutine_section>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<ARGTEXT>

ARGUMENTS

The arguments to the SYS\$NONE routine are identical to those used by the SYS\$NULL routine. See the description of the SYS\$NULL routine for more information on these arguments.

<ARGUMENT>

Emphasizes an argument name in text.

SYNTAX <ARGUMENT>(argument name)

ARGUMENTS *argument name*
Specifies an argument name to be emphasized.

related tags

- <KEYWORD>
- <VARIABLE>

DESCRIPTION The <ARGUMENT> tag emphasizes an argument name in text. This tag causes the *argument name* to appear in an altered font (such as boldface). However, the tag does not alter the case of its argument.

EXAMPLE The following example shows a sample use of the <ARGUMENT> tag.

<P>
The <ARGUMENT>(newadr) argument to the \$ADJSTK function provides the address of a longword to receive the updated value.

This example produces the following output:

The **newadr** argument to the \$ADJSTK function provides the address of a longword to receive the updated value.

SOFTWARE Doctype Tag Reference

<ARG_SEP>

<ARG_SEP>

Creates a separator character (\) between arguments in the FORMAT section of a tag description.

SYNTAX

<TAG_NAME>(*argument-1*<ARG_SEP>*argument-2*)

ARGUMENTS

argument-1

Whenever you specify an argument to a tag, use the <ARG_SEP> tag to create the argument separator character, the backslash (\).

argument-2

Whenever you specify an argument to a tag, use the <ARG_SEP> tag to create the argument separator character, the backslash (\).

related tags

- <FTAG>
-

restrictions

Valid only in the FORMAT section of a <FTAG> tag description in the Tag template.

DESCRIPTION

The <ARG_SEP> tag creates a separator character (\) between arguments in the FORMAT section of a tag description.

The <ARG_SEP> tag has no other function than to output the tag separator character (the backslash) in a Tag template format section.

EXAMPLE

The following example shows how to use the <ARG_SEP> tag to separate the arguments in a tag description that has one required and two optional arguments.

```
<format>
<ftag>(ROUTINE\name[<arg_sep>info1[<arg_sep>info2]])
<endformat>
```

This example produces the following output:

FORMAT

<ROUTINE>(*name*[\ *info1*[\ *info2*]])

<AUTHOR>

Places the name of an author and one or two additional lines of information about the author in the front matter portion of a document processed using the SOFTWARE.SPECIFICATION doctype.

SYNTAX

<AUTHOR>(*author name* [\ *author info-1*] [\ *author info-2*])

ARGUMENTS

author name

Specifies the name of the author. If you use the *author info-n* arguments, this line will be the top line.

author info-n

This is an optional argument. It specifies any additional information on the author. Information specified as *author info-1* is placed above information specified as *author info-2*.

related tags

- <BYLINE>
- <SIGNATURES>
- The global <FRONT_MATTER> tag

restrictions

Available only in the SOFTWARE.SPECIFICATION doctype following the global <FRONT_MATTER> tag.

DESCRIPTION

The <AUTHOR> tag places the name of an author and one or two additional lines of information about the author in the front matter portion of a document processed using the SOFTWARE.SPECIFICATION doctype. This tag accepts two optional arguments to provide additional information about the author.

If you want a signatory line for the author in the front matter, use the <BYLINE> and <SIGNATURES> tags. See the reference descriptions of those tags in this chapter for more information.

SOFTWARE Doctype Tag Reference

<AUTHOR>

EXAMPLE

The following example shows how to use the <AUTHOR> tag in the front matter of a document. Note how the optional second argument to the <AUTHOR> tag lists the position of the author.

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(The NYUC Simulator Reference Manual)
<ORDER_NUMBER>(AA-Z0000-TE)
<ABSTRACT>
This manual describes the NYUC Simulator.
This program simulates a conversation between three people
by analyzing the syntactic and semantic components of three
related statements, and then synthesizing statements and responses
based upon these original statements.
<ENDABSTRACT>
<REVISION_INFO>(This revision is personally signed.)
<AUTHOR>(Mr. Jones\Research Head, STG Inc.)
<SIGNATURES>
<BYLINE>(Nat Jones\Author)
<DATE>(July 11, 1985)
<PRINT_DATE>(June 1987)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

<BYLINE>

Places a name and other optional information below a ruled line in a signature list processed using the SOFTWARE.SPECIFICATION doctype.

SYNTAX <BYLINE>(name [\ *additional info*])

ARGUMENTS

name

This is an optional argument. It specifies the name of the signatory. This name is placed under the beginning of the signature line on the left side of the page.

additional info

Specifies any additional information about the signatory. This information is placed on the same line as the *name* argument with an em dash (—) between the two arguments.

related tags

- <AUTHOR>
- <SIGNATURES>
- The global <FRONT_MATTER> tag

restrictions

Valid only in the context of the <SIGNATURES> tag.

DESCRIPTION

The <BYLINE> tag places a name and other optional information below a ruled line in a signature list processed using the SOFTWARE.SPECIFICATION doctype. One use of this tag is to create an approval line in the front matter of a document and to place the name of the signatory beneath that line. You can optionally place additional information about the signer by using the *additional info* argument. Additional information formats to the right of the name of the signer, on the same line, separated by an em dash (—).

Use as many <BYLINE> tags as you want to create approval lines in the front matter of a document, as long as all these tags follow the <SIGNATURES> tag. Use the <SIGNATURES> tag to begin all the approval lines on a separate page of the front matter. See the reference description of the <SIGNATURES> tag in this section for more information on that tag.

SOFTWARE Doctype Tag Reference

<BYLINE>

EXAMPLE

The following example shows three occurrences of the <BYLINE> tag. The first two occurrences list the positions of the signers using the optional *additional info* argument; the third occurrence omits the optional argument. Note that all three tags follow the <SIGNATURES> tag.

```
<FRONT_MATTER>
<TITLE_PAGE>
<TITLE>(The NYUC Simulator Reference Manual)
<REVISION_INFO>(This revision is personally signed.)
<AUTHOR>(Mr. Jones\Research Head, STG Inc.)
<SIGNATURES>
<BYLINE>(Nat Jones\Author)
<BYLINE>(Cecil Mills\Co-author)
<BYLINE>(Matt Smith)
<DATE>(July 11, 1985)
<PRINT_DATE>(June 1987)
<ENDTITLE_PAGE>
<ENDFRONT_MATTER>
```

<COMMAND>

Begins a new command description.

SYNTAX <COMMAND>(command name[\ informational name] \ symbol name)

ARGUMENTS *command name*

Names the command described in the section.

informational name

This is an optional argument. It specifies an optional description of the command's function.

symbol name

Specifies the name of the symbol used in all references to the command.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <COMMAND_SECTION>
- <SET_TEMPLATE_ARGITEM>
- <SET_TEMPLATE_COMMAND>

restrictions

Valid only in the context of the Command template.

DESCRIPTION

The <COMMAND> tag to begins a new command description. This description is for a single command in the context of the <COMMAND_SECTION> tag. This tag has the following default format:

- Each <COMMAND> tag begins a new page of output.
- Each output page carries a single running title, which is the current command name.
- If the optional *informational name* argument is used with the <COMMAND> tag, this argument will be output after the *command name* argument and the two arguments will be separated by an em dash (—).

Use the <SET_TEMPLATE_COMMAND> tag to replace the <COMMAND> tag with a tag specific to your task (for example, <INTERNAL_COMMAND>), or to change the default attributes of the <COMMAND> tag. See the description of the <SET_TEMPLATE_COMMAND> tag in this chapter for more information.

SOFTWARE Doctype Tag Reference

<COMMAND>

EXAMPLES

The following example shows a command section begun using the <COMMAND_SECTION> tag. In this command section, the <COMMAND> tag begins the command description for the OPEN command.

1 <COMMAND_SECTION>
<COMMAND>(OPEN)
<OVERVIEW>
.
.
.

In the following example, the <COMMAND> tag has two arguments. The command name, CLOSE, appears at the beginning of the command description. The text specified in the second argument, Close a File, is printed on the same line as the command name, separated by an em dash (—).

2 <COMMAND_SECTION>
<COMMAND>(CLOSE\Close a File)

SOFTWARE Doctype Tag Reference

<COMMAND_SECTION>

- <RESTRICTIONS>
- <SET_TEMPLATE_ARGITEM>
- <SET_TEMPLATE_COMMAND>
- <SET_TEMPLATE_HEADING>
- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_PARA>
- <SET_TEMPLATE_TABLE>

restrictions

Valid only in the context of the Command template.

required terminator

<ENDCOMMAND_SECTION>

DESCRIPTION

The <COMMAND_SECTION> tag begins a new command description. Place a command section in a chapter or an appendix, or following a part page (that is, in a document section begun with the <PART_PAGE> tag). You code a command section in a chapter or an appendix in the same manner; command sections in parts are handled differently.

If your command section follows a part page, and you include text between the part page and the command section, specify the NEWPAGE keyword as the third argument to the <COMMAND_SECTION> tag. This causes the command section to begin on a new page. The following code fragment shows a command section that begins on a new page:

```
<COMMAND_SECTION> (\CD\NEWPAGE)
```

EXAMPLES

The following example shows how to begin a command section in a document part.

```
1 <PART>
  <PART_PAGE>
  <TITLE> (Part III\Command Dictionary)
  <ENDPART_PAGE> (RENUMBER)
  <COMMAND_SECTION> (Command Dictionary\CD)
  <SET_TEMPLATE_COMMAND> (DCL_COMMAND)

  <DCL_COMMAND> (GOTO)
  <OVERVIEW>
  Transfers control to a labeled statement in a command procedure.
  <ENDOVERVIEW>
  .
  .
  <ENDCOMMAND_SECTION>
```

SOFTWARE Doctype Tag Reference

<COMMAND_SECTION>

The tags in the previous example perform the following functions:

- The global <PART> tag begins the part.
- The global <PART_PAGE> tag creates a part page.
- The global <TITLE> tag is used in the context of the <PART_PAGE> tag to create a title on the part page.
- The RENUMBER argument to the global <ENDPART_PAGE> tag specifies that the pages should be renumbered beginning with the part page. This causes the first page of text following the part page to be numbered page 3 (page 1 is the unnumbered page the part page title is placed on, page 2 is the back of page 1, and page 3 is the first numbered page after the part page).
- The <COMMAND_SECTION> tag begins the command section and specifies the running title Command Dictionary as the running title for the command section. If the <SET_TEMPLATE_COMMAND> tag were used with the DOUBLERUNNINGHEADS argument, the title Command Dictionary would be used as the top running title.

The <COMMAND_SECTION> tag also specifies that the prefix CD should be used to construct numbers for pages and for formal figures, tables, and examples in the command section (for example, CD-11, CD-32, Table CD-1, Example CD-2, and so on).

- The <SET_TEMPLATE_COMMAND> tag specifies that all command descriptions in this command section will be identified using the <DCL_COMMAND> tag rather than the default <COMMAND> tag. The <DCL_COMMAND> tag will have the default attributes of the <COMMAND> tag.

The following example shows how to create a command section in which each command description (begun with a <COMMAND> tag) is in a separate SDML file, and all these descriptions are included into a primary command description file. For example, the file MYCOM.SDML contains the following SDML tags:

```
<INCLUDE> (CLOSE .SDML)
<INCLUDE> (OPEN .SDML)
<INCLUDE> (READ .SDML)
<INCLUDE> (WRITE .SDML)
```

Each of the included files contains one command reference description begun with a <COMMAND> tag. For these files to process correctly, you must precede them with the <COMMAND_SECTION> tag, which enables the <COMMAND> tag. These files can have the necessary tags processed before them by specifying the /INCLUDE qualifier on the command line to include a startup definition file.

SOFTWARE Doctype Tag Reference

<COMMAND_SECTION>

- 2 <COMMAND_SECTION>(Command Dictionary\CD)
<SET_TEMPLATE_COMMAND>(COMMAND\DOUBLERUNNINGHEADS)

If this startup file were named CM_DCT_ST_UP.SDML, it could be included using the DOCUMENT /INCLUDE qualifier, as in the following example:

```
$DOCUMENT mycom SOFT.REF LN03 /INCLUDE=CM_DCT_ST_UP.SDML
```

When each individual file in MYCOM.SDML is processed, the correct sequence of tags will be read in to begin the command section.

Process multiple files together by using the <INCLUDE> tag to include them into a single master file (such as MYCOM.SDML), or include them into a bookbuild profile.

Use the <ELEMENT> tags to include multiple files into a profile. For example, the bookbuild profile file CDPRO.SDML could contain the following tags:

```
<PROFILE>  
<ELEMENT>(CLOSE.SDML)  
<ELEMENT>(OPEN.SDML)  
<ELEMENT>(READ.SDML)  
<ELEMENT>(WRITE.SDML) <COMMENT>(contains <ENDCOMMAND_SECTION> tag)  
<ENDPROFILE>
```

Note that the PROFILE file includes the <ENDCOMMAND_SECTION> tag in the appropriate file, so that the template terminates and the book builds correctly.

<CONSTRUCT>

Specifies a variable construct and gives its expansion.

SYNTAX <CONSTRUCT>[(*construct name*)]

ARGUMENTS *construct name*

This is an optional argument. It specifies the name of a construct. If you omit this argument, text following the <CONSTRUCT> tag formats as if you had specified text. The text is set at the same margin as additional construct items.

related tags

- <CONSTRUCT_LIST>
- <STATEMENT_LINE>

restrictions

Valid only in the context of the <STATEMENT_FORMAT> tag in the Statement template.

DESCRIPTION

The <CONSTRUCT> tag specifies a variable construct and gives its expansion.

EXAMPLE

The following example shows statement format for a statement with a single construct list. Note the use of <LIST> tags to specify the output and the <KEYWORD> tag to control the representation of programming keywords in variable name text.

```
<statement_format>
<FCMD>(REMAP) <FPARMS>((map nam) remap item,<hellipsis>)
<construct_list>(remap item:)
<construct>(remap item:)<list>(stacked\braces)
  <le>num vbl nam
  <LE>str array nam ( [ int exp,<hellipsis>] ) [ = int exp ]
  <LE>[ data type ] <keyword>(FILL) [ ( int exp ) ] [ = int exp ]
  <LE><keyword>(FILL%) [ ( int exp ) ]
  <LE><keyword>(FILL$) [ ( int exp ) ] [ = int exp ]
  <endlist>
<endconstruct_list>
<endstatement_format>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<CONSTRUCT>

Format

REMAP (*map nam*) *remap item*, . . .

remap item: {
 num vbl nam
 str array nam ([*int exp*, . . .]) [= *int exp*]
 [*data type*] **FILL** [(*int exp*)] [= *int exp*]
 FILL% [(*int exp*)]
 FILL\$ [(*int exp*)] [= *int exp*]
}

<CONSTRUCT_LIST>

Begins a list of construct items and definitions that expand on variables specified in the context of the <STATEMENT_FORMAT> tag.

SYNTAX <CONSTRUCT_LIST>(construct name)

ARGUMENTS *construct name*
Specifies the text of the longest name referenced in the construct list; that is, a name specified as an argument to <CONSTRUCT>.

related tags

- <CONSTRUCT>
- <STATEMENT_LINE>

restrictions Valid only in the context of a <STATEMENT_FORMAT> tag section in the Statement template.

required terminator <ENDCONSTRUCT_LIST>

DESCRIPTION The <CONSTRUCT_LIST> tag begins a list of construct items and definitions that expand on variables specified in the context of the <STATEMENT_FORMAT> tag. A **construct list** is a set of semantic rules for a programming language. Each item in a construct list is a semantic entity that may expand to one or more sets of additional constructs or entities. Using the global <LIST> tag in a construct list in a statement section, you can present the semantic rules for language statements in a highly structured way.

EXAMPLES The following are two examples of various uses of the <CONSTRUCT_LIST>tag in the context of the <STATEMENT_FORMAT> tag. Output from these coding examples appear after the last input example.

SOFTWARE Doctype Tag Reference

<CONSTRUCT_LIST>

The following example shows a construct list with only one construct.

```

1 <statement_format>
  <fcmd>(<list>(stacked\braces)
    <le>COM
    <le>COMMON<endlist>) <fparms>([ ( com nam ) ] {[data type ] com item },<hellipsis>)
    <construct_list>(com item:)
    <construct>(com item:)
    <list>(stacked\braces)
    <le><VARIABLE>(num unsubs vbl nam)
    <le><VARIABLE>(num array nam ( int const,<hellipsis>))
    <le><VARIABLE>(str unsubs vbl nam = int const)
    <le><VARIABLE>(str array nam ( int const,<hellipsis> ) [ = int const ])
    <le><VARIABLE>(<keyword>(FILL) [ ( int const ) ][ = int const ])
    <le><VARIABLE>(<keyword>(FILL%) [ ( int const ) ])
    <le><VARIABLE>(<keyword>(FILL$) [ ( int const ) ][ = int const ])
    <endlist>
  <endconstruct_list>
<endstatement_format>

```

The following example illustrates a construct list with more than one construct. The argument to the <CONSTRUCT_LIST> tag sets the margins for the individual items identified by <CONSTRUCT> tags. Note that the text of the longest item, *formal param.*, is specified.

```

2 <FCMD>(<list>(stacked\braces)
  <le>END SUB
  <le>SUBEND<endlist>) <FPARMS>()
  <construct_list>(formal param:)
  <construct>(pass mech:)<list>(stacked\braces)
    <le><keyword>(BY DESC)
    <le><keyword>(BY REF)<endlist>
  <construct>(formal param:)
  [ data type ] <list>(stacked\braces)
    <le>unsubs vbl nam
    <le>array nam ( <list>(stacked\brackets)
      <le>int const
      <le><keyword>(,<hellipsis>)<endlist>
      <list>(stacked)
      <le><keyword>(,<hellipsis>)
      <le><keyword>(,<hellipsis>)<endlist><endlist>
    <endconstruct_list>
  <endstatement_format>

```

The SDML examples produce the following output:

Format

**{ COM
COMMON }** [(*com nam*)] { [*data type*] *com item* }, . . .

com item: $\left\{ \begin{array}{l} \textit{num unsubs vbl nam} \\ \textit{num array nam (int const, . . .)} \\ \textit{str unsubs vbl nam = int const} \\ \textit{str array nam (int const, . . .) [= int const]} \\ \textbf{FILL} [(int const)][= int const] \\ \textbf{FILL\%} [(int const)] \\ \textbf{FILL\$} [(int const)][= int const] \end{array} \right\}$

Format

**{ END SUB
SUBEND }**

pass mech: $\left\{ \begin{array}{l} \textbf{BY DESC} \\ \textbf{BY REF} \end{array} \right\}$

formal param: [*data type*] $\left\{ \begin{array}{l} \textit{unsubs vbl nam} \\ \textit{array nam ([int const] [, \dots])} \end{array} \right\}$

SOFTWARE Doctype Tag Reference

<CPOS>

<CPOS>

Marks the cursor position in a screen display.

SYNTAX

<CPOS> (*placement character*)

ARGUMENTS

placement character

Specifies the character occupying the position of the cursor. This character can be a blank space.

related tags

- <DISPLAY>
 - <KEY>
 - <KEY_SEQUENCE>
-

DESCRIPTION

The <CPOS> tag marks the cursor position in a screen display. It places a special character in the screen display.

EXAMPLE

The following example shows how to use the <CPOS> tag to indicate a cursor's position. In this case the cursor is placed on the letter X in the directory specification [TRXTFILES].

<P>

To correct the file specification, move the cursor to the incorrect letter as shown in the following example:

```
<DISPLAY>  
  $ COPY ABC.TXT, [TR<CPOS>(X) TFILES]  
<ENDDISPLAY>
```

This example produces the following output:

To correct the file specification, move the cursor to the incorrect letter as shown in the following example:

```
$ COPY ABC.TXT [TRXTFILES]
```

<DELETE_KEY>

Creates a special character resembling a DELETE key on a keyboard.

SYNTAX <DELETE_KEY>

ARGUMENTS *None.*

related tags

- <GRAPHIC>
- <KEY>

DESCRIPTION The <DELETE_KEY> tag creates a special character resembling a DELETE key on a keyboard.

EXAMPLE The following example shows how to use the <DELETE_KEY> tag to create the delete key symbol.

<P>
Press the DELETE key (<DELETE_KEY>) to delete the text.

This example produces the following output:

Press the DELETE key (<X>) to delete the text.

SOFTWARE Doctype Tag Reference

<DESCRIPTION>

<DESCRIPTION>

Begins a section of descriptive text providing detailed information on the current reference element.

SYNTAX

<DESCRIPTION>*[(alternate heading)]*

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading. The default heading is Description. For information on how to modify the default headings for all subsequent <DESCRIPTION> tags, refer to the description of the <SET_TEMPLATE_HEADING> tag in this section.

related tags

- <OVERVIEW>
 - <SET_TEMPLATE_HEADING>
 - Any reference element tag: <COMMAND>, <ROUTINE>, <STATEMENT>, <FUNCTION>, or <SDML_TAG>
-

restrictions

Valid only in the context of a reference template. Valid only in the context of a reference section tag: <COMMAND_SECTION>, <ROUTINE_SECTION>, <STATEMENT_SECTION>, or <TAG_SECTION>.

required terminator

<ENDDescription>

DESCRIPTION

The <DESCRIPTION> tag begins a section of descriptive text providing detailed information on the current reference element. It separates and labels detailed descriptive text concerning the current reference element (command, routine, and so on). This text can describe the following aspects of the reference element:

- Suggested use
- Special considerations
- Use with other reference elements

Do not use a <P> tag immediately after the <DESCRIPTION> tag. The <DESCRIPTION> tag generates the initial open line so that you need only begin typing the first paragraph of text.

EXAMPLE

The following example shows a sample use of the <OVERVIEW> and <DESCRIPTION> tags in the context of the Command template. The Command template <COMMAND> tag and the global <FORMAT> and <PARAMDEFLIST> tags are included to make a more complete example. Notice that the <COMMAND> tag forces a page break by default.

```
<COMMAND_SECTION>
<COMMAND>(SET TIME)
<OVERVIEW>
Resets the system time to the time specified as a parameter to this command.
<ENDOVERVIEW>
<format>(Syntax)
<FCMD>(SET TIME)
<FPARMS>(time specification)
<ENDFORMAT>
<PARAMDEFLIST>
<PARAMITEM>(time specification)
<PARAMDEF> Specifies the time to which the system should be set.
<ENDPARAMDEFLIST>
<DESCRIPTION>
The SET TIME command resets the system time to the time specified as a
parameter to this command. You generally use this command to reset the
system clock on your VMS system, for example to allow for daylight
savings time and other such events.
<ENDDescription>
<ENDCOMMAND_SECTION>
```

SOFTWARE Doctype Tag Reference

<DISPLAY>

<DISPLAY>

Simulates the appearance and position of characters on a terminal screen.

SYNTAX

<DISPLAY>(display text)

<DISPLAY>[(KEEP
WIDE[\ MAXIMUM])]

ARGUMENTS

display text

Specifies a display fragment you want to insert into your text. If you do not specify this argument, the <ENDDISPLAY> tag is required.

KEEP

This is an optional keyword argument. It specifies that the display example should be placed on the next page rather than breaking it between the current page and the next page. If the display example is longer than a single page, it breaks between the current page and the next page.

WIDE

This is an optional keyword argument. It specifies a wide display that may exceed the normal right margin of the text.

MAXIMUM

This is an optional keyword argument. It specifies that the example may require adjustment to fit in the bounds of the text page. May be used only with the WIDE keyword. Using this argument may result in the display text being output in a smaller type face.

related tags

- <SYNTAX>
 - The global <INTERACTIVE> tag
 - The global <VALID_BREAK> tag
-

restrictions

You cannot use tab characters, index tags (such as <X> and <Y>), text element tags (such as <P>, <LIST>, or <NOTE>), or the <MATH> tag in any example.

required terminator

<ENDDISPLAY> Required if you do not specify the display example as the *display text* argument.

DESCRIPTION

The <DISPLAY> tag simulates the appearance and position of characters on a terminal screen. This tag lets you create display examples either as fragments in text or as extended examples. Such display examples are distinguished typographically from the surrounding text upon output.

To place a display fragment in the text of your document, specify the *display text* argument as the only argument to the <DISPLAY> tag and do not specify the <ENDDISPLAY> tag. This format is shown first in the format section of the <DISPLAY> tag.

If you have a long display example that you want to appear distinctly separate from the text of your document, place the text for the example between the <DISPLAY> and <ENDDISPLAY> tags. This format is shown second in the format section of the <DISPLAY> tag.

Display examples specified in this way retain all spaces and line breaks as entered. This lets you position the text in the example so that it appears similar to a terminal screen display. When you specify display examples in this form, you use the KEEP, WIDE, and MAXIMUM keyword arguments to control the formatting of your example.

If you have an especially long display example, you may want to use the <VALID_BREAK> tag in your example to indicate where a page may break.

EXAMPLES

The following example shows a display example that uses the <DISPLAY> and <ENDDISPLAY> tags. Note the use of the KEEP keyword to ensure that the display example is not broken across the page.

1 <P>
The operating system indicates success with the following message:
<DISPLAY>(KEEP)
 Welcome to VAX/VMS Version 5.2
 Username:
<ENDDISPLAY>

This example produces the following output:

The operating system indicates success with the following message:

```
    Welcome to VAX/VMS Version 5.2  
    Username:
```

The following example shows the <DISPLAY> tag used to create a display fragment in text. Note that the <ENDDISPLAY> tag is omitted and that no keywords to the <DISPLAY> tag can be specified.

2 <P>The message <DISPLAY>(Welcome to VAX/VMS Version 5.2)
is issued by default when a user presses RETURN.

This example produces the following output:

The message *Welcome to VAX/VMS Version 5.2* is issued by default when a user presses RETURN.

SOFTWARE Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

<DOCUMENT_ATTRIBUTES>

Enables doctype-specific tags that override the default design format of the SOFTWARE doctype.

SYNTAX <DOCUMENT_ATTRIBUTES>

ARGUMENTS *None.*

required terminator <ENDDOCUMENT_ATTRIBUTES>

DESCRIPTION The <DOCUMENT_ATTRIBUTES> tag enables doctype-specific tags that override the default design format of the SOFTWARE doctype. This tag can be used in three doctypes:

- ARTICLE
- REPORT
- SOFTWARE

The <DOCUMENT_ATTRIBUTES> tag enables a group of tags in each of these doctypes that allow you to modify the default format of that doctype. These tags are recognized only in the context of the <DOCUMENT_ATTRIBUTES> tag. If other VAX DOCUMENT tags occur in this context, they are ignored just as if they had occurred in the context of a <COMMENT> tag.

Typically, use the <DOCUMENT_ATTRIBUTES> tag at the beginning of an input file (or in a file specified using the /INCLUDE qualifier in the DOCUMENT command line) to alter the default format of a doctype for the processing of that entire file.

Table 10-12 summarizes the formatting tags enabled by the <DOCUMENT_ATTRIBUTES> tag in the SOFTWARE doctypes.

SOFTWARE Doctype Tag Reference

<DOCUMENT_ATTRIBUTES>

Table 10–12 Doctype-specific Tags Enabled by the <DOCUMENT_ATTRIBUTES> Tag

Formatting Tags	Description
<SET_RUNNING_TITLES>(BY_HEADONE)	<p>The <SET_RUNNING_TITLES> tag specifies that the running title at the top of each page is composed of two lines: the first line is the title of the chapter, and the second line is the heading text of the current <HEAD1> tag. Note that the argument BY_HEADONE is required.</p> <p>By default, the chapter title is used as a single-line running title.</p>

EXAMPLE

The following example of a file processed with the SOFTWARE doctype shows how you specify a 2-line running title using the <SET_RUNNING_TITLES> tag. This title has the title of the current chapter as the first line, and the heading text of the current <HEAD1> tag as the second line.

```
<DOCUMENT_ATTRIBUTES>
<SET_RUNNING_TITLES>(BY_HEADONE)
<ENDDOCUMENT_ATTRIBUTES>
```

SOFTWARE Doctype Tag Reference

<EXAMPLE_SEQUENCE>

<EXAMPLE_SEQUENCE>

Begins a numbered sequence of informal examples.

SYNTAX

<EXAMPLE_SEQUENCE>[({ *EXAMPLE*
 heading text
 NOHEAD
 \ *NONUMBER*)]

ARGUMENTS

EXAMPLE

This is an optional keyword argument. It causes the singular heading Example to be output. It also suppresses numbering of the example. Use this argument when you supply only one example.

heading text

This is an optional argument. It causes the specified text to be used as a heading rather than the default heading Examples.

NOHEAD

This is an optional keyword argument. It suppresses the output of a heading for the section.

NONUMBER

This is an optional keyword argument. It suppresses numbering of the examples. When **EXAMPLE** is supplied as the first argument, the **NONUMBER** argument is unnecessary.

related tags

- <EXAMPLES_INTRO>
- <EXC>
- <EXI>
- <EXTTEXT>

required terminator

<ENDEXAMPLE_SEQUENCE>

DESCRIPTION

The <EXAMPLE_SEQUENCE> tag begins a numbered sequence of informal examples. Use this tag (and the tags it enables to create the examples) inside or outside the reference templates.

Even though the examples are numbered, they are generally not referred to by these numbers. Instead, explanatory text directly follows each example. The example sections in this chapter provide examples of output from the <EXAMPLE_SEQUENCE> tag.

SOFTWARE Doctype Tag Reference

<EXAMPLE_SEQUENCE>

Following the <EXAMPLE_SEQUENCE> tag, each example begins with either the <EXC> or <EXI> tags. Use the <EXC> tag to present code examples; use the <EXI> tag to present interactive examples. Use the <EXTEXT> tag to terminate an example begun using the <EXI> or <EXC> tags and to begin the text that explains the example.

For some doctypes the <EXC> and <EXI> tags use the full page width, which causes the example to begin on the far left of the page. The <EXTEXT> tag, however, causes the text to be left justified at the normal text margin.

EXAMPLE

The following example shows two short interactive examples, a code example, and their explanatory text. The <S> and <U> tags label the system and user portions of the interaction.

```
<example_sequence>(Debugging Examples)
<exi><s>(DBG> )<u>(SCROLL/LEFT)
<extext>This command scrolls the screen left by 8 spaces or columns.
<exi><s>(DBG> )<u>(SCROLL/UP:4)
<extext>This command scrolls 4 lines up through the display.
<exc>PSL:  CMP TP FPD IS CURMOD PRVMOD IPL DV FU IV TN Z V C
          0  0  0  0  USER  USER  0  0  0  1  0  0  0  0
                                     !Display formatted the PSL.
                                     !All bits are cleared.

<extext>
This shows you the contents of the PSL after issuing the EXAMINE command.
<endexample_sequence>
```

This example produces the following output:

DEBUGGING EXAMPLES

1 DBG> SCROLL/LEFT

This command scrolls the screen left by 8 spaces or columns.

2 DBG> SCROLL/UP:4

This command scrolls 4 lines up through the display.

```
3   PSL:  CMP TP FPD IS CURMOD PRVMOD IPL DV FU IV TN Z V C
          0  0  0  0  USER  USER  0  0  0  1  0  0  0  0
                                     !Display formatted the PSL.
                                     !All bits are cleared.
```

This shows you the contents of the PSL after issuing the EXAMINE command.

SOFTWARE Doctype Tag Reference

<EXAMPLES_INTRO>

<EXAMPLES_INTRO>

Provides introductory text before an example.

SYNTAX <EXAMPLES_INTRO>

ARGUMENTS *None.*

related tags

- <EXAMPLE_SEQUENCE>
- <EXC>
- <EXI>
- <EXTTEXT>

restrictions Available only in the context of the <EXAMPLE_SEQUENCE> tag.

DESCRIPTION The <EXAMPLES_INTRO> tag provides introductory text before an example.

EXAMPLE The following example shows introductory text to two interactive examples and their explanatory text.

```
<EXAMPLE_SEQUENCE>(Debugging Examples)
<EXAMPLES_INTRO>
The following examples show various uses of the DBG Utility:
<EXI><S>(DBG)<U>(SCROLL/LEFT)
<EXTTEXT>
This command scrolls the screen left by 8 spaces or columns.
<EXI><S>(DBG)<U>(SCROLL/UP:4)
<EXTTEXT>
This command scrolls 4 lines up through the display.
<ENDEXAMPLE_SEQUENCE>
```

This example produces the following output:

DEBUGGING EXAMPLES

The following examples show various uses of the DBG Utility:

1 DBG>SCROLL/LEFT

This command scrolls the screen left by 8 spaces or columns.

2 DBG>SCROLL/UP:4

This command scrolls 4 lines up through the display.

<EXC>

Begins a code example in a series of numbered informal examples.

SYNTAX <EXC>

ARGUMENTS *None.*

related tags

- <EXAMPLE_SEQUENCE>
- <EXAMPLES_INTRO>
- <EXI>
- <EXTTEXT>
- The global <CODE_EXAMPLE> tag

restrictions

- Valid only in the context of the <EXAMPLE_SEQUENCE> tag.
- The first line of the code example must be on the same source file line as the <EXC> tag.
- Indexing tags such as <X> and <Y> tags are invalid in the context of the code example.

required terminator

<EXTTEXT>

DESCRIPTION

The <EXC> tag begins a code example in a series of numbered informal examples. The text following the <EXC> tag is treated as the text of the code example until you terminate that example with the <EXTTEXT> tag.

Use the <EXC> and <EXTTEXT> tags much like the global <CODE_EXAMPLE> and <ENDCODE_EXAMPLE> tags, with the exception that when using the <EXC> tag to begin a code example, the first line of the code example text must be on the same source file line as the <EXC> tag. This ensures that the code example formats correctly on the page.

EXAMPLE

See the example in the <EXAMPLE_SEQUENCE> tag description.

SOFTWARE Doctype Tag Reference

<EXI>

<EXI>

Begins an interactive example in a series of numbered informal examples.

SYNTAX

<EXI>

ARGUMENTS

None.

related tags

- <EXAMPLE_SEQUENCE>
 - <EXC>
 - <EXTEXT>
 - The global <INTERACTIVE> tag
 - The global <S> tag
 - The global <U> tag
-

restrictions

- Valid only in the context of the <EXAMPLE_SEQUENCE> tag.
 - The first line of the interactive example must be on the same source file line as the <EXI> tag.
-

required terminator

<EXTEXT>

DESCRIPTION

The <EXI> tag begins an interactive example in a series of numbered informal examples. The text following the <EXI> tag is taken as the text of the interactive example until that example is terminated by the <EXTEXT> tag. Use the global <U> and <S> tags in the context of the <EXI> tag in the same way they are used in the context of the global <INTERACTIVE> tag.

The <EXI> and <EXTEXT> tags can be used just like the <INTERACTIVE> and <ENDINTERACTIVE> tags, with the exception that when using the <EXI> tag to begin an interactive example, the first line of the interactive example text must be on the same source file line as the <EXI> tag. This ensures that the interactive example formats correctly on the page.

EXAMPLE

The following example shows how to use the <EXI> tag in the context of the <EXAMPLE_SEQUENCE> tag to begin an interactive example.

SOFTWARE Doctype Tag Reference

<EXI>

To produce a space between the DCL prompt and the user-entered command, include the space in the argument to the <S> tag.

```
<EXAMPLE_SEQUENCE>(DCL Examples)
<EXI><S>($ )<U>(SHOW TIME)
<S>(16-APR-1984 15:18:44)
<EXTEXT>This example shows how to use the DCL command SHOW TIME to
request a display of the date and time.
<ENDEXAMPLE_SEQUENCE>
```

This example produces the following output:

```
$ SHOW TIME
16-APR-1984 15:18:44
```

This example shows how to use the DCL command SHOW TIME to request a display of the date and time.

SOFTWARE Doctype Tag Reference

<EXTEXT>

<EXTEXT>

Terminates an example and begins an explanation in a sequence of numbered examples.

SYNTAX

<EXTEXT>

ARGUMENTS

None.

related tags

- <EXAMPLES_INTRO>
 - <EXC>
 - <EXI>
-

restrictions

Required in the context of an <EXAMPLE_SEQUENCE> tag.

DESCRIPTION

The <EXTEXT> tag terminates an example and begins an explanation in a sequence of numbered examples. In an <EXAMPLE_SEQUENCE> tag section, each numbered example begins with either the <EXC> or <EXI> tag, depending on whether you want a code example or an interactive example. You terminate both kinds of examples with the <EXTEXT> tag, which labels the beginning of the text that explains the previous example.

Use the <EXAMPLES_INTRO> tag to place explanatory text before the first example in the example sequence. See the description of the <EXAMPLES_INTRO> tag for more information.

EXAMPLE

See the examples in the <EXAMPLE_SEQUENCE> and <EXI> tag descriptions.

<FARG>

Adds a single argument line to a list of arguments in a routine syntax format section.

SYNTAX <FARG>(argument name)

ARGUMENTS *argument name*
Specifies a single-line item to be formatted for a routine's argument list.

related tags

- <FARGS>
- <FFUNC>
- <FORMAT>
- <FRTN>
- <ROUTINE_SECTION>

restrictions Valid only in the context of the <FORMAT> tag following an <FRTN> tag and an <FARGS> tag sequence.

DESCRIPTION The <FARG> tag adds a single argument line to a list of arguments in a routine syntax format section.

SOFTWARE Doctype Tag Reference

<FARG>

EXAMPLE

The following example shows how to use the <FARG> tag to format a routine that allows a number of keyword arguments, each of which may specify several values.

```
<format>
<frtn>($FAB)<fargs>(ALQ = allocation quantity,)
<farg>(BKS = bucket size,)
<farg>(DEQ = extension quantity,)
<farg>(DNA = default filespec address,)
<farg>(FNA = filespec string address,)
<farg>(ORG = <list>(stacked\braces)
      <le>REL
      <le>SEQ
      <le>IDX<endlist>,)
<farg>(RAT = <list>(stacked\braces)
      <le>CR
      <le><BLK FTN>
      <le>PRN<endlist>,)
<farg>(RTV = window size,)
<farg>(SHR = <list>(stacked)
      <le><PUT GET DEL
      <le>UPD NIL MSE UPI><endlist>,)
<farg>(XAB = xab address)
<endformat>
```

This example produces the following output:

FORMAT SAMPLE

\$FAB *ALQ = allocation quantity,*
BKS = bucket size,
DEQ = extension quantity,
DNA = default filespec address,
FNA = filespec string address,
*ORG = { REL
 SEQ
 IDX },*
*RAT = { CR
 <BLK FTN>
 PRN },*
RTV = window size,
*SHR = { <PUT GET DEL
 UPD NIL MSE UPI> },*
XAB = xab address

<FARGS>

Provides the argument portion of a routine syntax statement in the context of the <FORMAT> tag.

SYNTAX

<FARGS> (*argument list*[\ *STACK*])

ARGUMENTS

argument list

Lists the routine arguments. If there are no arguments, specify the argument list as null in the following way:

```
<FRTN> ($HIBER) <FARGS> ()
```

Parameters specified in this argument differ on output from parameters specified using the second argument to the <FRTN> tag. For distinctions between these output differences, see the Examples section in the <FRTN> tag description.

STACK

This is an optional keyword argument. It specifies that the routine's argument list is not formatted on the same line as the routine name, but placed on the next line.

This keyword is required only for long routine names in certain doctypes. See if the output looks wrong before you use this argument.

related tags

- <FARG>
- <FFUNC>
- <FRTN>

restrictions

Use the <FARGS> tag with the <FRTN> tag in the context of the <FORMAT> tag in the Routine template.

DESCRIPTION

The <FARGS> tag provides the argument portion of a routine syntax statement in the context of the <FORMAT> tag.

When you use this tag to contain an argument list with multiple words or text strings, enter blank spaces between the words and text strings (including punctuation) according to the syntax rule of the routine you are describing. If the *parameter list* argument text does not fit on one line, suitable line breaks are chosen based on the presence of spaces. Hyphenated text is not broken across lines.

Select explicit line breaks in an argument list by using the <FARG> tag.

SOFTWARE Doctype Tag Reference

<FARGS>

EXAMPLES

The following examples show how to use the <FARG> tag to code a format argument list in the context of a <FORMAT> tag in a routine section.

In the first example, the <FARGS> tag specifies a long argument list. By convention, if routine arguments are normally delimited by commas, you should place blank spaces in your SDML file preceding the commas. This allows suitable page breaks to be chosen during page composition.

1 <FRTN>(LIB\$CRF_OUTPUT) <FARGS>(ctl tbl ,width ,pag1
,pag2 ,mode ind ,del sav ind)

This example produces the following output:

SYNTAX

LIB\$CRF_OUTPUT *ctl tbl ,width ,pag1 ,pag2 ,mode ind
,del sav ind*

The following example shows how to position an argument list when a routine name is very long. You will want to use this form only if examination of your output indicates a formatting problem.

2 <FRTN>(SMG\$BEGIN_PASTEBOARD_UPDATE)
<FARGS>(pasteboard id\stack)

This example produces the following output:

SYNTAX

SMG\$BEGIN_PASTEBOARD_UPDATE
pasteboard id

<FCMD>

Specifies a statement or function keyword and an optional parameter list in the context of the <STATEMENT_FORMAT> tag.

SYNTAX

<FCMD> (*statement keyword*[\ *parameter list*])

ARGUMENTS***statement keyword***

Specifies the name of the first part of the format statement; typically, this is the principal statement name or the function name. This text outputs in the left portion of the format description.

parameter list

This is an optional argument. It specifies one or more parameters of the statement or function keyword. This text outputs to the right of the statement or function in the format description with no space between the *statement keyword* and the *parameter list* text.

Parameters you specify using the *parameter list* argument are output differently than parameters specified using the <FPARMS> tag. See the examples in this tag description for illustrations of these differences.

related tags

- <COMMAND_SECTION>
- <FORMAT>
- <FPARMS>
- <STATEMENT_FORMAT>

restrictions

- Valid only in the context of the <FORMAT> or <STATEMENT_FORMAT> tags in the Statement template.
- If you do not provide a second argument to the <FCMD> tag, you should explicitly declare the absence of parameters by using the <FPARMS> tag as follows:

```
<FCMD>(STATEMENT KEYWORD) <FPARMS>()
```

If you do not specify a second argument to the <FCMD> tag and the <FPARMS> tag is not specified, VAX DOCUMENT issues a warning message.

DESCRIPTION

The <FCMD> tag specifies a statement or function keyword and an optional parameter list in the context of the <STATEMENT_FORMAT> tag. Use the *parameter list* argument to this tag to create a list of one or more parameters that format with no space between the statement or function keyword and the parameter list.

SOFTWARE Doctype Tag Reference

<FCMD>

Use the <FPARMS> tag in conjunction with the <FCMD> tag to create a statement format with the statement or function keyword and the one or more parameters separated by a space.

If the text of the *parameter list* argument does not fit on a single line in the statement format section, the text formatter selects suitable line breaks based on the presence of spaces in the *parameter list* text. Hyphenated text is not broken across lines.

EXAMPLES

The following are five examples of various uses of the <FCMD> tag in the context of the <STATEMENT_FORMAT> tag. Output from these coding examples appear after the last input example.

The first input example specifies a statement keyword with no parameters. Use the <FPARMS> after the <FCMD> to explicitly specify the absence of any parameters.

1 <STATEMENT_FORMAT>
<FCMD>(EXIT) <FPARMS>()
<ENDSTATEMENT_FORMAT>

The second input example also specifies a statement keyword with no parameters. This coding, however, specifies a null second argument to the <FCMD> tag rather than using the <FPARMS> tag to explicitly specify the absence of any parameters.

2 <STATEMENT_FORMAT>
<FCMD>(EXIT\)
<ENDSTATEMENT_FORMAT>

The third input example specifies both the *statement keyword* and the *parameter list* arguments to the <FCMD> tag. Note in the output sample how these two arguments format together with no intervening spaces.

3 <STATEMENT_FORMAT>
<FCMD>(RETURN\ident_field, numeric_field)
<ENDSTATEMENT_FORMAT>

The fourth input example specifies a statement keyword with a single parameter using the <FCMD> and <FPARMS> tags. Note that coding the parameters using the <FPARMS> tag outputs a space between the command keyword and the parameter.

4 <STATEMENT_FORMAT>
<FCMD>(RECORD) <FPARMS>(rec nam)
<ENDSTATEMENT_FORMAT>

The fifth input example specifies the <FCMD> tag using the global <KEYWORD> tag as part of the *parameter list* argument text. Note how the global <KEYWORD> tag alters the output.

5 <STATEMENT_FORMAT>
<FCMD>(RETURN\ident_field, numeric_field,<keyword>([/LENGTH]))
<ENDSTATEMENT_FORMAT>

These input examples produce the following outputs:

Format

EXIT

Format

Format

RETURN*ident_field, numeric_field*

Format

RECORD *rec nam*

Format

RETURN*ident_field, numeric_field*,[/LENGTH]

SOFTWARE Doctype Tag Reference

<FFUNC>

<FFUNC>

Labels a function in the context of a <FORMAT> or <STATEMENT_FORMAT> tag section.

SYNTAX

<FFUNC>({ *function name*[\ *arg list*]
 return value \ *function name*[\ *arg list*] })

ARGUMENTS

function name

Specifies the name of the function.

arg list

This is an optional argument. It specifies the function's argument list.

return val

This is an optional argument. It specifies a variable name to which a function returns its value.

related tags

- <STATEMENT_FORMAT>

restrictions

Valid only in the context of the <STATEMENT_FORMAT> tag in the Statement template.

DESCRIPTION

The <FFUNC> tag labels a function in the context of a <FORMAT> or <STATEMENT_FORMAT> tag section. The <FFUNC> tag lets you create a syntax that includes the following:

- The name of the function
- The return value of the function
- Any arguments that the function may require

EXAMPLES

The following two input examples show various uses of the <FCMD> tag in the context of the <STATEMENT_FORMAT> tag. Output from these coding examples appear after the second input example.

The first input example shows a statement format in which a function has several forms.

```
❏ <STATEMENT_FORMAT>  
<ffunc>(real vbl\=ABS<VARIABLE>((real exp))  
<ENDSTATEMENT_FORMAT>
```

The second input example shows the 3-argument form of the <FFUNC> tag.

SOFTWARE Doctype Tag Reference

<FFUNC>

2 <STATEMENT_FORMAT>
<FFUNC>(status\=AAA\$CODE\ (arg-one , arg-two ,arg-three ,arg-four ,arg-five
,arg-six ,arg-seven))
<ENDSTATEMENT_FORMAT>

These input examples produce the following output:

Format

real vbl=ABS(*real exp*)

Format

**status=AAA\$CODE(*arg-one , arg-two ,arg-three ,arg-four ,arg-five*
,arg-six ,arg-seven)**

SOFTWARE Doctype Tag Reference

<FORMAT>

<FORMAT>

Begins a section that illustrates the syntax of a routine, command, or tag, including keywords and arguments.

SYNTAX

<FORMAT>*[(alternate heading)]*

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading for this use of the <FORMAT> tag. The default heading provided by VAX DOCUMENT is Format. See the reference description of the <SET_TEMPLATE_HEADING> tag for information on how to modify the default headings for all <FORMAT> tags.

related tags

- <FARG>
- <FARGS>
- <FFUNC>
- <FRTN>

required terminator

<ENDFORMAT>

DESCRIPTION

The <FORMAT> tag begins a section that illustrates the syntax of a routine, command, or tag, including keywords and arguments. This tag is a global tag and is not restricted to routine sections. However, in the context of the <ROUTINE_SECTION> and <TAG_SECTION> tags, the <FORMAT> tag enables certain additional tags for specialized format descriptions. These additional tags are not available outside the context of routine or tag sections.

The global <FORMAT> tag enables the <FCMD>, <FPARMS>, and <FPARM> tags, which label specific portions of a format statement. In a routine section, the <FORMAT> tag also enables these additional tags.

You can use the <FORMAT> tag and the tags it enables in a variety of ways to show the syntax of a routine. The following list of code examples show some of the most regularly used format section tag combinations:

- *<FRTN>(routine keyword) <FARGS>(argument list)*

This is the standard form, in which the routine keyword and its argument list are separated by a blank space. If the output argument list is more than a single line, additional lines are aligned at the beginning of the argument list.

SOFTWARE Doctype Tag Reference

<FORMAT>

- *<FRTN>(routine keyword\argument list)*

Use this form for routine functions, in which a routine and its arguments are not separated by blank spaces.

- *<FRTN>(return val\routine keyword\argument list)*

This form provides three distinct elements for a routine description: the return value, the function name (shown here as the routine keyword), and the argument list.

- *<FRTN>(keyword part) <FARGS>(argument-1) <FARG>(argument-2)*

This form is useful for routines with long argument lists and in which the argument names themselves are either long or need additional information.

- *<FFUNC>(routine keyword\argument list)*

This form does not separate the routine name from its argument list and so is appropriate for function routines.

- *<FFUNC>(return val\routine keyword\argument list)*

This form provides three distinct values: the return value, the function name, and the argument list.

EXAMPLES

The following example shows how to use the <FRTN> and <FARGS> tags in the context of the <FORMAT> tag to format a routine name and argument list. In this example, the argument list is null.

```
1 <FORMAT>
  <FRTN>(SYS$HIBER) <FARGS>( )
<ENDFORMAT>
```

The following example shows how to use multiple <FFUNC> tags in a format section.

```
2 <FORMAT>
  <ffunc>(MTH$SQRT\ (x))
  <ffunc>(MTH$DSQRT\ (x))
  <ffunc>(MTH$GSQRT\ (x))
  <ffunc>(MTH$HSQRT\ (x))
<ENDFORMAT>
```

SOFTWARE Doctype Tag Reference

<FORMAT_SUBHEAD>

<FORMAT_SUBHEAD>

Introduces one of a set of multiple formats in a statement template.

SYNTAX <FORMAT_SUBHEAD> (*heading text*)

ARGUMENTS *heading text*
Specifies the text of the heading to be used before a specific format presentation.

related tags

- <FCMD>
- <FPARMS>
- <STATEMENT_FORMAT>

restrictions Valid only in the context of the <STATEMENT_FORMAT> tag if you specify the MULTIPLE keyword as the second argument to the <STATEMENT_FORMAT> tag in the Statement template.

DESCRIPTION The <STATEMENT_FORMAT> tag introduces one of a set of multiple formats in a statement template.

EXAMPLE The following example shows how to use the <FORMAT_SUBHEAD> tag in a series of formats for a single statement.

```
<statement_format>(\multiple)
<FORMAT_SUBHEAD>(String Variable To Array)
<FCMD>(CHANGE) <FPARMS>(str exp <KEYWORD>(TO) num array)
<FORMAT_SUBHEAD>(Array to String Variable)
<FCMD>(CHANGE) <FPARMS>(num array <KEYWORD>(TO) str vbl)
<endstatement_format>
```

This example produces the following output:

Format

String Variable To Array

CHANGE *str exp TO num array*

Array to String Variable

CHANGE *num array TO str vbl*

<FPARM>

Adds a single parameter line to a list of parameters in a command or statement syntax format section.

SYNTAX <FPARM>(parameter)

ARGUMENTS *parameter*
Adds a single parameter to a parameter list in a command or statement format section.

related tags

- <FCMD>
- <FPARMS>
- <STATEMENT_FORMAT>
- <STATEMENT_SECTION>

restrictions Valid only after an <FCMD> tag and <FPARMS> tag sequence in a format section in the Command and Statement templates.

DESCRIPTION The <FPARM> tag adds a single parameter line to a list of parameters in a command or statement syntax format section.

EXAMPLE The following example shows how to use the <FPARM> tag.

```
<format>
<fcmd>(SET TERMINAL)
<fparms>([device name[:]])
<fparm>/DEVICE_TYPE=<list>(stacked\braces)
           <le><keyword>(UNKNOWN)
           <le><keyword>(LA34)
           <le><keyword>(VT100)
           <le><keyword>(PRO_SERIES)<endlist>)
<fparm>(<list>(stacked\brackets)
           <le>LINE_EDITING
           <le>NOLINE_EDITING<endlist>)
<fparm>(<list>(stacked\brackets)
           <le>PASSALL
           <le>NOPASSALL<endlist>)
<ENDFORMAT>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<FPARM>

SYNTAX

SET TERMINAL [*device name[:]*]

/DEVICE_TYPE= { UNKNOWN
LA34
VT100
PRO_SERIES }

[*/LINE_EDITING*
/NOLINE_EDITING]

[*/PASSALL*
/NOPASSALL]

<FPARMS>

Specifies the parameters to a command or statement keyword.

SYNTAX

<FPARMS>(parameter)

ARGUMENTS

parameter

Specifies the parameters to a command or statement keyword in a format section.

Use this command also to explicitly declare that a command or keyword marked with the <FCMD> tag has no parameters, as shown in this example.

```
<FCMD>(EXIT) <FPARMS>()
```

Parameters you specify using the <FPARMS> tag are printed differently than parameters specified in the parameter list argument to the <FCMD> tag. See the examples in this tag description for illustrations of the different output styles.

related tags

- <COMMAND_SECTION>
- <FCMD>
- <FORMAT>
- <FPARM>
- <STATEMENT_FORMAT>

restrictions

Valid only in the context of the Command and Statement templates. The <FPARMS> tag should be used with the <FCMD> tag in the context of the <FORMAT> tag.

DESCRIPTION

The <FPARMS> tag specifies the parameters to a command or statement keyword. The command or keyword is identified with the <FCMD> tag. These tags should only be used in a format section. That is, they should only occur following the <FORMAT> tag and before the <ENDFORMAT> tag.

When you use this tag to define a parameter list that contains multiple words or text strings, include blank spaces between the words and text strings (including punctuation) according to the syntax rules of the command you are describing.

If the text of the *parameter list* argument will not fit on a single line in the format section, the text formatter selects suitable line breaks based on the presence of spaces in the *parameter list* text. Hyphenated text does not break across lines.

Use the <FPARM> tag to select explicit line breaks in a parameter list.

SOFTWARE Doctype Tag Reference

<FPARMS>

EXAMPLES

The following two input examples show how to use the <FPARMS> tag. Output from these coding examples appear after the second input example.

The following input example uses the <FPARMS> tag to list additional command keywords in the parameters portion of a command format.

1 <FCMD>(ON)
 <FPARMS>(condition <keyword>(THEN [\$]) command)

The following input example shows how to format a line in which command parameters appear before the command keyword.

2 <FCMD>()<FPARMS>(input_specifier output_specifier<keyword>(/OVERLAY))

These input examples produce the following output:

FORMAT **ON** *condition* **THEN** [**\$**] *command*

FORMAT *input_specifier* *output_specifier*/**OVERLAY**

<FRTN>

Specifies the routine-keyword portion of a routine syntax statement in the context of the <FORMAT> tag.

SYNTAX

<FRTN> (*[return val \]routine keyword[\ argument list]*)

ARGUMENTS***return val***

This is an optional argument. It indicates that the routine's format includes the descriptive name of a variable. The routine returns a value to this argument. Note that this argument is optional.

routine keyword

Specifies the name of the first part of the format statement; typically, this is the routine name or an operating system keyword.

argument list

This is an optional argument. It specifies the arguments to be listed in the format statement.

Parameters specified in this argument are different from parameters specified using the <FARGS> tag. The Examples section in this description shows the differences.

related tags

- <FARG>
- <FARGS>
- <FFUNC>

restrictions

If you specify <FRTN> with a null second argument, you should explicitly declare the absence of arguments using the <FARGS> tag as follows:

```
<FRTN> (KEYWORD PART) <FARGS> ()
```

If you do not specify a second argument and do not specify the <FARGS> tag, VAX DOCUMENT issues a warning message.

DESCRIPTION

The <FRTN> tag specifies the routine-keyword portion of a routine syntax statement in the context of the <FORMAT> tag.

EXAMPLES

The following four input examples show various uses of the <FRTN> tag. Output from these coding examples appears after the last input example.

<FTAG>

Specifies the name of a tag and its arguments in the context of a <FORMAT> tag.

SYNTAX

<FTAG>(*tag name*[\ *argument list* [\ *OPTIONAL*]])

ARGUMENTS

tag name

Is the name of the tag. This name must be a valid tag name.

argument list

This is an optional argument. It lists the arguments, if any. If you do not specify this argument, you indicate that the tag has no arguments.

When you specify arguments, use the <ARG_SEP> tag to denote the argument separator character, the backslash (\).

OPTIONAL

This is an optional keyword argument. When you specify this keyword, the *argument list* argument is placed in square brackets to indicate that those arguments are optional.

related tags

- <ARG_SEP>
- <FORMAT>

restrictions

Valid only in a <FORMAT> tag section in the Tag template.

DESCRIPTION

The <FTAG> tag specifies the name of a tag and its arguments in the context of a <FORMAT> tag. This tag uses the <ARG_SEP> tag to create tag separator characters (\) in the *argument list* argument.

The <ARG_SEP> tag has no other function than to output the tag separator character (the backslash) in a Tag template format section.

EXAMPLES

The following four input examples show various uses of the <FTAG> tag. Output from these coding examples appears after the last input example.

The following input example shows how to use the <FTAG> tag to show the syntax of a tag that has no argument list.

1 <format>(Syntax)
<ftag>(OVERVIEW)
<endformat>

The following input example illustrates how to specify the format of a tag that has a single optional argument by using the OPTIONAL keyword.

SOFTWARE Doctype Tag Reference

<FTAG>

2

```
<format>
<ftag>(DESCRIPTION\heading text\OPTIONAL)
<endformat>
```

The following input example illustrates how to specify a tag that has one required and two optional arguments. Note how the <ARG_SEP> tag separates the arguments.

3

```
<format>
<ftag>(ROUTINE\name[<arg_sep>info-1[<arg_sep>info2]])
<endformat>
```

The following input example shows a tag format in which all of the tag's arguments are optional and positional.

4

```
<format>
<ftag>(ROUTINE_SECTION\[running title]<line>
[<arg_sep>prefix]<line>
[<arg_sep>NEWPAGE]\OPTIONAL)
<endformat>
```

These input examples produce the following output:

FORMAT **<OVERVIEW>**

FORMAT **<DESCRIPTION>***[(heading text)]*

FORMAT **<ROUTINE>***(name[\ info-1[\ info2]])*

FORMAT **<ROUTINE_SECTION>***[[[running title]
[\ prefix]
[\ NEWPAGE]]]*

<FUNCTION>

Begins a new function description.

SYNTAX <FUNCTION> (*function name*)

ARGUMENTS *function name*
Specifies the name of the function to be described.

related tags

- <SET_TEMPLATE_STATEMENT>
- <STATEMENT>
- <STATEMENT_SECTION>

DESCRIPTION The <FUNCTION> tag begins a new function description. This description is for a single function in the context of the <STATEMENT_SECTION> tag. This tag has the following default format:

- Each <FUNCTION> tag begins a new page of output.
- Each output page carries a single running title, which is the current function name.

Use the <SET_TEMPLATE_STATEMENT> tag to replace the <FUNCTION> tag with a tag specific to your task (for example, <ABC_FUNCTION>), or to change the default attributes of the <FUNCTION> tag. See the description of the <SET_TEMPLATE_STATEMENT> tag in this chapter for more information.

Note that the <FUNCTION> and the <STATEMENT> tags work in exactly the same manner. The two tag names are provided so that you may encode your source file more generically.

EXAMPLE

In the following example, the <STATEMENT_SECTION> tag enables the tags for a function description. The description of the function OPEN will, by default, have the following attributes:

- The function description begins on a new page.
- If the function carries for more than a page, the name OPEN is carried as a running top title on each page.

```
<STATEMENT_SECTION>  
<FUNCTION> (OPEN)  
<OVERVIEW>
```

SOFTWARE Doctype Tag Reference

<GRAPHIC>

<GRAPHIC>

Displays terminal graphics characters.

SYNTAX **<GRAPHIC>**(*char-1* \ *char-2*)

ARGUMENTS ***char-1***
Specifies the character to be used as the top portion of the graphics character.

char-2
Specifies the character to be used as the bottom portion of the graphics character.

related tags • <KEY>

DESCRIPTION The <GRAPHIC> tag displays terminal graphics characters. It creates a single graphics terminal character (such as the linefeed or formfeed characters) by combining the two characters you specify as arguments. The second character you specify appears lower and adjacent to the first character.

EXAMPLE The following example shows how to use the <GRAPHIC> tag to create the linefeed and formfeed characters that can appear on a computer terminal.

<P>
The <GRAPHIC>(F\F) and the <GRAPHIC>(L\F)
are two characters that the terminal displays.

This example produces the following output:

The F_F and the L_F are two characters that the terminal displays.

<KEY>

Depicts a key from a keyboard or keypad graphically.

SYNTAX

<KEY>(*key label-1*[*\ key label-2*]{ [*\ BOX*] }
{ [*\ TEXT*] })

ARGUMENTS***key label-1***

Labels the key.

key label-2

This is an optional argument. It stacks a second key label under *key label-1*.

BOX

This is an optional keyword argument. It draws a box around the key labels for devices that support this feature. BOX is the default key format.

TEXT

This is an optional keyword argument. It encloses the key labels in angle brackets. This format is useful when you specify keys in a body of text.

related tags

- <GRAPHIC>
 - <KEY_NAME>
 - <KEY_PLUS>
 - <KEY_SEQUENCE>
-

restrictions

Only specify the argument *key label-2* when using the <KEY> tag in a key sequence example. For more information, refer to the <KEY_SEQUENCE> tag in this section.

Use the BOX keyword argument only for devices that support graphics (such as laser printers).

DESCRIPTION

The <KEY> tag depicts a key from a keyboard or keypad graphically. If you are using the <KEY> tag with the <KEY_SEQUENCE> tag, you can specify a second label that this tag stacks under the first.

The optional keyword arguments BOX and TEXT determine how the key labels appear in your document. If you specify BOX (the default), this tag draws a box around the labels. If you specify TEXT, this tag encloses the labels in angle brackets.

SOFTWARE Doctype Tag Reference

<KEY>

EXAMPLE

In the following example, note that in the context of the <KEY_SEQUENCE> tag, the <KEY> tag accepts two *key label-n* arguments. Outside the context of the <KEY_SEQUENCE> tag, it accepts only a single *key label-n* argument.

Note also that the first <KEY> tag is specified with no keyword argument. This tag uses the default keyword argument BOX. The second <KEY> tag includes the BOX keyword argument to specify BOX formatting. The third <KEY> tag includes the TEXT keyword argument.

```
<P>
You would use the following sequence of keys:
<KEY_SEQUENCE>
<KEY>(Next\Screen) <KEY_PLUS> <KEY>(PF3\BOX)
<ENDKEY_SEQUENCE>
<P>
These keys are not associated with the <KEY>(WHITE\TEXT) keys.
```

This example produces the following output:

You would use the following sequence of keys:

Next Screen	+	PF3
----------------	---	-----

These keys are not associated with the <WHITE> keys.

<KEYPAD>

Specifies an individual keypad diagram and optionally supplies a title for that diagram.

SYNTAX

<KEYPAD>[({ *alternate heading*[\ *DISPLAY*] })]
DISPLAY

ARGUMENTS

alternate heading

Specifies the text of a heading for a keypad diagram.

DISPLAY

This is an optional keyword argument that lets you specify individual key labels as arguments to the tags <KEYPAD_ROW> and <KEYPAD_ENDROW>.

related tags

- <KEYPAD_ENDROW>
- <KEYPAD_ROW>
- <KEYPAD_SECTION>

restrictions

Valid only in the context of the <KEYPAD_SECTION> tag.

required terminator

<ENDKEYPAD>

DESCRIPTION

The <KEYPAD> tag specifies an individual keypad diagram and optionally supplies a title for that diagram. If you use the DISPLAY keyword argument, the <KEYPAD> tag lets you specify individual key labels as arguments to the <KEYPAD_ROW> and <KEYPAD_ENDROW> tags. For more information, see the <KEYPAD_ROW> and <KEYPAD_ENDROW> tag descriptions in this chapter.

EXAMPLES

The following examples show three keypads.

The following example does not include the DISPLAY keyword argument.

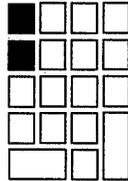
```
❏ <KEYPAD_SECTION>
<KEYPAD>(Using the Command Keypad Function)
<KEYPAD_ROW>(CLOSED\\)
<KEYPAD_ROW>(CLOSED\\)
<KEYPAD_ROW>(\\)
<KEYPAD_ROW>(\\)
<KEYPAD_ENDROW>(\\)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

SOFTWARE Doctype Tag Reference

<KEYPAD>

This example produces the following output:

Using the Command Keypad Function



The following example includes the DISPLAY keyword argument.

```
2 <KEYPAD_SECTION>
<KEYPAD>(The Editing Keypad with Key Labels\DISPLAY)
<KEYPAD_ROW>(PF1\PF2\PF3\PF4)
<KEYPAD_ROW>(7\8\9\-)
<KEYPAD_ROW>(4\5\6\,)
<KEYPAD_ROW>(1\2\3\ )
<KEYPAD_ENDROW>(0\.\ENTER)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

The Editing Keypad with Key Labels

PF1	PF2	PF3	PF4
7	8	9	-
4	5	6	.
1	2	3	ENTER
0		.	

The following example includes the DISPLAY keyword argument, and shades two keys.

```
3 <KEYPAD_SECTION>
<KEYPAD>(The Editing Keypad with Key Labels and Shaded Keys\DISPLAY)
<KEYPAD_ROW>(CLOSED\PF2\PF3\PF4)
<KEYPAD_ROW>(7\8\9\-)
<KEYPAD_ROW>(CLOSED\5\6\,)
<KEYPAD_ROW>(1\2\3\ )
<KEYPAD_ENDROW>(0\.\ENTER)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<KEYPAD>

The Editing Keypad with Key Labels and Shaded Keys

	PF2	PF3	PF4
7	8	9	-
	5	6	.
1	2	3	ENTER
0	.		

SOFTWARE Doctype Tag Reference

<KEYPAD_ENDROW>

<KEYPAD_ENDROW>

Displays the bottom row of an editing keypad that has up to three keys.

SYNTAX **<KEYPAD_ENDROW>**(*col-1 arg* \ *col-2 arg* \ *col-3 arg*)

ARGUMENTS ***col-1 arg***
col-2 arg
col-3 arg
Represents one of the three keys that make up the bottom row in the editing keypad. By default, each column accepts one of the following three arguments:

- **CLOSED** — Shades the key in that column.
- **NONE** — No key appears in that column.
- **OPEN** — Creates an unshaded box. This is the default.

If you use the **DISPLAY** keyword argument to the **<KEYPAD>** tag, you can specify alphanumeric strings as individual key labels.

related tags

- **<KEYPAD>**
- **<KEYPAD_ROW>**
- **<KEYPAD_SECTION>**

restrictions

Valid only in the context of a **<KEYPAD_SECTION>** tag.

DESCRIPTION The **<KEYPAD_ENDROW>** tag displays the bottom row of an editing keypad that has up to three keys. The first key is double in width, the second is standard size, and the third is connected with the last key in the previous keypad row. For more information, refer to the examples in the description of **<KEYPAD_SECTION>** in this section.

EXAMPLE

See the example in the **<KEYPAD_SECTION>** tag description.

<KEYPAD_ROW>

Displays a row of an editing keypad that has up to four keys.

SYNTAX **<KEYPAD_ROW>**(*col-1 arg* | *col-2 arg* | *col-3 arg* | *col-4 arg*)

ARGUMENTS *col-1 arg*
 col-2 arg
 col-3 arg
 col-4 arg

Represents one of the four keys that make up a row in the editing keypad. By default, each column accepts one of the following three arguments:

- CLOSED — This tag shades the key in that column.
- NONE — No key appears in that column.
- OPEN — This tag creates an unshaded box. This is the default.

If you use the DISPLAY keyword argument in the <KEYPAD> tag, you can specify alphanumeric strings as individual key labels.

related tags

- <KEYPAD>
- <KEYPAD_ENDROW>
- <KEYPAD_SECTION>

restrictions

Valid only in the context of a <KEYPAD_SECTION> tag.

When this tag is used just before the <KEYPAD_ENDROW> tag, the *col-4 arg* is ignored to leave room for the large ENTER key created by the third argument to the <KEYPAD_ENDROW> tag. See the example in the discussion of the <KEYPAD_SECTION> tag for more information.

DESCRIPTION

The <KEYPAD_ROW> tag displays a row of an editing keypad that has up to four keys. For more information, refer to the examples in the description of the <KEYPAD_SECTION> tag in this section.

EXAMPLE

See the example in the <KEYPAD_SECTION> tag description.

SOFTWARE Doctype Tag Reference

<KEYPAD_SECTION>

<KEYPAD_SECTION>

Begins a series of one or more keypad diagrams.

SYNTAX

<KEYPAD_SECTION>

related tags

- <KEYPAD>
- <KEYPAD_ENDROW>
- <KEYPAD_ROW>

restrictions

Only create files containing keypad diagrams for output on devices that support graphics (such as laser printers).

required terminator

<ENDKEYPAD_SECTION>

DESCRIPTION

The <KEYPAD_SECTION> tag begins a series of one or more keypad diagrams. This tag enables the <KEYPAD>, <KEYPAD_ENDROW>, and <KEYPAD_ROW> tags to graphically depict one or more terminal editing keypads. You must begin each of the keypads (or partial keypads) illustrated in a keypad section with the <KEYPAD> tag and terminate it with the <ENDKEYPAD> tag.

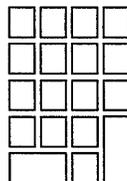
EXAMPLES

The following examples show several uses of the <KEYPAD_SECTION> tag. In the first example, notice how you use the <KEYPAD_ENDROW> tag to create the large-sized keys found on many calculator keypads.

```
1 <KEYPAD_SECTION>
  <KEYPAD>(The VT200 Series Editing Keypad)
  <KEYPAD_ROW>(\\)
  <KEYPAD_ROW>(\\)
  <KEYPAD_ROW>(\\)
  <KEYPAD_ROW>(\\)
  <KEYPAD_ROW>(\\)
  <KEYPAD_ENDROW>(\\)
  <ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

The VT200 Series Editing Keypad



SOFTWARE Doctype Tag Reference

<KEYPAD_SECTION>

The following example shows two keypads created in a single keypad section.

The first keypad shows a command keypad with two keys filled in.

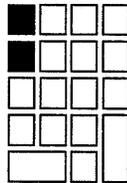
```
2 <KEYPAD_SECTION>
<KEYPAD>(Using the Command Keypad Function)
<KEYPAD_ROW>(CLOSED\\)
<KEYPAD_ROW>(CLOSED\\)
<KEYPAD_ROW>(\\)
<KEYPAD_ROW>(\\)
<KEYPAD_ENDROW>(\\)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

The second keypad is an irregularly shaped keypad.

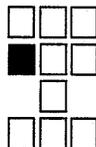
```
3 <KEYPAD_SECTION>
<KEYPAD>(Using the SELECT Key)
<KEYPAD_ROW>(\\NONE)
<KEYPAD_ROW>(CLOSED\\NONE)
<KEYPAD_ROW>(NONE\\NONE\\NONE)
<KEYPAD_ROW>(\\NONE)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

Using the Command Keypad Function



Using the SELECT Key



The following example shows how to create a keypad with the key names placed on each keypad key. Note how the DISPLAY keyword argument is used with the <KEYPAD> tag.

```
4 <KEYPAD_SECTION>
<KEYPAD>(The Editing Keypad with Key Labels\DISPLAY)
<KEYPAD_ROW>(PF1\PF2\PF3\PF4)
<KEYPAD_ROW>(7\8\9\-)
<KEYPAD_ROW>(4\5\6\,)
<KEYPAD_ROW>(1\2\3\ENTER)
<KEYPAD_ENDROW>(0\.\ENTER)
<ENDKEYPAD>
<ENDKEYPAD_SECTION>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<KEYPAD_SECTION>

The Editing Keypad with Key Labels

PF1	PF2	PF3	PF4
7	8	9	.
4	5	6	.
1	2	3	ENTER
0		.	

<KEY_NAME>

Emphasizes the name of a key in text.

SYNTAX <KEY_NAME>(key name)

ARGUMENTS *key name*
Specifies the name of the key. This often corresponds to the name on a key label.

related tags

- <CPOS>
- <GRAPHIC>
- <KEY>
- <KEY_SEQUENCE>

DESCRIPTION The <KEY_NAME> tag emphasizes the name of a key in text. This tag alters the appearance of a key name in text using uppercase letters, boldface, or italics.

EXAMPLE In the following example, the <KEY_NAME> tag alters the appearance of the GOLD key.

<P>
In EDT, the <KEY_NAME>(GOLD) key changes the function of the other keys on the keypad.

This example produces the following output:

In EDT, the GOLD key changes the function of the other keys on the keypad.

SOFTWARE Doctype Tag Reference

<KEY_PLUS>

<KEY_PLUS>

Creates a plus sign between keys in a key sequence example.

SYNTAX <KEY_PLUS>

ARGUMENTS *None.*

related tags

- <KEY>
- <KEY_SEQUENCE>
- <KEY_TYPE>

restrictions Valid only in the context of the <KEY_SEQUENCE> tag.

DESCRIPTION The <KEY_PLUS> tag creates a plus sign between keys in a key sequence example.

EXAMPLE The following example shows how you use the <KEY_PLUS> tag to create a plus sign between the BREAK and F10 keys in the context of the <KEY_SEQUENCE> tag.

```
<KEY_SEQUENCE>  
<KEY>(BREAK) <KEY_PLUS> <KEY>(F10) = Exit from Function.  
<ENDKEY_SEQUENCE>
```

This example produces the following output:

BREAK + **F10** = Exit from Function.

<KEY_SEQUENCE>

Begins a section containing one or more key representations.

SYNTAX <KEY_SEQUENCE>

ARGUMENTS *None.*

related tags

- <CPOS>
- <GRAPHIC>
- <KEY>
- <KEY_PLUS>
- <KEY_TYPE>

restrictions Not available in the context of the example-producing tags (<CODE_EXAMPLE>, <DISPLAY>, <SYNTAX>, and so on) or the <MATH> tag.

required terminator <ENDKEY_SEQUENCE>

DESCRIPTION The <KEY_SEQUENCE> tag begins a section containing one or more key representations. This tag labels an informal example of keys and sequences of keys, and enables two tags that make it easier for you to combine keys and text in your document. The two enabled tags are <KEY_PLUS> and <KEY_TYPE>. For more information concerning these tags, refer to the program example in this section, or refer to the <KEY_PLUS> and <KEY_TYPE> tag descriptions in this section.

EXAMPLES The following examples show how to create informal examples of keys and key sequences in your documentation.

The following example shows how you use the <KEY_PLUS> and the <KEY> tags.

1 <KEY_SEQUENCE>
<key>(HELP) = <key>(PF1) <KEY_PLUS> <KEY>(PF2)
<ENDKEY_SEQUENCE>

This example produces the following output:

HELP = **PF1** + **PF2**

SOFTWARE Doctype Tag Reference

<KEY_SEQUENCE>

The following example shows how you use other tags in a <KEY_SEQUENCE> tag section and then gives a sample of that output.

```
2 <KEY_SEQUENCE>
The <CPOS>(q)uick brown fox
jumped over the lazy dog.
<KEY>(ADV)<KEY>(WORD)
The quick <CPOS>(b)rown fox
jumped over the lazy dog.
<KEY>(WORD)
The quick brown <CPOS>(f)ox
jumped over the lazy dog.
<KEY>(DEL W)
The quick brown <CPOS>( )
jumped over the lazy dog.
<ENDKEY_SEQUENCE>
```

This example produces the following output:

```
The quick brown fox
jumped over the lazy dog.
[ADV|WORD]
The quick brown fox
jumped over the lazy dog.
[WORD]
The quick brown fox
jumped over the lazy dog.
[DEL W]
The quick brown  
jumped over the lazy dog.
```

<KEY_TYPE>

Provides a descriptive label for keys in a key sequence.

SYNTAX <KEY_TYPE> (*labeling text string* \ *key text string*)

ARGUMENTS ***labeling text string***
Labels the keys and text specified in the *key text string* argument.

key text string
Specifies keys and related text.

related tags

- <GRAPHIC>
- <KEY_PLUS>
- <KEY_SEQUENCE>

restrictions Valid only in the context of a <KEY_SEQUENCE> tag.

DESCRIPTION The <KEY_TYPE> tag provides a descriptive label for keys in a key sequence. For instance, if you discuss keys from various keyboards (such as the VT125, VT240, and so forth), you may want to label the different types of keys as you describe them in a <KEY_SEQUENCE> tag example.

The <KEY_TYPE> tag outputs the *labeling text string* argument in a bold face font and appends a colon (:) to it. The *key text string* argument outputs on the same line in the standard text font.

EXAMPLE The following example shows how to use the <KEY_TYPE> tag to label a sequence of keys.

```
<KEY_SEQUENCE>  
<KEY_TYPE>(LK201\<KEY>(F10) = Exit from Function.)  
<ENDKEY_SEQUENCE>
```

This example produces the following output:

LK201: **F10** = Exit from Function.

SOFTWARE Doctype Tag Reference

<MESSAGE_SECTION>

<MESSAGE_SECTION>

Begins a section of error message descriptions.

SYNTAX <MESSAGE_SECTION>

ARGUMENTS *None.*

related tags

- <MESSAGE_TYPE>
- <MSG>
- <MSGS>
- <MSG_ACTION>
- <MSG_FACILITY>
- <MSG_SEVERITY>
- <MSG_TEXT>

required terminator

<ENDMESSAGE_SECTION>

DESCRIPTION

The <MESSAGE_SECTION> tag begins a section of error message descriptions. This tag enables the message tags listed as related tags in this tag description.

Messages generally come in one of three forms:

- A message text string only; for example, System Resources Unavailable.
- A message text string preceded by a text string identification code; for example, %DIRECT-W-NOFILES, no files found.
- A message text string preceded by a numeric string identification code; for example, %1288374 file lookup failed.

Select the tags you need to use in the message description section based upon the following criteria:

- The format your messages most closely resemble
- The number of lines each message occupies on the terminal display

Specify the format for your message section by using the <MESSAGE_TYPE> tag. The <MESSAGE_TYPE> tag accepts one of three keywords:

- **NOIDENT**—Specifies that the message contains no message identification string and that only message text will be specified.

SOFTWARE Doctype Tag Reference

<MESSAGE_SECTION>

- **TEXTIDENT**—Specifies that the message contains a text message identification string as well as message text.
- **NUMIDENT**—Specifies that the message contains a numeric message identification string as well as message text.

Select one of two message labeling tags (<MSG> or <MSGS>) based upon the number of messages or message text lines you need to describe in a single tag.

- <MSG>

Describes a single message with one or two lines of message text.

- <MSGS>

Describes either a single message with up to nine lines of message text, or up to four message identification string/message text pairs.

Use the <MSG_TEXT> tag to describe the error messages. Each use of the <MSG_TEXT> tag creates a separate description section that you can label with a single word heading (such as Facility or Severity). If you do not specify a heading for this tag, it uses the heading, Explanation:. You can use the <MSG_TEXT> tag as many times as you find necessary in the error message description section.

To complete your error message section, repeat using the <MSG> (or <MSGS>) and <MSG_TEXT> tags until you describe all your messages. End the error message section by using the <ENDMESSAGE_SECTION> tag.

EXAMPLES

The following examples illustrate the two syntaxes available with the <MSG> tag and how they are specified. The second example also shows a sample use of the <MSGS> tag so you can compare the output of the <MSG> and <MSGS> tags.

In the first example, the NOIDENT keyword argument is used with the <MESSAGE_TYPE> tag. Therefore, the <MSG> tag only accepts two arguments.

In the second example, the TEXTIDENT keyword is used with the <MESSAGE_TYPE> tag. Therefore, the <MSG> tags in this section will accept one message identifier and up to two message text arguments. Note also that the <MSGS> tag used in this example accepts two pairs of message identifier/message text arguments.

```
1 <MESSAGE_SECTION>
  <MESSAGE_TYPE>(NOIDENT)
  <MSG>(error initiating system\initialization file not found)
  <MSG_TEXT>
  The system could not begin operation because it could not find
  the system initialization file.
  <MSG_TEXT>(User Action)
  Check for the existence of the system initialization file.
  If it exists, check that it is in your current default directory.
  <ENDMESSAGE_SECTION>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference

<MESSAGE_SECTION>

error initiating system
initialization file not found

Explanation: The system could not begin operation because it could not find the system initialization file.

User Action: Check for the existence of the system initialization file. If it exists, check that it is in your current default directory.

2

```
<MESSAGE_SECTION>
<MESSAGE_TYPE>(TEXTIDENT)
<MSG>(BCK-F-BADLNK\Incorrect directory back link\Directory not found)
<MSG_TEXT>(Facility) VERIFY, Verify Utility
<MSG_TEXT>(Severity) Fatal
<MSG_TEXT>
The Verify Utility could not process your command. Please check the
syntax of your statement.
<MSG_TEXT>(User Action)
Check the syntax of the command, especially the directory specification,
and reenter the command.
<MSG>(UAF-E-NAOFIL\unable to open file SYSUAF.DAT\~RMS-E-FNF\file not found)
<MSG_TEXT>
The AUTHORIZE Utility could not locate the file SYSUAF.DAT.
<MSG_TEXT>(User Action)
Check that your process is currently set to the system default directory,
SYS$SYSTEM, and then reissue the command.
<ENDMESSAGE_SECTION>
```

This example produces the following output:

```
BCK-F-BADLNK, Incorrect directory back link
Directory not found
```

Facility: VERIFY, Verify Utility

Fatal: The Verify Utility could not process your command. Please check the syntax of your statement.

User Action: Check the syntax of the command, especially the directory specification, and reenter the command.

```
UAF Utility could not locate the file SYSUAF.DAT,
```

User Action: Check that your process is currently set to the system default directory, SYS\$SYSTEM, and then reissue the command.

<MESSAGE_TYPE>

Establishes the format for error messages in the context of the <MESSAGE_SECTION> tag.

SYNTAX

<MESSAGE_TYPE>({ NOIDENT
 NUMIDENT
 TEXTIDENT })

ARGUMENTS

NOIDENT

This keyword argument indicates that only message text will be used as arguments to the <MSG> or <MSGS> tag. This is the default.

NUMIDENT

This keyword argument indicates that the message has two parts, a numeric identification string and a line of message text. These two arguments are then required by the <MSG> or <MSGS> tag.

TEXTIDENT

This keyword argument indicates that the message has two parts, a text identification string and a line of message text. These two arguments are then required by the <MSG> or <MSGS> tag.

related tags

- <MESSAGE_SECTION>
- <MSG>
- <MSGS>
- <MSG_TEXT>

restrictions

Valid only in the context of a <MESSAGE_SECTION> tag.

DESCRIPTION

The <MESSAGE_TYPE> tag establishes the format for error messages in the context of the <MESSAGE_SECTION> tag. The format determines the number of arguments given to the <MSG> and <MSGS> tags.

For a complete description of the message section tags, refer to the description of <MESSAGE_SECTION> in this section.

EXAMPLE

See the example in the <MESSAGE_SECTION> tag description.

SOFTWARE Doctype Tag Reference

<MSG>

<MSG>

Formats the text of a message in a series of error message descriptions.

SYNTAX

<MSG> (*message id*[\ *message text-1*[\ *message text-2*]])

ARGUMENTS

message id

Specifies a unique message identification string. This argument can be either a text string or a numeric string. You can only specify this keyword argument if you have specified either the NUMIDENT or the TEXTIDENT keyword argument to the <MESSAGE_TYPE> tag.

message text-1

Specifies the text of the message.

message text-2

This is an optional argument. It specifies the second line of the text of the message.

related tags

- <MESSAGE_SECTION>
 - <MESSAGE_TYPE>
 - <MSG_TEXT>
 - <MSGS>
-

restrictions

Valid only in the context of a <MESSAGE_SECTION> tag.

DESCRIPTION

The <MSG> tag formats the text of a message in a series of error message descriptions.

The *message id* argument corresponds to the %FAC-S-IDENT portion of a system or application message in the VMS operating system programming environment.

If you are using the NUMIDENT format, the *message id* argument must be a numeric message identification string of not more than 6 picas (approximately 10 characters) in length.

If you are using the TEXTIDENT format, the *message id* argument has a comma (,) placed between it and the following *message text* argument.

For a complete description of the message section tags, refer to the description of the <MESSAGE_SECTION> tag in this section.

EXAMPLE

See the example in the <MESSAGE_SECTION> tag description.

SOFTWARE Doctype Tag Reference

<MSG>

<MSG>

Formats the text of one or more messages in a series of error message descriptions.

SYNTAX

<MSG>(*message text-1*[\ *message text-2* . . .
[\ *message text-9*]])

<MSG>(*message id-1* \ *message text-1*
[\ *message id-2* \ *message text-2*]
[\ *message id-3* \ *message text-3*]
[\ *message id-4* \ *message text-4*])

ARGUMENTS

message text-n

Specifies up to nine lines of text for the message.

message id-n

Specifies up to four unique message identification strings. This argument can be either a text string or a numeric string. You can only specify this argument if you have specified either the NUMIDENT or the TEXTIDENT keyword arguments to the <MESSAGE_TYPE> tag.

related tags

- <MESSAGE_SECTION>
- <MESSAGE_TYPE>
- <MSG>
- <MSG_TEXT>

restrictions

Valid only in the context of a <MESSAGE_SECTION> tag.

DESCRIPTION

The <MSG> tag formats the text of one or more messages in a series of error message descriptions. If you are using the NOIDENT keyword argument to the <MESSAGE_TYPE> tag, or if you do not specify the <MESSAGE_TYPE> tag, you must use the first syntax listed.

If you are using either the TEXTIDENT or NUMIDENT keywords as arguments to the <MESSAGE_TYPE> tag, you must use the second syntax listed. Note that if you use this second syntax, you must specify the *message id* and *message text* arguments as pairs.

The *message id* argument corresponds to the %FAC-S-IDENT portion of a system or application message in the VMS operating system programming environment.

SOFTWARE Doctype Tag Reference

<MSG>

If you are using the NUMIDENT format, the *message id* argument must be a numeric message identification string of not more than 6 picas (approximately 10 characters).

If you are using the TEXTIDENT format, the *message id* argument has a comma (,) placed between it and the following *message text* argument.

For a complete description of the message section tags, refer to the description of <MESSAGE_SECTION> in this section.

EXAMPLE

See the example in the <MESSAGE_SECTION> tag description.

SOFTWARE Doctype Tag Reference

<MSG_ACTION>

<MSG_ACTION>

Labels the text explanation of what action is to be taken in response to an error message from a system or application.

SYNTAX

<MSG_ACTION>

related tags

- <MESSAGE_SECTION>
 - <MESSAGE_TYPE>
 - <MSG>
 - <MSGS>
 - <MSG_FACILITY>
 - <MSG_SEVERITY>
 - <MSG_TEXT>
-

restrictions

Valid only in the context of the <MESSAGE_SECTION> tag.

DESCRIPTION

The <MSG_ACTION> tag labels the text explanation of what action is to be taken in response to an error message from a system or application.

EXAMPLES

See the example in the discussion of the <MESSAGE_SECTION> tag.

<MSG_FACILITY>

Labels up to four message sources in a series of system or application error messages.

SYNTAX <MSG_FACILITY>(*facility* \ *facility* \ *facility* \ *facility*)

ARGUMENTS *facility*
The name of the software component reporting the message. If more than one facility reports the same message, you can specify up to four facility names as arguments to the <MSG_FACILITY> tag.

related tags

- <MESSAGE_SECTION>
- <MESSAGE_TYPE>
- <MSG>
- <MSGS>
- <MSG_ACTION>
- <MSG_SEVERITY>

restrictions Valid only in the context of the <MESSAGE_SECTION> tag.

DESCRIPTION The <MSG_FACILITY> tag labels up to four message sources in a series of system or application error messages.

EXAMPLES See the example in the <MESSAGE_SECTION> tag description.

SOFTWARE Doctype Tag Reference

<MSG_SEVERITY>

<MSG_SEVERITY>

Labels the severity of a message in a series of system or application error messages.

SYNTAX

<MSG_SEVERITY>

related tags

- <MESSAGE_SECTION>
 - <MESSAGE_TYPE>
 - <MSG>
 - <MSGS>
 - <MSG_ACTION>
 - <MSG_FACILITY>
 - <MSG_TEXT>
-

restrictions

Valid only in the context of the <MESSAGE_SECTION> tag.

DESCRIPTION

The <MSG_SEVERITY> tag labels the severity of a message in a series of system or application error messages.

EXAMPLES

See the example in the <MESSAGE_SECTION> tag description.

<MSG_TEXT>

Labels text that describes a message in a <MESSAGE_SECTION> tag section.

SYNTAX

<MSG_TEXT>[(*alternate heading*)]

ARGUMENTS

alternate heading

This is an optional heading. It specifies the label for the text of the message description. This text automatically has a colon (:) appended to the end of it. If you do not specify this argument, the default heading is Explanation:.

restrictions

Valid only in the context of the <MESSAGE_SECTION> tag.

required terminator

<ENDMESSAGE_SECTION, MSG, OR MSGS>

The text labeled by the <MSG_TEXT> tag is terminated either by the next <MSG> or <MSGS> tag, or by the <ENDMESSAGE_SECTION> tag.

DESCRIPTION

The <MSG_TEXT> tag labels a message description in an error message section. The text of this description begins after the <MSG_TEXT> tag and continues until the next message section tag is encountered.

To use this tag to label various portions of the message explanation, you specify this tag several times using various alternate headings.

For example, first you could have a brief explanation of the message under the default heading Explanation:. Then you could specify the <MSG_TEXT> tag again, with the alternate heading of User Action. This use of the <MSG_TEXT> tag labels the tasks the user should perform to correct the condition that caused the error message. Note that when specifying alternate headings, a colon (:) is supplied at the end of the heading.

For a complete description of all the message section tags, refer to the description of the <MESSAGE_SECTION> tag in this section.

EXAMPLE

See the example in the <MESSAGE_SECTION> tag description.

SOFTWARE Doctype Tag Reference

<OVERVIEW>

<OVERVIEW>

Provides a summary description of a reference element.

SYNTAX

<OVERVIEW>

related tags

- <DESCRIPTION>
-

restrictions

Valid only in the context of a SOFTWARE reference template.

required terminator

<ENDOVERRIDE>

DESCRIPTION

The <OVERVIEW> tag provides a summary description of a reference element. Use the <DESCRIPTION> tag to create a separate subsection that contains more detailed information to the user concerning the reference element. See the description of the <DESCRIPTION> tag for more information.

You do not need to use a <P> tag immediately after the <OVERVIEW> tag. The <OVERVIEW> tag generates the initial open line; you need only type the first paragraph of text.

You can use the <HELP_ONLY> and <ENDHELP_ONLY> tags in the overview to provide text more appropriate to the context of the VMS HELP facility.

EXAMPLE

See the example in the <DESCRIPTION> tag description.

<PARAMDEF>

Begins the text that defines an item in a parameter definition list.

SYNTAX <PARAMDEF>

ARGUMENTS *None.*

related tags • <PARAMDEFLIST>
 • <PARAMITEM>

restrictions Valid only in the context of the <PARAMDEFLIST> tag.

DESCRIPTION The <PARAMDEF> tag begins the text that defines an item in a parameter definition list. This text describes the item listed by the previous <PARAMITEM> tag. The text begun by the <PARAMDEF> tag is terminated by the next <PARAMITEM> or <ENDPARAMDEFLIST> tag.

EXAMPLE See the example in the <PARAMDEFLIST> tag description.

SOFTWARE Doctype Tag Reference

<PARAMDEFLIST>

<PARAMDEFLIST>

Begins a definition list of parameters or arguments.

SYNTAX

<PARAMDEFLIST>[({ *alternate heading* })]
NOHEAD
NONE]]

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading. The default heading for the <PARAMDEFLIST> tag can vary. See the DESCRIPTION section for more information on default parameter definition list headings.

NOHEAD

This is an optional keyword argument. It suppresses the output of the default heading for the <PARAMDEFLIST> tag.

NONE

This is an optional keyword argument. It causes the text *None.* to be output beneath the heading for the parameter definition list to indicate that no parameters are available. Note that if you use the *NONE* keyword, you should not use the <ENDPARAMDEFLIST> tag.

related tags

- <ARGDEFLIST>
- <PARAMDEF>
- <PARAMITEM>
- <QUALDEFLIST>
- <SET_TEMPLATE_HEADING>
- The global <DEFINITION_LIST> tag

required terminator

<ENDPARAMDEFLIST>

Required unless you specify the *NONE* keyword as an argument to the <PARAMDEFLIST> tag.

DESCRIPTION

The <PARAMDEFLIST> tag begins a definition list of parameters or arguments. This tag is available both inside and outside the context of the reference templates.

The <PARAMDEFLIST> tag is similar in format and syntax to the global <DEFINITION_LIST> tag. See *VAX DOCUMENT Using Global Tags* for more information on the <DEFINITION_LIST> tag. The <PARAMDEFLIST> tag enables two tags to create a parameter definition list. The <PARAMITEM>

SOFTWARE Doctype Tag Reference

<PARAMDEFLIST>

tag labels the list item being defined, and the <PARAMDEF> tag begins the definition of the list item.

A default heading is provided when you use the <PARAMDEFLIST> tag in a reference template; no default heading is provided when you use the <PARAMDEFLIST> tag outside a reference template.

Create your own heading for an individual parameter definition list by specifying that heading as the *alternate heading* argument. A heading specified in this way overrides any existing default headings.

Use the <SET_TEMPLATE_HEADING> tag to alter the default headings used by all subsequent <PARAMDEFLIST> tags. See the description of the <SET_TEMPLATE_HEADING> tag for more information.

The following informal table lists the default headings for the <PARAMDEFLIST> by their context.

Context	Default Heading
Command Template	Parameters
Routine Template	Arguments
Tag Template	Arguments
Statement Template	No default heading
Outside a Template	No default heading

EXAMPLES

The following examples show variations on the use of the <PARAMDEFLIST> tag.

This chapter uses the Tag template for its reference descriptions. In this template, the heading for a parameter definition list is defined as Arguments. You can see a sample of this heading just after the format section at the start of this tag description.

The following example uses the NOHEAD argument to the <PARAMDEFLIST> tag to suppress the output of a heading in a template. If this example were coded outside a template, there would be no default heading for the parameter definition list, and so there would be no need to use the NOHEAD argument to suppress a heading.

```
1 The system maintains logical names and their associated equivalence strings in
two types of tables:
<PARAMDEFLIST> (NOHEAD)
<PARAMITEM> (process-private)
<PARAMDEF>These tables contain logical names that are available only
to your process.
<PARAMITEM> (shareable)
<PARAMDEF>These tables contain logical names that are available to other
processes on the system.
<ENDPARAMDEFLIST>
```

This example produces the following output:

The system maintains logical names and their associated equivalence strings in two types of tables:

SOFTWARE Doctype Tag Reference

<PARAMDEFLIST>

process-private

These tables contain logical names that are available only to your process.

shareable

These tables contain logical names that are available to other processes on the system.

The following example shows how to use the global <ALIGN_AFTER> tag for additional formatting flexibility in the parameter definition list. Note that this <PARAMDEFLIST> tag is used in the Command template and so has the default heading Parameters.

```
2 <COMMAND_SECTION>
.
.
.
<PARAMDEFLIST>
<PARAMITEM>(STATUS:arg\
<ALIGN_AFTER>(STATUS:)COMMAND\
<ALIGN_AFTER>(STATUS:)TASK)
<PARAMDEF>Specifies whether status information is to be returned
from the RUN command.
<ENDPARAMDEFLIST>
.
<ENDCOMMAND_SECTION>
```

This example produces the following output:

PARAMETERS **STATUS:arg**
 COMMAND
 TASK

Specifies whether status information is to be returned from the RUN command.

<PARAMITEM>

Labels one to seven items to be defined in a parameter definition list.

SYNTAX <PARAMITEM>(item-1[\ item-2 . . . [\ item-7]])

ARGUMENTS *item-n*
Specifies the item in the parameter list to be defined. This tag accepts a minimum of one *item-n* argument and a maximum of seven *item-n* arguments. When you specify more than one argument, each subsequent optional *item-n* argument after the initial argument formats flush left under the first argument.

related tags

- <PARAMDEF>
- <PARAMDEFLIST>

restrictions Valid only in the context of the <PARAMDEFLIST> tag.

required terminator <ENDPARAMDEF>

DESCRIPTION The <PARAMITEM> tag labels one to seven items to be defined in a parameter definition list. You can specify one to seven arguments as items requiring a single definition in the parameter definition list. If you specify more than one *item-n* argument, the *item-n* arguments are stacked from top to bottom in the order in which they were specified.

If you need to format the *item-n* arguments differently than the default flush left formatting, use the global <ALIGN_AFTER> tag. A sample use of this tag is illustrated in the following example. See *VAX DOCUMENT Using Global Tags* for more information on the <ALIGN_AFTER> tag.

EXAMPLE The following example shows how to use the global <ALIGN_AFTER> tag in the context of a parameter definition list for special formatting purposes. Note how it is used outside the <PARAMITEM> tag.

```
<PARAMDEFLIST> (NOHEAD)
<PARAMITEM> (STATUS:arg\
<ALIGN_AFTER> (STATUS:) COMMAND\
<ALIGN_AFTER> (STATUS:) TASK)
<PARAMDEF> Specifies whether status information is to be returned
from the RUN command.
<ENDPARAMDEFLIST>
```

This example produces the following output:

SOFTWARE Doctype Tag Reference
<PARAMITEM>

STATUS:arg
COMMAND
TASK

Specifies whether status information is to be returned from the RUN command.

SOFTWARE Doctype Tag Reference

<PROMPT>

The following example shows how to specify the prompts for a SET PASSWORD command. In this example, the *prompt width* argument extends the width of the prompt text to 7 picas.

```
2 <PROMPTS>
  <PROMPT>(Old password:\old password\7)
  <PROMPT>(New password:\new password\7)
  <PROMPT>(Verification:\new password\7)
  <ENDPROMPTS>
```

This example produces the following output:

prompts

```
Old password:  old password
New password:  new password
Verification:  new password
```

<PROMPTS>

Begins a summary of interactive prompts.

SYNTAX

<PROMPTS>[(*alternate heading* [\ *NONE*])]

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading for this use of the <PROMPTS> tag. The default heading provided by VAX DOCUMENT is Prompts. If you want to use a different heading, specify it here. Also see the <SET_TEMPLATE_HEADING> tag description for information on modifying the default headings for all the <PROMPTS> tags.

NONE

This is an optional keyword argument. It indicates that there are no prompts for this command. If you use the NONE keyword, do not use the <ENDPROMPTS> tag to end the <PROMPTS> list.

related tags

- <COMMAND_SECTION>
- <PROMPT>
- <SET_TEMPLATE_HEADING>

required terminator

<ENDPROMPTS>

DESCRIPTION

The <PROMPTS> tag begins a summary of interactive prompts.

EXAMPLES

The following example illustrates a command with a single prompt.

```
❏ <PROMPTS>(Prompt)
   <PROMPT>(File:\filespec)
   <ENDPROMPTS>
```

This example produces the following output:

prompt

File: filespec

The following example illustrates a command with no prompts. Note that the terminator, <ENDPROMPTS>, is not specified.

SOFTWARE Doctype Tag Reference

<PROMPTS>

2 <PROMPTS> (NONE)

This example produces the following output:

prompts

None.

<QPAIR>

Labels a qualifier pair in a qualifier format list.

SYNTAX **<QPAIR>** (*qualifier name \ default qualifier name*)

ARGUMENTS

qualifier name

The command qualifier to be listed. A common convention indicates the negative form of the qualifier by placing brackets around the negative prefix, as in [NO]CHECK.

default qualifier name

The default value of the qualifier.

related tags

- <QUAL_LIST>
- <QUAL_LIST_HEADS>

restrictions

Valid only in the context of a <QUAL_LIST> tag.

DESCRIPTION The <QPAIR> tag labels a qualifier pair in a qualifier format list.

EXAMPLE See the example in the <QUAL_LIST> tag description.

SOFTWARE Doctype Tag Reference

<QUALDEF>

<QUALDEF>

Begins the text that defines an item in a qualifier definition list.

SYNTAX <QUALDEF>

ARGUMENTS *None.*

related tags

- <QUALDEFLIST>
- <QUALITEM>

restrictions Valid only in the context of a <QUALDEFLIST> tag.

DESCRIPTION The <QUALDEF> tag begins the text that defines an item in a qualifier definition list. This text describes the item listed by the previous <QUALITEM> tag. The text begun by the <QUALDEF> tag is terminated by the next <QUALITEM> or <ENDQUALDEFLIST> tag.

EXAMPLE See the example in the <QUALDEFLIST> tag description.

<QUALDEFLIST>

Begins a definition list describing command qualifiers.

SYNTAX

<QUALDEFLIST>[({ *alternate heading* })]
 { *NOHEAD* }
 { *NONE* }

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading. The default heading provided by VAX DOCUMENT for the <QUALDEFLIST> tag can vary. See the DESCRIPTION section for more information on default qualifier definition list headings.

NOHEAD

This is an optional keyword argument. It suppresses the output of the default heading for the <QUALDEFLIST> tag.

NONE

This is an optional keyword argument. It causes the text None to be output beneath the heading for the qualifier definition list to indicate that no qualifiers are available. Note that when you use the NONE keyword, you do not use the <ENDQUALDEFLIST> tag.

related tags

- <ARGDEFLIST>
- <PARAMDEFLIST>
- <QUALDEF>
- <QUALITEM>
- <SET_TEMPLATE_ARGITEM>
- <SET_TEMPLATE_HEADING>
- The global <DEFINITION_LIST> tag

required terminator

<ENDQUALDEFLIST> Required unless you specify the NONE keyword as an argument to the <QUALDEFLIST> tag.

DESCRIPTION

The <QUALDEFLIST> tag begins a definition list describing command qualifiers. It is similar in format and syntax to the global <DEFINITION_LIST> tag. See *VAX DOCUMENT Using Global Tags* for more information.

The <QUALDEFLIST> tag enables two tags to create a qualifier definition list. The <QUALITEM> tag labels the list item being defined, and the <QUALDEF> tag begins the definition of the list item. These tags are functionally the same as the <DEFLIST_ITEM> and <DEFLIST_DEF> tags enabled by the

SOFTWARE Doctype Tag Reference

<QUALDEFLIST>

global <DEFINITION_LIST> tag. A default heading is provided when the <QUALDEFLIST> tag is used in a reference template; no default heading is provided when the <QUALDEFLIST> tag is used outside a reference template.

Create your own heading for an individual qualifier definition list by specifying that heading as the *alternate heading* argument. A heading specified in this way overrides any existing default headings.

Use the <SET_TEMPLATE_HEADING> tag to alter the default headings used by all subsequent <QUALDEFLIST> tags. See the description of the <SET_TEMPLATE_HEADING> tag for more information.

The following informal table lists the default headings for the <QUALDEFLIST> by their context.

Context	Default Heading
Command Template	Qualifiers
Tag Template	Qualifiers
Routine Template	No default heading
Statement Template	No default heading
Outside a Template	No default heading

EXAMPLE

The following example shows a qualifier definition list in the context of the Command template.

```
<COMMAND_SECTION>
.
.
<QUALDEFLIST>
<QUALITEM>(/LOG\NOLOG (D))
<QUALDEF>Specifies whether or not output logging is used.
The default is /NOLOG.
<QUALITEM>(/ECHO\NOECHO (D))
<QUALDEF>Specifies whether or not input echoing is used.
The default is /NOECHO.
<ENDQUALDEFLIST>
.
.
<ENDCOMMAND_SECTION>
```

This example produces the following output:

QUALIFIERS

/LOG

/NOLOG (D)

Specifies whether or not output logging is used. The default is /NOLOG.

/ECHO

/NOECHO (D)

Specifies whether or not input echoing is used. The default is /NOECHO.

<QUALITEM>

Labels one to seven items to be defined in a qualifier definition list.

SYNTAX

<QUALITEM>(item-1[\ item-2 . . . [\ item-7]])

ARGUMENTS

item-n

Specifies the item in the qualifier list to be defined. This tag accepts a minimum of one *item-n* argument and a maximum of seven *item-n* arguments. When you specify more than one argument, each subsequent *item-n* argument after the initial argument formats flush left under the first optional argument.

related tags

- <QUALDEF>
- <QUALDEFLIST>

restrictions

Valid only in the context of a <QUALDEFLIST> tag.

required terminator

<QUALDEF>

DESCRIPTION

The <QUALITEM> tag labels one to seven items to be defined in a qualifier definition list. You can specify one to seven arguments as items requiring a single definition in the qualifier definition list. If you specify more than one *item-n* argument, the *item-n* arguments are stacked from top to bottom in the order in which they were specified.

You may find it convenient to use the *item-n* arguments to the <QUALDEFLIST> tag in pairs. The first item in the pair could be the positive form of the qualifier, and the second item could be the negative form of the qualifier (for example, /LOG and /NOLOG). Use the global <ALIGN_AFTER> to format the *item-n* arguments differently than the default flush left formatting.

A sample use of this tag is illustrated in the following example. See *VAX DOCUMENT Using Global Tags* for more information on the <ALIGN_AFTER> tag.

SOFTWARE Doctype Tag Reference

<QUALITEM>

EXAMPLE

The following examples show several uses of the <QUALITEM> tag. The first two uses of the <QUALITEM> tag show how positive and negative forms of a qualifier may be grouped together. The third use of the <QUALITEM> tag shows how you use the global <ALIGN_AFTER> tag for special formatting.

```
<QUALDEFLIST>
<QUALITEM>(/LOG\NOLOG (D))
<QUALDEF>Specifies whether or not output logging should be used.
The default is /NOLOG.
<QUALITEM>(/ECHO\NOECHO (D))
<QUALDEF>Specifies whether or not input echoing should be used.
The default is /NOECHO.
<QUALITEM>(/DEVICE=device type\
<ALIGN_AFTER>(DEVICE=)VT100\
<ALIGN_AFTER>(DEVICE=)VT220)
<QUALDEF>Specifies the type of device to be used.
<ENDQUALDEFLIST>
```

This example produces the following output:

QUALIFIERS

/LOG

/NOLOG (D)

Specifies whether or not output logging should be used. The default is /NOLOG.

/ECHO

/NOECHO (D)

Specifies whether or not input echoing should be used. The default is /NOECHO.

/DEVICE=device type

VT100

VT220

Specifies the type of device to be used.

<QUAL_LIST>

Begins a qualifier summary list.

SYNTAX

<QUAL_LIST>[[{ *alternate heading*
 NONE
 SPECIAL }
 { \ *alternate heading*
 \ *column width* } \ *WIDE*)]

ARGUMENTS

alternate heading

This is an optional argument. It causes this text to be used instead of the default heading in the first column, Command Qualifiers. This default heading can be modified or suppressed by using the <QUAL_LIST_DEFAULT_HEADS> tag.

NONE

This is an optional keyword argument. It indicates that there are no qualifiers or defaults and causes the text None to appear under the default headings for the first and second columns (Command Qualifiers and Defaults).

SPECIAL

This is an optional keyword argument. It labels an unusual case and causes VAX DOCUMENT to expect a value in the second argument that will set the width of the first qualifier column.

alternate heading

This is an optional argument. It causes this text to be used in column two in place of the default heading Defaults. This default heading can be modified or suppressed by using the <QUAL_LIST_DEFAULT_HEADS> tag.

column width

This is an optional argument. It provides VAX DOCUMENT with the size in picas of the desired width of the first column (there are 6 picas to an inch). This value must be a nonzero positive integer and can be used only when the first argument is SPECIAL.

WIDE

This is an optional keyword argument. It causes the margins to be adjusted to accommodate a wide list.

SOFTWARE Doctype Tag Reference

<QUAL_LIST>

WIDE

This is an optional keyword argument. It causes the margins to be adjusted to accommodate a wide list. Use this third argument only in the two following cases:

- When you have used the first two arguments to specify headings for the first two columns.
- When you have used the first two arguments to specify a special list and the width of the first column.

related tags

- <QPAIR>
- <QUAL_LIST_DEFAULT_HEADS>
- <QUAL_LIST_HEADS>

restrictions

If you use the SPECIAL argument, you must use the <QUAL_LIST_HEADS> tag to specify headings for the qualifier list.

required terminator

<ENDQUAL_LIST>

DESCRIPTION

The <QUAL_LIST> tag begins a qualifier summary list. A qualifier summary list provides a short table listing the qualifiers that are applicable for a system command. It is an optional part of the command template, but you can use it in any context in a SOFTWARE doctype document. In the context of the command template, it provides a brief listing of qualifiers for quick lookup.

The arguments you specify to the <QUAL_LIST> tag provide you with formatting flexibility, and the choice of overriding the default headings that are output.

- When you specify the tag with no arguments, the column widths are set using the doctype design's default settings, and the default headings Command Qualifiers and Defaults are output.
- When you specify text in the arguments to the <QUAL_LIST> tag, the default headings are replaced:

```
<QUAL_LIST>(Subsystem Qualifier\Comment)
```

This changes the default heading for this qualifier summary list only.

- You override the default headings for all qualifier summary lists in your document by using the <QUAL_LIST_DEFAULT_HEADS> tag, as follows:

```
<QUAL_LIST_DEFAULT_HEADS>(Subsystem Qualifier\Comment)
```

To suppress either heading, enter the alternate heading text as a null argument:

```
<QUAL_LIST_DEFAULT_HEADS>(\)
```

SOFTWARE Doctype Tag Reference

<QUAL_LIST>

- If items in the second column of your list result in output that does not fit horizontally in the SOFTWARE design you are using, specify the keyword argument WIDE in any of the following positions:

```
<QUAL_LIST> (WIDE)
```

This example uses the default headings, and shifts both columns of the list to the left, as far left as the doctype design allows.

```
<QUAL_LIST> (heading text\WIDE)
```

This example modifies the default heading for the first column, uses the default heading for the second column, and shifts both columns of the list to the left, as far left as the document design allows.

```
<QUAL_LIST> (heading text\second heading text\WIDE)
```

This example modifies the default headings for both the first and second columns, and shifts both columns of the list to the left, as far left as the doctype design allows.

```
<QUAL_LIST> (SPECIAL\18)  
<QUAL_LIST_HEADS> (Qualifiers)
```

If the default column width of the first column of the qualifier summary list is too narrow (for example, when qualifier names are long or include lengthy argument specifications), widen that column by specifying the SPECIAL keyword as the first argument to the <QUAL_LIST> tag.

The <QUAL_LIST> tag will then accept the *column width* argument as a second argument that specifies the width of the first column of the list in picas (there are 6 picas to an inch). If you use the <QUAL_LIST> tag in this manner, you must explicitly enter the headings for the qualifier summary list using the <QUAL_LIST_HEADS> tag.

EXAMPLES

The following example shows how to use the <QUAL_LIST> tag to create a qualifier summary list.

```
1 <QUAL_LIST>  
  <QPAIR> (/BOOK\None.)  
  <QPAIR> (/ [NO]CHECK\CHECK)  
  <QPAIR> (/ [NO]FAMILY=keyword\NOFAMILY)  
  <QPAIR> (/OUTPUT=file spec\OUTPUT=input file name)  
  <QPAIR> (/PROFILE=file spec\None.)  
  <QPAIR> (/TYPE=keyword\See text.)  
<ENDQUAL_LIST>
```

This example produces the following output:

Command Qualifiers	Defaults
/BOOK	None.
/[NO]CHECK	/CHECK
/[NO]FAMILY=keyword	/NOFAMILY
/OUTPUT=file spec	/OUTPUT=input file name
/PROFILE=file spec	None.
/TYPE=keyword	See text.

SOFTWARE Doctype Tag Reference

<QUAL_LIST>

The following example shows how to control the headings of the two columns of output by specifying the headings you want in arguments one and two. Compare the results here with the example in the discussion of the <QUAL_LIST_HEADS> tag.

```
2 <QUAL_LIST>(Input Save Set Qualifiers\Default)
  <QPAIR>(/[NO]REWIND\REWIND)
  <QPAIR>(/SAVE_SET\None.)
  <QPAIR>(/SELECT=(file spec[,...])\None.)
  <ENDQUAL_LIST>
```

This example produces the following output:

Input Save Set Qualifiers	Default
<i>/[NO]REWIND</i>	<i>/REWIND</i>
<i>/SAVE_SET</i>	<i>None.</i>
<i>/SELECT=(file spec[, ...])</i>	<i>None.</i>

SOFTWARE Doctype Tag Reference

<QUAL_LIST_DEFAULT_HEADS>

Command Qualifiers	Defaults
<i>/BOOK</i>	<i>None.</i>
<i>/[NO]CHECK</i>	<i>/CHECK</i>
<i>/[NO]FAMILY=keyword</i>	<i>/NOFAMILY</i>
<i>/OUTPUT=file spec</i>	<i>/OUTPUT=input file name</i>
<i>/PROFILE=file spec</i>	<i>None.</i>
<i>/TYPE=keyword</i>	<i>See text.</i>

<QUAL_LIST_HEADS>

Labels the headings for one or both of the columns in a qualifier format list when you use the SPECIAL argument qualifier to the <QUAL_LIST> tag in unusual cases for formatting control.

SYNTAX <QUAL_LIST_HEADS>(heading-1 \ heading-2)

ARGUMENTS *heading-1*

Specifies the heading text for the left column.

heading-2

Specifies the heading text for the right column. If you do not specify this heading, you will obtain the default heading, Defaults.

related tags

- <QPAIR>
- <QUAL_LIST>

restrictions

Valid only in the context of a <QUAL_LIST> tag.

DESCRIPTION

The <QUAL_LIST_HEADS> tag labels the headings for one or both of the columns in a qualifier format list when you use the SPECIAL argument qualifier to the <QUAL_LIST> tag in unusual cases for formatting control.

EXAMPLE

The following example shows how to use the <QUAL_LIST_HEADS> tag to create qualifier summary list headings.

```
<P>The following is a partial list of the qualifiers you may use  
with the BACKUP command:  
<QUAL_LIST>(SPECIAL\18)  
<QUAL_LIST_HEADS>(Output File Qualifiers\Qualifier Defaults)  
<QPAIR>( /OWNER_UIC[=option] \ /OWNER_UIC=DEFAULT)  
<QPAIR>( /REPLACE\None.)  
<ENDQUAL_LIST>
```

This example produces the following output:

The following is a partial list of the qualifiers you may use with the BACKUP command:

Output File Qualifiers
/OWNER_UIC[=option]
/REPLACE

Qualifier Defaults
/OWNER_UIC=DEFAULT
None.

SOFTWARE Doctype Tag Reference

<RELATED_ITEM>

<RELATED_ITEM>

Provides a text description of a tag or set of tags that may be related to the tag being described.

SYNTAX <RELATED_ITEM>

ARGUMENTS *None.*

related tags

- <RELATED_TAG>
- <RELATED_TAGS>

restrictions

Valid only in the context of a <RELATED_TAGS> tag section in the Tag template.

DESCRIPTION

The <RELATED_ITEM> tag provides a text description of a tag or set of tags that may be related to the tag being described. This text begins right after the <RELATED_ITEM> tag and is terminated by the next related tag section tag.

EXAMPLE

The following example shows how to enter text describing information related to the use of a tag.

```
<RELATED_TAGS>  
<RELATED_ITEM> Use the <TAG>(INTRO) and <TAG>(ENDINTRO) tags  
to label introductory material in your book.  
<ENDRELATED_TAGS>
```

This example produces the following output:

related tags

- Use the <INTRO> and <ENDINTRO> tags to label introductory material in your book.

<RELATED_TAG>

Specifies a single tag that is related to the current tag.

SYNTAX **<RELATED_TAG>**(*tag name*)

ARGUMENTS ***tag name***

Specifies the name of a VAX DOCUMENT tag. Do not place angle brackets around the tag name; the <RELATED_TAG> tag supplies them by default.

related tags

- <RELATED_ITEM>
- <RELATED_TAGS>

restrictions

Valid only in the context of the <RELATED_TAGS> tag in the Tag template.

DESCRIPTION

The <RELATED_TAG> tag specifies a single tag that is related to the current tag. The <RELATED_TAG> tag automatically places angle brackets around the *tag name* argument, so angle brackets should not be specified.

Use the <RELATED_ITEM> tag to list related tag information in another format.

EXAMPLE

The following example specifies the names of two related tags in the context of the <RELATED_TAGS> tag. Note how the names of the related tags are specified without the angle brackets.

```
<RELATED_TAGS>
<RELATED_TAG>(SET_TEMPLATE_TAG)
<RELATED_TAG>(TAG_SECTION)
<ENDRELATED_TAGS>
```

This example produces the following output:

related tags

- <SET_TEMPLATE_TAG>
- <TAG_SECTION>

SOFTWARE Doctype Tag Reference

<RELATED_TAGS>

<RELATED_TAGS>

Provides a summary of tags whose use is related to the tag being described.

SYNTAX **<RELATED_TAGS>[(NONE)]**

ARGUMENTS **NONE**

This is an optional keyword argument. It indicates that there are no tags whose use is related to the tag being described.

related tags

- <RELATED_ITEM>
 - <RELATED_TAG>
-

restrictions

Valid only in the context of the Tag template. If you specify NONE, do not specify the <ENDRELATED_TAGS> tag.

required terminator

<ENDRELATED_TAGS>

DESCRIPTION The <RELATED_TAGS> tag provides a summary of tags whose use is related to the tag being described.

EXAMPLES

The following example shows how to specify two related tags in a <RELATED_TAGS> tag section.

```
❏ <RELATED_TAGS>
   <RELATED_TAG>(SET_TEMPLATE_TAG)
   <RELATED_TAG>(TAG_SECTION)
   <ENDRELATED_TAGS>
```

This example produces the following output:

related tags

- <SET_TEMPLATE_TAG>
- <TAG_SECTION>

The following example shows how to use the NONE keyword to show that there are no related tags. Note that, in this case, the <ENDRELATED_TAGS> tag is omitted.

SOFTWARE Doctype Tag Reference

<RELATED_TAGS>

2 <RELATED_TAGS> (NONE)

This example produces the following output:

related tags

None.

SOFTWARE Doctype Tag Reference

<RESTRICTIONS>

<RESTRICTIONS>

Provides the restrictions on the use of a tag.

SYNTAX

<RESTRICTIONS>[({ *alternate heading*[\ *LIST*] }
NONE
LIST)]]

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading for this use of the <RESTRICTIONS> tag. The default heading provided by VAX DOCUMENT is Restrictions. See the reference description of the <SET_TEMPLATE_HEADING> tag for information on how to modify the default headings for the <RESTRICTIONS> tag.

NONE

This is an optional keyword argument. It indicates that there are no restrictions on the use of the tag. If you specify the NONE keyword, do not use the <ENDRESTRICTIONS> tag to end the <RESTRICTIONS> tag section.

LIST

This is an optional keyword argument. It indicates that a number of restrictions are to be listed. To list the restrictions, use the <RITEM> for each of the individual restriction items in the list.

related tags

- <RITEM>
- <TAG_SECTION>

restrictions

Valid only in the context of the Tag template. If you specify NONE, do not specify the <ENDRESTRICTIONS> tag.

required terminator

<ENDRESTRICTIONS>

DESCRIPTION

The <RESTRICTIONS> tag provides the restrictions on the use of a tag.

EXAMPLES

The following example shows how to set a simple paragraph of text for the restrictions section using the <RESTRICTIONS> tag.

1

```
<RESTRICTIONS>
Valid only in the context of the
<tag>(COMMAND_SECTION) tag.
<ENDRESTRICTIONS>
```

This example produces the following output:

restrictions

Valid only in the context of the <COMMAND_SECTION> tag.

The following example shows how to use the NONE keyword to indicate that there are no restrictions on the use of a tag.

2

```
<RESTRICTIONS> (NONE)
```

This example produces the following output:

restrictions

None.

The following example shows how to create a set of restrictions in a list.

3

```
<RESTRICTIONS>(LIST)
<RITEM>If you specify NONE, you must not specify <tag>(ENDPARAMDEFLIST).
<RITEM>Use of a default heading is restricted to the reference templates.
<ENDRESTRICTIONS>
```

This example produces the following output:

restrictions

- If you specify NONE, you must not specify <ENDPARAMDEFLIST>.
- Use of a default heading is restricted to the reference templates.

SOFTWARE Doctype Tag Reference

<RETTEXT>

<RETTEXT>

Provides general information about the attributes of the value returned by the routine.

SYNTAX

<RETTEXT>

related tags

- <RETURNS>

required terminator

<ENDRETTEXT>

restrictions

Valid only following the <RETURNS> tag in the Routine template.

DESCRIPTION

The <RETTEXT> tag provides general information about the attributes of the value returned by the routine. It places one or more paragraphs of information after some usage of the <RETURNS> tag. This information begins immediately after the <RETTEXT> tag and continues until the <ENDRETTEXT> tag is encountered.

You typically use this tag when the arguments given to the <RETURNS> tag are not sufficiently descriptive.

EXAMPLE

See the example in the <RETURNS> tag description.

<RETURNS>

Provides information about the value returned by a routine.

SYNTAX

<RETURNS> (*usage information*
 \ *data type* \ *access* \ *mechanism*
 [\ *optional info*])
<RETURNS> (**HEADONLY**)

ARGUMENTS***usage information***

Specifies a keyword indicating the category of data to which the return value belongs. These keywords are system dependent, and are specified by agreed-upon conventions.

data type

Indicates the data type of the return value, for example, longword, byte, G_floating, and so on.

access

Indicates the access applied to the return value, for example, read-only, write-only, and so on.

mechanism

Specifies the mechanism by which the return value is passed; for example, by descriptor, by reference, or by value.

optional info

This is an optional argument. It specifies additional information which may be appended to the *mechanism* argument output.

HEADONLY

This alternative keyword argument indicates that specific usage information is not relevant, and that text will be provided to describe the return attributes of the routine.

related tags

- <RETTEXT>
-

restrictions

Valid only in the context of the Routine template.

DESCRIPTION

The <RETURNS> tag provides information about the value returned by a routine. The returns section of the Routine reference template and its specialized argument list are used only by convention for those documenting callable routines.

SOFTWARE Doctype Tag Reference

<RETURNS>

EXAMPLES

The following two input examples show various uses of the <RETURNS> tag. Outputs from these coding examples appear after the last input example.

The following input example shows how to specify multiple arguments to the <RETURNS> tag.

1 `<RETURNS>(floating_point\
F_Floating, D_Floating, or G_Floating\write only\by value in R0)`

The following input example shows a use of the HEADONLY argument. The return value from a routine or set of related routines may be too complex to express using the conventional arguments.

2 `<RETURNS>(headonly)
<RETTEXT>The square roots of F_Floating, D_Floating, and G_Floating
input parameters are returned by immediate value in R0 and R1. The
square root of an H_Floating parameter is returned by reference
in the output parameter <VARIABLE>(sqrt).
<ENDRETTEXT>`

These input examples produce the following output:

RETURNS

VMS Usage: **floating_point**
type: **F_Floating, D_Floating, or G_Floating**
access: **write only**
mechanism: **by value in R0**

RETURNS

The square roots of F_Floating, D_Floating, and G_Floating input parameters are returned by immediate value in R0 and R1. The square root of an H_Floating parameter is returned by reference in the output parameter *sqrt*.

<RETURN_VALUE>

Labels a character string return value.

SYNTAX <RETURN_VALUE>[(*alternate heading*)]

ARGUMENTS *alternate heading*
This is an optional argument. It specifies a heading to override the current default text heading for this use of the <RETURN_VALUE> tag. The default heading provided by VAX DOCUMENT is Return Value. See the reference description of the <SET_TEMPLATE_HEADING> tag for information on how to modify the default headings for all <RETURN_VALUE> tags.

restrictions Valid only in the context of the Command template.

required terminator <ENDRETURN_VALUE>

DESCRIPTION The <RETURN_VALUE> tag labels a character string return value.

EXAMPLE The following example shows how to use the <RETURN_VALUE> tag.

```
<RETURN_VALUE>  
abc-def-ghi  
<ENDRETURN_VALUE>
```

This example produces the following output:

return value abc-def-ghi

SOFTWARE Doctype Tag Reference

<RITEM>

<RITEM>

Labels an item in a list of restrictions.

ARGUMENTS *None.*

SYNTAX **<RITEM>**

related tags • <RESTRICTIONS>

restrictions Valid only in the context of the <RESTRICTIONS> tag.

DESCRIPTION The <RITEM> tag labels an item in a list of restrictions.

EXAMPLE The following example shows how to create a list of restrictions. Note how you specify the <RESTRICTIONS> tag with the LIST keyword argument. Note how this enables the <RITEM> tag.

```
<RESTRICTIONS> (LIST)
<RITEM>You must not unplug this appliance while it is operating.
<RITEM>This appliance should not be immersed in water.
<ENDRESTRICTIONS>
```

This example produces the following output:

restrictions

- You must not unplug this appliance while it is operating.
- This appliance should not be immersed in water.

<ROUTINE>

Begins a new routine description.

SYNTAX **<ROUTINE>**(*routine name*[\ *info1*[\ *info2*]] \ *symbol name*)

ARGUMENTS

routine name

Specifies the name of the routine to be described.

info1

info2

These are optional arguments. They specify an optional text description of the routine's function.

symbol name

Specifies the name of the symbol used in all references to the routine.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <ROUTINE_SECTION>
- <SET_TEMPLATE_ROUTINE>

restrictions

Valid only in the context of the Routine template.

DESCRIPTION

The <ROUTINE> tag begins a new routine description. This description is for a single routine in the context of the <ROUTINE_SECTION> tag. This tag has the following default format:

- Each <ROUTINE> tag begins a new page of output.
- Each output page carries a single running title, which is the current routine name.
- If you use the optional *info* arguments with the <ROUTINE> tag, these arguments output after the *routine name* argument and are separated from the *routine name* argument by an em dash (—).

To replace the <ROUTINE> tag with a tag more specific to a certain task (for example, the <DCL_ROUTINE>) tag, or to change the default attributes of the <ROUTINE> tag, use the <SET_TEMPLATE_ROUTINE> tag. See the description of the <SET_TEMPLATE_ROUTINE> tag in this chapter for more information.

SOFTWARE Doctype Tag Reference

<ROUTINE>

EXAMPLES

In the following example, the <ROUTINE_SECTION> tag enables the tags for a routine description.

1 <ROUTINE_SECTION>
<ROUTINE>(\$OPEN)
<OVERVIEW>
.
.
.

The description of the routine \$OPEN has the following default attributes:

- The routine description starts a new page.
- If the routine carries for more than a page, the name \$OPEN is carried as a running top title on each page.

2 <ROUTINE>(\$CLOSE\Close a File)

When two arguments are specified to the <ROUTINE> tag, the routine name, \$CLOSE, appears at the beginning of the routine description. The text specified in the second argument, Close a File, follows the routine name at the top of the page, separated from the routine name by an em dash (—).

3 <ROUTINE>(SMG\$BEGIN_PASTEBOARD_UPDATE\\Begin Batching of Pasteboard Updates)

This example illustrates special coding required when a routine's name is extremely long and may require special formatting. If the second argument is null, the third argument is stacked under the first argument. No em dash is output:

```
SMG$BEGIN_PASTEBOARD_UPDATE  
Begin Batching of Pasteboard Updates
```

Use this form only if the output from the other formats appears wrong.

4 <ROUTINE>(OTSS\$COPY_R_DX\Copy a Source String \Passed by Reference to a Destination String)

This example illustrates special coding required when a routine's name and descriptive name creates unreasonable output using the default formatting attributes. If you specify three arguments, the first and second arguments are output on the first line, separated by an em dash. The third argument is stacked under the first argument (instead of wrapping) following the em dash:

```
OTSS$COPY_R_DX---Copy a Source String  
given by Reference to a Destination String
```

Use this form only if examination of your output indicates a formatting problem.

SOFTWARE Doctype Tag Reference

<ROUTINE_SECTION>

- <RETURNS>
- <RETTEXT>
- <ROUTINE>
- <RSDEFLIST>
- <RSITEM>
- <SET_TEMPLATE_HEADING>
- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_PARA>
- <SET_TEMPLATE_TABLE>

**required
terminator**

<ENDROUTINE_SECTION>

DESCRIPTION

The <ROUTINE_SECTION> tag begins a routine reference section, enables tags reserved for use in routine sections, and sets paging attributes. You can place more than one routine section in a single document. However, you must end each previous routine section before you begin the next one.

You can tailor the default format of the routine reference template to meet your own documentation requirements. Either you can alter the default attributes of the <ROUTINE_SECTION> tag by specifying one of the arguments to that tag, or you can use the <SET_TEMPLATE_ROUTINE> tag to alter the default attributes for the <ROUTINE> tag that begins each new routine description.

You can place a routine section in a chapter or an appendix, or following a part page (that is, in a document section begun with the <PART_PAGE> tag). You code a routine section in a chapter or an appendix in the same manner; command sections in parts are handled differently.

If your routine section follows a part page, and you include text between the part page and the routine section, specify the NEWPAGE keyword as the third argument to the <ROUTINE_SECTION> tag. This causes the routine section to begin on a new page. The following code fragment shows a routine section that begins on a new page:

```
<ROUTINE_SECTION>(\AB\NEWPAGE)
<HEAD1>(Routine Format\42_RoutineFormat)
```

When you use the <ROUTINE_SECTION> tag in a chapter or an appendix, and you want to place text after the routine section in that chapter or appendix, you must end the routine section with the <ENDROUTINE_SECTION> tag and place the text after that tag. By default, this text begins on a new page of output.

Specify the NONEWPAGE argument to the <ENDROUTINE_SECTION> tag if you do not want the text to begin on a new page of output. The following code fragment shows the end of a routine section that specifies that the subsequent text not be placed on a new page:

SOFTWARE Doctype Tag Reference

<ROUTINE_SECTION>

<ENDROUTINE_SECTION> (NONEWPAGE)

When the <ENDROUTINE_SECTION> tag is specified in the context of a chapter or appendix, it resets the default running titles to those in effect for the chapter or appendix, so the last page of the last routine description in the routine section may not carry the last routine's name as the running heading. Instead it may carry the running title used by the chapter or appendix.

EXAMPLES

The following example shows how to begin a routine section in a document part.

```
1 <PART>
  <PART_PAGE>
  <TITLE>(PART II\AAA Routine Descriptions)
  <ABSTRACT>This Part contains complete reference descriptions of
  the AAA routines.
  <ENDABSTRACT>
  <ENDPART_PAGE>(RENUMBER)
  <endtag_section>

  <ROUTINE_SECTION>(AAA Routines\AAA)
  <SET_TEMPLATE_ROUTINE>(AAA_ROUTINE\DOUBLERUNNINGHEADS)

  <AAA_ROUTINE>(AAA$CLOSE)

  <OVERVIEW>
  Closes the specified file.
  <ENDOVERRIDEVIEW>
  .
  .
  .
  <ENDROUTINE_SECTION>
```

The tags in the previous example perform the following functions:

- The global <PART> tag begins the part.
- The global <PART_PAGE> tag creates a part page.
- The global <TITLE> tag is used in the context of the <PART_PAGE> tag to create a title on the part page.
- The RENUMBER argument to the global <ENDPART_PAGE> tag specifies that the pages are renumbered beginning with the part page. This causes the first page of text following the part page to be numbered page 3 (page 1 is the unnumbered page the part page title is placed on, page 2 is the back of page 1, and page 3 is the first numbered page after the part page).
- The <ROUTINE_SECTION> tag begins the routine section and specifies the running title AAA Routines as the running title for the routine section.

The <ROUTINE_SECTION> tag also specifies that the prefix AAA should be used to construct numbers for pages and for formal figures, tables, and examples in the routine section (for example, AAA-11, AAA-32, Table AAA-1, Example AAA-2, and so on).

SOFTWARE Doctype Tag Reference

<ROUTINE_SECTION>

- The <SET_TEMPLATE_ROUTINE> tag specifies that all routine descriptions in this routine section are identified using the tag <AAA_ROUTINE> rather than the default <ROUTINE> tag. The <AAA_ROUTINE> tag has the default attributes of the <ROUTINE> tag.

The DOUBLERUNNINGHEADS argument to the <SET_TEMPLATE_ROUTINE> tag specifies that the routine section has a double running heading at the top of the page. The top heading is the *running title* specified as an argument to the <ROUTINE_SECTION> tag, and the lower heading is the name of the current routine.

The following example shows how to create a routine section in which each routine description (begun with a <ROUTINE> tag) is in a separate SDML file, and all these descriptions are included into a primary routine description file. For example, the file MYROUTINES.SDML contains the following SDML tags:

```
<INCLUDE> (AAA$CLOSE.SDML)
<INCLUDE> (AAA$OPEN.SDML)
<INCLUDE> (AAA$READ.SDML)
<INCLUDE> (AAA$WRITE.SDML)
```

Each of the included files contains one routine reference description begun with a <ROUTINE> tag. For these files to process correctly, you must precede them with the <ROUTINE_SECTION> tag, which enables the <ROUTINE> tag. These files can have the necessary tags processed before them by specifying the /INCLUDE qualifier on the command line to include a startup definition file. This startup file might include the following tags:

```
2 <ROUTINE_SECTION>(AAA Routines\AAA)
   <SET_TEMPLATE_ROUTINE>(ROUTINE\DOUBLERUNNINGHEADS)
```

If this startup file were named AAAROUTINE_STARTUP.SDML, it could be included using the DOCUMENT /INCLUDE qualifier as in the following example:

```
$ DOCUMENT myroutines SOFT.REF LN03-
_ $ /INCLUDE=AAAROUTINE_STARTUP.SDML
```

When each individual file in MYROUTINES.SDML is processed, the correct sequence of tags will be read in to begin the routine section.

Process multiple files together by using the <INCLUDE> tag to include them into a single master file (such as MYROUTINES.SDML), or include them into a bookbuild profile.

Use the <ELEMENT> tags to include multiple files into a profile. For example, the bookbuild profile file AAAPRO.SDML could contain the following tags:

```
<PROFILE>
<ELEMENT>(AAA$CLOSE.SDML)
<ELEMENT>(AAA$OPEN.SDML)
<ELEMENT>(AAA$READ.SDML)
<ELEMENT>(AAA$WRITE.SDML) <COMMENT>(contains <ENDROUTINE_SECTION> tag)
<ENDPROFILE>
```

SOFTWARE Doctype Tag Reference <ROUTINE_SECTION>

Note that the PROFILE file should include the <ENDROUTINE_SECTION> tag in the appropriate file, so that the template terminates and the book builds correctly.

SOFTWARE Doctype Tag Reference

<RSDEFLIST>

<RSDEFLIST>

Begins a return status definition list in the Routine template.

SYNTAX

$$\langle \text{RSDEFLIST} \rangle [(\left\{ \begin{array}{l} \textit{alternate heading} [\left\{ \begin{array}{l} \backslash \text{NONE} \\ \backslash \text{TEXT} \end{array} \right\} \\ \text{NONE} \end{array} \right\})]]$$

ARGUMENTS

alternate heading

This is an optional argument. It specifies a heading to override the current default text heading for this use of the <RSDEFLIST> tag. The default heading provided by VAX DOCUMENT is Return Values. See the reference description of the <SET_TEMPLATE_HEADING> tag for information on how to modify the default headings for all <RSDEFLIST> tags.

TEXT

This is an optional keyword argument. It specifies that a block of text appears in place of a list of return status values. If you use this keyword, you must position it as the second argument to the <RSDEFLIST> tag.

To use the default heading and to indicate that text follows, specify the following:

```
<RSDEFLIST> (\TEXT)
```

NONE

This is an optional keyword argument. It indicates that the routine does not return any values.

related tags

- <RSITEM>

restrictions

Valid only in the context of the Routine template. If you specify NONE as an argument to the <RSDEFLIST> tag, do not use the <ENDRSDEFLIST> tag.

required terminator

```
<ENDRSDEFLIST>
```

DESCRIPTION

The <RSDEFLIST> tag begins a return status definition list in the Routine template. VAX DOCUMENT formats this list as a 2-column, multipage table. This lets you use the <TABLE_ROW_BREAK>(FIRST) and <TABLE_ROW_BREAK>(LAST) tags to control page breaks in the table if such control is needed.

EXAMPLES

The following three input examples show various uses of the <RSDEFLIST> tag. Outputs from these coding examples appear after the last input example.

The following input example illustrates a return status definition list for a routine with two possible return values.

1 <RSDEFLIST>
 <RSITEM>(SS\$_NORMAL\Service successfully completed.)
 <RSITEM>(SS\$_ACCVIO\Access violation.)
 <ENDRSDEFLIST>

The following input example uses the NONE keyword argument to indicate that a routine does not return any status values. Note that the <ENDRSDEFLIST> tag must not be specified.

2 <RSDEFLIST>(NONE)

The following input example illustrates a single line of descriptive text in a Return Status section.

3 <RSDEFLIST>(\TEXT)
 Any condition values returned by the Record Management Service (RMS),
 Parse.
 <ENDRSDEFLIST>

These input examples produce the following outputs:

**RETURN
VALUES**

SS\$_NORMAL	Service successfully completed.
SS\$_ACCVIO	Access violation.

**RETURN
VALUES**

None.

**RETURN
VALUES**

Any condition values returned by the Record Management Service (RMS),
Parse.

SOFTWARE Doctype Tag Reference

<RSITEM>

<RSITEM>

Specifies the return status value of a routine and lists its meaning.

SYNTAX

<RSITEM>(*code* | *code description*)

ARGUMENTS

code

Specifies the system keyword assigned to the status value.

code description

Specifies the descriptive text explaining the meaning of the return status value.

related tags

- <RSDEFLIST>

restrictions

Valid only in the context of the <RSDEFLIST> tag.

DESCRIPTION

The <RSITEM> tag specifies the return status value of a routine and lists its meaning.

EXAMPLES

These examples show only code fragments, and no actual output. See the <RSDEFLIST> tag description for sample output.

```
1 <RSDEFLIST>
  <RSITEM>(SS$_NORMAL\Normal, successful completion.)
  <RSITEM>(SS$_ACCVIO\Access violation.)
<ENDRSDEFLIST>
```

This example illustrates a Return Status section for a routine that has two possible return values.

```
2 <RSDEFLIST>
  <RSITEM>(SS$_NORMAL\Normal successful completion.)
  <RSITEM>(SMG$_WRONNUMARG\Wrong number of arguments.)
  <RSITEM>(<SPAN>(2\LEFT)Any condition values returned by
  LIB$SCOPY_DXDX, SMG$ADD_KEY_DEF, or by CLI$ routines.)
<ENDRSDEFLIST>
```

This example illustrates the Return Status definition list for a routine that has two specific return values, and uses the tag to place additional explanatory text in the table.

<RUNNING_FEET>

Creates a single line heading at the bottom of each page in a document processed using the SOFTWARE.SPECIFICATION doctype.

SYNTAX <RUNNING_FEET>(title text)

ARGUMENTS *title text*
Specifies the text to be used as a running heading at the foot of the page.

related tags • <RUNNING_TITLE>

restrictions Valid only in the SOFTWARE.SPECIFICATION doctype.

DESCRIPTION The <RUNNING_FEET> tag creates a single line heading at the bottom of each page in a document processed using the SOFTWARE.SPECIFICATION doctype. This heading is called a footer because it appears at the foot of the page. When the same footer is used for several pages, the footers are collectively called running feet.

This tag accepts one argument, the text that should appear as the footer at the bottom of the page. This text outputs exactly as entered, including spacing and capitalization.

Use the <RUNNING_TITLE> tag to create a heading at the top of the page. See the reference description of the <RUNNING_TITLE> tag for more information on that tag.

EXAMPLE The following example shows how to use the <RUNNING_FEET> tag to place the heading Getting the Piece of Paper at the bottom of each page. Note that the running footer will be output exactly as entered.

```
<RUNNING_FEET>(Getting the Piece of Paper)
<HEAD2>(Getting the Piece of Paper\36_GettingthePieceofPaper)
<P>
You can buy clean paper in most major supermarkets, department stores,
and hardware stores. You should try to get ruled paper so that
your letter will be neat and easy to read.
```

SOFTWARE Doctype Tag Reference

<RUNNING_TITLE>

<RUNNING_TITLE>

Creates a 1- or 2-line running heading at the top of each page in a document processed using the SOFTWARE.SPECIFICATION doctype.

SYNTAX

$$\langle \text{RUNNING_TITLE} \rangle (\left\{ \begin{array}{l} \text{OFF} \\ \text{title-1} [\backslash \text{title-2}] \\ [\backslash \text{FIRST_PAGE}] \end{array} \right\})$$

ARGUMENTS

OFF

Specifies that any existing running titles created using the <RUNNING_TITLE> tag should be disabled for the page on which this tag occurs and on any subsequent pages.

title-1

Specifies the text of a running title. If a 2-line title is specified, this title outputs on the upper title line.

title-2

Specifies the optional bottom line of a running title that has two lines.

FIRST_PAGE

This is an optional keyword argument. It specifies that the running title is to begin output on the first output page. If you do not specify this keyword, the running title is output on the page after the current page.

related tags

- <RUNNING_FEET>

restrictions

Valid only in the SOFTWARE.SPECIFICATION doctype.

DESCRIPTION

The <RUNNING_TITLE> tag creates a 1- or 2-line running heading at the top of each page in a document processed using the SOFTWARE.SPECIFICATION doctype. Use the FIRST_PAGE argument to the <RUNNING_TITLE> tag to begin the title lines on the first page of output, rather than on the page after the current page as is the default.

Use the OFF argument to disable any existing running titles created using the <RUNNING_TITLE> tag. These titles will then be disabled for the page on which this tag occurs and on any subsequent pages.

Use the <RUNNING_FEET> tag to create a heading that appears at the bottom of the page. See the reference description of the <RUNNING_FEET> tag for more information on that tag.

EXAMPLES

The following example shows how to use the <RUNNING_TITLE> tag to create the 2-line running title An E. B. Bartz Course: and Writing Quality Correspondence. Since the FIRST_PAGE argument is used, the 2-line running title will appear at the top of the first page.

1 <RUNNING_TITLE>(An E. B. Bartz Course:\Writing Quality Correspondence\FIRST_PAGE)
<HEAD1>(How to Write a Letter\37_HowtoWriteaLetter)
<P>
The first thing that you should do in writing a letter is to get a clean piece of paper and a well-sharpened pencil.

The following example shows how to disable a running title by using the OFF argument to the <RUNNING_TITLE> tag.

2 <COMMENT>(turn off running titles for the following example page)
<RUNNING_TITLE>(OFF)
<HEAD>(An Example of a Letter\38_AnExampleofaLetter)
.
.
.

SOFTWARE Doctype Tag Reference

<SDML_TAG>

<SDML_TAG>

Begins a new tag description.

SYNTAX

<SDML_TAG> (*tag name \ symbol name*)

ARGUMENTS

tag name

Specifies the name of the tag to be described.

symbol name

Specifies the name of the symbol used in all references to the tag.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <SET_TEMPLATE_TAG>
 - <TAG_SECTION>
-

restrictions

Valid only in the context of the Tag template.

DESCRIPTION

The <SDML_TAG> tag begins a new tag description. The <SDML_TAG> tag has the following default format:

- Each <SDML_TAG> tag begins a new page of output.
- Each output page carries a single running title, which is the current SDML tag name.

Use the <SET_TEMPLATE_TAG> tag to replace the <SDML_TAG> tag with a tag specific to your task (for example, <LOCAL_TAG>) or if you want to change the default attributes of the <SDML_TAG> tag. See the description of the <SET_TEMPLATE_TAG> tag in this chapter for more information.

EXAMPLE

The following example shows the Tag template begun using the <TAG_SECTION> tag. In this tag section, the <SDML_TAG> tag begins the tag description for the local <LEVEL1> tag.

```
<TAG_SECTION>(Local Tags)
<SDML_TAG>(LEVEL1\level_tag_sym)
<OVERVIEW>
Labels the first level of a diagram.
<ENDOVERVIEW>
```

<SET_TEMPLATE_ARGITEM>

Sets up a user-defined argument list for documenting callable routines for multiple operating system platforms. Also allows translation of the argument list.

SYNTAX

```
<SET_TEMPLATE_ARGITEM>(tag name
                        \ text one
                        \ text two
                        \ text three
                        [\ text four]
                        [\ text five]
                        \ longest text)
```

ARGUMENTS***tag name***

Defines the <ARGITEM> tag name.

text one

Defines the first related text to be automatically output for argument two in the list.

text two

Defines the second related text to be automatically output for argument three in the list.

text three

Defines the third related text to be automatically output for argument four in the list.

text four

This is an optional argument; if you do not use it, you must specify it as null. Defines the fourth related text to be automatically output for argument five in the list.

text five

This is an optional argument; if you do not use it, you must specify it as null. Defines the fifth related text to be automatically output for argument six in the list.

longest text

Defines the longest text in the argument list. It is positional; it must be argument seven.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_ARGITEM>

related tags

- <ARGDEFLIST>
 - <ARGITEM>
 - <ROUTINE_SECTION>
-

restrictions

Valid only in the context of a <ROUTINE_SECTION> tag.

Only one <SET_TEMPLATE_ARGITEM> tag has effect at a time, so that an occurrence of the <SET_TEMPLATE_ARGITEM> tag cancels the setting of a previous one.

DESCRIPTION

The <SET_TEMPLATE_ARGITEM> tag sets up a user-defined argument list for documenting callable routines for multiple operating system platforms. This tag also allows translation of the argument list.

EXAMPLE

The following example shows how to use the <SET_TEMPLATE_ARGITEM> tag.

```
<ROUTINE_SECTION>
<SET_TEMPLATE_ARGITEM>(UNIX_ARGS
    \UNIX type
    \MS-DOS type
    \access
    \mechanism
    \special
    \MS-DOS type) <COMMENT>( Longest text )

<ARGDEFLIST>

<UNIX_ARGS>(unix argument\one\two\three\four)
<ARGDEF>This invocation of <UNIX_ARG> tag has four parameters.

<UNIX_ARGS>(unix argument two\one\two\three\four\five)
<ARGDEF>This invocation of <UNIX_ARG> tag has five parameters.
<ENDARGDEFLIST>

<SET_TEMPLATE_ARGITEM>(MSDOS_ARGS
    \MS-DOS type
    \access
    \mechanism
    \usage
    \ <COMMENT>(Note blank argument; longest
        text must be in argument 7!)
    \additionalongertext)

<ARGDEFLIST>
<ARGITEM>(argument\one\two\three\four)
<ARGDEF>This is a standard <ARGDEF> tag.

<MSDOS_ARGS>(msdos argument two\one\two\three\four)
<ARGDEF>This is a new tag, overriding the last.

<ARGITEM>(argument\one\two\three\four)
<ARGDEF>This is a standard <ARGDEF> tag.

<ENDARGDEFLIST>
<ARGDEFLIST>
<MSDOS_ARGS>(msdos argument two\one\two\three\four)
<ARGDEF>This is a new tag, overriding the last.

<ENDARGDEFLIST>
```

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_ARGITEM>

```
<SET_TEMPLATE_ARGITEM> (OST_ARGS
                        \OST type
                        \access
                        \mechanism
                        \ <COMMENT>(Note blank argument;
                          longest text must be in argument 7!)
                        \
                        \additionallongertext)
```

<ARGDEFLIST>

```
<MGDOS_ARGS>(msdos argument two\one\two\three)
<ARGDEF>This is a new tag, overriding the last.
```

```
<MGDOS_ARGS>(msdos argument two\one\two\three)
<ARGDEF>This is a new tag, overriding the last.
```

```
<MGDOS_ARGS>(msdos argument two\one\two\three\four)
<ARGDEF>This is a new tag, overriding the last.
```

<ENDARGDEFLIST>

<ENDROUTINE_SECTION>

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_COMMAND>

<SET_TEMPLATE_COMMAND>

Defines a new tag with the same function as the <COMMAND> tag, and changes the format of command descriptions produced using the new tag.

SYNTAX

<SET_TEMPLATE_COMMAND>(*tag name*
 {
 [\ DOUBLE-
 RUNNINGHEADS]
 [\ NONEWPAGE]
 [\ STACK]
 }
 [\ *symbol*
 name])

ARGUMENTS

tag name

Specifies the name of the template tag being defined. This tag name must be a valid tag name less than 31 characters and must not be the same as an existing tag name other than COMMAND (the default tag name).

DOUBLERUNNINGHEADS

This is an optional keyword argument. It enables two running titles at the top of every page. The top running title is set by the <COMMAND_SECTION> tag or by the heading of the most recent <CHAPTER> or <APPENDIX> tag. By default, if a doctype does not call for running top titles, only the current command name is placed at the top of each page.

NONEWPAGE

This is an optional keyword argument. It prevents command descriptions from starting on new pages. By default, each *tag name* template tag begins a command description on a new page.

STACK

This is an optional keyword argument. It stacks multiple arguments to the *tag name* tag. By default, when you specify multiple arguments, the second and third arguments are assumed to be optional descriptive information, and output on the same line as the command name.

symbol name

This is an optional argument for printed output, but is required for using the file in a bookbuild for Bookreader. This argument specifies the name of the symbol used in all references to this tag.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <COMMAND>
 - <COMMAND_SECTION>
-

restrictions

Valid only in the context of the <COMMAND_SECTION> tag in the Command template.

DESCRIPTION

The <SET_TEMPLATE_COMMAND> tag defines a new tag with the same function as the <COMMAND> tag, and changes the format of command descriptions produced using the new tag.

This tag also changes the default attributes associated with the <COMMAND> tag, or gives your new tag different attributes than the <COMMAND> tag.

EXAMPLES

1 <COMMAND_SECTION> (FDL Commands\\NEWPAGE)
<SET_TEMPLATE_COMMAND> (FDL_COMMAND\NONEWPAGE\STACK\DOUBLERUNNINGHEADS)
<FDL_COMMAND> (STARTUP_FILE\STF)

These tags begin a command section and define the <FDL_COMMAND> tag as the command descriptor tag. The attributes set are as follows:

- Commands do not start on new pages.
- When two arguments are specified to the <FDL_COMMAND> tag, the arguments are stacked. That is, the second argument is printed under the first.
- Each page carries a 2-line running title. The top line is FDL Commands and the second line is the name of the command description that is current at the top of the page.

Note that this command sequence is most appropriate for a series of command descriptions that occur in a chapter.

2 <PART>
<PART_PAGE>
<TITLE> (PART II\XYZ Commands)
<ENDPART_PAGE> (RENUMBER)
<COMMAND_SECTION> (XYZ Commands\XYZ)
<SET_TEMPLATE_COMMAND> (XYZ_COMMAND)

These tags begin a document part in which only command descriptions occur. The RENUMBER argument on the <ENDPART_PAGE> tag specifies that the first page of the new Part (the part page itself) should be numbered page 1. The <COMMAND_SECTION> tag sets the folio prefix to XYZ; that is, pages will be numbered XYZ-3, XYZ-4, and so on.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_COMMAND>

No attributes are specified for the command descriptions produced by the <XYZ_COMMAND> tag. These commands will have default attributes. The first command will start on a new page. Since RENUMBER was specified on the <ENDPART_PAGE>tag, the first command will be on the page numbered XYZ-3.

<SET_TEMPLATE_HEADING>

Overrides the heading for all subsequent uses of a template tag.

SYNTAX

<SET_TEMPLATE_HEADING>(*element keyword*
 \ *default heading*)

ARGUMENTS

element keyword

Specifies the standard reference template element whose default heading you want to override. The reference element keywords, the related template tags, and the system default headings are listed in the following table.

Keyword	Template Tag	Default Heading
Command Template		
DESCRIPTION	<DESCRIPTION>	Description
FORMAT	<FORMAT>	Format
PARAMDEFLIST	<PARAMDEFLIST>	Parameters
PROMPTS	<PROMPTS>	Prompts
QUALDEFLIST	<QUALDEFLIST>	Qualifiers
RETURN_VALUE	<RETURN_VALUE>	Return Value
RESTRICTIONS	<RESTRICTIONS>	Restrictions
Routine Template		
ARGDEFLIST	<ARGDEFLIST>	Arguments
DESCRIPTION	<DESCRIPTION>	Description
FORMAT	<FORMAT>	Format
RSDEFLIST	<RSDEFLIST>	Return Values
Statement Template		
STATEMENT_FORMAT	<STATEMENT_FORMAT>	Format
Tag Template		
All defaults established for the command template.		
ARGDEFLIST	<ARGDEFLIST>	Arguments
PARAMDEFLIST	<PARAMDEFLIST>	Parameters
RELATED_TAGS	<RELATED_TAGS>	Related Tags
TERMINATOR	<TERMINATING_TAG>	Required Terminator

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_HEADING>

related tags

- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_PARA>
- The global <LIST> tag

DESCRIPTION

The <SET_TEMPLATE_HEADING> tag overrides the heading for all subsequent uses of a template tag.

EXAMPLE

The following example shows how to use the <SET_TEMPLATE_HEADING> tag to change the Command template so that the default heading of the keyword **FORMAT** is changed from **Format** to **Syntax**.

```
<COMMAND_SECTION> (FDL Commands)
<SET_TEMPLATE_HEADING> (FORMAT\Syntax)
```

<SET_TEMPLATE_LIST>

Creates a user-defined set of tags for listing information.

SYNTAX

<SET_TEMPLATE_LIST>(*list tag name*
 \ *default list heading*
 \ *list item tag name*
 \ *list type*
 [\ *heading level*])

ARGUMENTS

list tag name

Specifies the name of the tag that will introduce the list of items in the list template being defined. This tag name must be a valid tag name less than 28 characters; it must not be the same as an existing SDML tag name.

default list heading

Specifies the default text heading to be output by the list template being defined.

item tag name

Specifies the name of the tag to be used to indicate individual items in the list being defined. This name must be a valid tag name.

list type

Specifies the keyword that indicates the type of list the list being defined is based on. These keywords create the same list format as the same keywords used with the global <LIST> tag. These keywords are as follows:

SIMPLE
NUMBERED
UNNUMBERED

heading level

This is an optional argument. It specifies the template heading level to be associated with the tag. Valid values are 1 and 2, indicating a primary template heading or a secondary template heading. If you do not specify this argument, the value defaults to 2, and the secondary template heading is used.

related tags

- <SET_TEMPLATE_HEADING>
- <SET_TEMPLATE_PARA>
- <SET_TEMPLATE_TABLE>
- The global <LIST> tag

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_LIST>

restrictions

Valid only in the context of a <COMMAND_SECTION>, <ROUTINE_SECTION>, <STATEMENT_SECTION>, or <TAG_SECTION> tag.

DESCRIPTION

The <SET_TEMPLATE_LIST> tag creates a user-defined set of tags for listing information.

EXAMPLES

The following examples in this section assume that the following tag has been specified to enable the <SHOPPING_LIST>, <SITEM>, and <ENDSHOPPING_LIST> tags.

```
<SET_TEMPLATE_LIST>(SHOPPING_LIST\Shopping List\SITEM\NUMBERED)
```

The following example generates the default heading Shopping List, and starts a numbered list. Each <SITEM> tag generates another numeric item in the list, as follows:

```
1 <SHOPPING_LIST>  
   <SITEM>one item  
   <SITEM>second  
   <ENDSHOPPING_LIST>
```

This example produces the following output:

shopping list

```
1 one item  
2 second
```

The following example shows how to override the default heading shopping list heading and produce the text None.

```
2 <SHOPPING_LIST>(Bloomingdales\NONE)
```

This example produces the following output:

bloomingdales

None.

<SET_TEMPLATE_PARA>

Defines a set of template tags for setting the format of a paragraph of information.

SYNTAX <SET_TEMPLATE_PARA> (*tag name*
 \ *default heading*
 [\ *heading level*])

ARGUMENTS *tag name*
Specifies the user-defined name of the tag that begins the paragraph template and marks the beginning of the text in that paragraph. This tag name must be a valid tag name less than 28 characters; it must not be the same as an existing SDML tag name.

default heading
Specifies a default heading to be associated with the paragraph template. You can override this default heading in the tag invocation, if needed.

heading level
This is an optional argument. It specifies the template heading level to be associated with the tag. Valid values are 1 and 2, indicating a primary template heading or a secondary template heading. If you do not specify this argument, it defaults to 2 and the secondary template heading is used.

related tags

- <SET_TEMPLATE_HEADING>
- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_TABLE>

restrictions Valid only in the context of a reference template.

DESCRIPTION The <SET_TEMPLATE_PARA> tag defines a set of template tags for setting the format of a paragraph of information.

EXAMPLES The following examples show how to use the <SET_TEMPLATE_PARA> tag.

The <SET_TEMPLATE_PARA> tag in the following example creates the template tags <SIDE_EFFECTS> and <ENDSIDE_EFFECTS> as the beginning and ending tags of a paragraph template with a default heading of Side Effects:.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_PARA>

Note that this definition of the <SIDE_EFFECTS> tag is the one used in the following examples.

1 <SET_TEMPLATE_PARA>(SIDE_EFFECTS\Side Effects:)
:
:
<SIDE_EFFECTS>
Modifying the arguments to the PLACE command changes the positioning of the page number.
<ENDSIDE_EFFECTS>

The following example shows how to specify the template tag <SIDE_EFFECTS> with the alternate heading Desired Effect=.

2 <SIDE_EFFECTS>(Desired Effect=)
Modifying the arguments to the PLACE command changes the positioning of the page number.
<ENDSIDE_EFFECTS>

The following example shows how to specify the NONE keyword to the <SIDE_EFFECTS> template tag.

3 <SIDE_EFFECTS>(NONE)

<SET_TEMPLATE_ROUTINE>

Defines a new reference element tag name to use in the routine template, and specifies the formatting attributes for the new tag.

SYNTAX

<SET_TEMPLATE_ROUTINE> (*tag name* { [*\ DOUBLE-RUNNINGHEADS*] } { [*\ NONEWPAGE*] } { [*\ STACK*] } [*\ symbol name*])

ARGUMENTS

tag name

Specifies the name of the template tag being defined. This tag name must be a valid tag name less than 31 characters and must not be the same as an existing tag name other than ROUTINE (the default tag name).

DOUBLERUNNINGHEADS

This is an optional keyword argument. It specifies that the routine descriptions will have two running titles at the top of every page. The top running title is either the title you set with the <ROUTINE_SECTION> tag, or the heading set in the most recent <CHAPTER> or <APPENDIX> tag. By default, if a doctype does not call for running top titles, only the current routine name is placed at the top of each page.

NONEWPAGE

This is an optional keyword argument. Routine descriptions are not to start on new pages. By default, each *tag name* template tag begins a routine description on a new page.

STACK

This is an optional keyword argument. It specifies that when you give multiple arguments to the *tag name* tag, the arguments are stacked at the beginning of the page.

By default, when you give multiple arguments, the second and third arguments are assumed to be optional descriptive information, and output as shown in the examples in the <ROUTINE> tag description.

symbol name

This is an optional argument for printed output, but is required for using the file in a bookbuild for Bookreader. This argument specifies the name of the symbol used in all references to this tag.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_ROUTINE>

related tags

- <ROUTINE>
 - <ROUTINE_SECTION>
-

restrictions

Valid only in the context of the <ROUTINE_SECTION> tag in the Routine template.

DESCRIPTION

The <SET_TEMPLATE_ROUTINE> tag defines a new reference element tag name to use in the routine template, and specifies the formatting attributes for the new tag.

This tag also changes the default attributes associated with the <ROUTINE> tag or your new tag.

EXAMPLES

The following two examples show how to use the <SET_TEMPLATE_ROUTINE> tag.

The first tag sequence begins a routine section and defines the <FDL_ROUTINE> tag as the routine descriptor tag. The attributes set are as follows:

- Routines do not start on new pages. However, because NEWPAGE is specified to the <ROUTINE_SECTION> tag, the first reference description begins on a new page.
- When two arguments are specified to the <FDL_ROUTINE> tag, the arguments are stacked; that is, the second argument is placed under the first on output.
- Each page carries a 2-line running title. The top line is FDL Routines and the second line is the name of the routine description that is current at the top of the page.

Note that this routine sequence is most appropriate for a series of routine descriptions that occur in a chapter.

1 <ROUTINE_SECTION>(FDL Routines\\NEWPAGE)
<SET_TEMPLATE_ROUTINE>(FDL_ROUTINE\NONEWPAGE\STACK\DOUBLERUNNINGHEADS)
<FDL_ROUTINE>(\$STARTUP_FILE\STF)

The second tag sequence begins a document part in which only routine descriptions occur. The RENUMBER argument on the <ENDPART_PAGE> tag specifies that the first page of the new part (the part page itself) is numbered page 1. The <ROUTINE_SECTION> tag sets the folio prefix to XYZ; that is, pages will be numbered XYZ-3, XYZ-4, and so on.

No attributes are specified for the routine descriptions produced by the <XYZ_ROUTINE> tag. These routines will have default attributes. The first routine will start on a new page. Because RENUMBER was specified on the <ENDPART_PAGE> tag, the first routine will be on the page numbered XYZ-3.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_ROUTINE>

2 <PART>
<PART_PAGE>
<TITLE> (PART II\XYZ Routines)
<ENDPART_PAGE> (RENUMBER)

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_STATEMENT>

<SET_TEMPLATE_STATEMENT>

Defines a new reference element tag name to use in the statement template and specifies the formatting attributes for the new tag.

SYNTAX

<SET_TEMPLATE_STATEMENT>(*tag name*
 { [\ DOUBLE-
 RUNNINGHEADS]
 [\ NONEWPAGE]
 [\ STACK]
 [\ *symbol*
 name])

ARGUMENTS

tag name

Specifies the name of the template tag being defined. This tag name must be a valid tag name less than 31 characters and must not be the same as an existing tag name other than STATEMENT or FUNCTION (which are the default tag names).

DOUBLERUNNINGHEADS

This is an optional keyword argument. It specifies that the statement descriptions will have two running titles at the top of every page. The top running title is set by the <STATEMENT_SECTION> tag or by the heading of the most recent <CHAPTER> or <APPENDIX> tag. By default, if a doctype does not call for running top titles, only the current statement name is placed at the top of each page.

NONEWPAGE

This is an optional keyword argument. It specifies that statement descriptions are not to start on new pages. By default, each *tag name* template tag begins a statement description on a new page.

STACK

This is an optional keyword argument. It specifies that when multiple arguments are specified for the *tag name* tag, the arguments should be stacked at the beginning of the page.

By default, when multiple arguments are specified, the second and third arguments are assumed to be optional descriptive information, and are output on the same line as the statement name.

symbol name

This is an optional argument for printed output, but is required for using the file in a bookbuild for Bookreader. This argument specifies the name of the symbol used in all references to this tag.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_STATEMENT>

related tags

- <FUNCTION>
 - <STATEMENT>
 - <STATEMENT_SECTION>
-

restrictions

Valid only in the context of the <STATEMENT_SECTION> tag.

DESCRIPTION

The <SET_TEMPLATE_STATEMENT> tag defines a new reference element tag name to use in the statement template and specifies the formatting attributes for the new tag.

This tag also lets you change the default attributes associated with the <STATEMENT> tag or your new tag.

EXAMPLES

The following tag sequence begins a statement section and defines the tag <BASIC_STATEMENT> as the statement descriptor tag. The attributes set are as follows:

- Statements do not start on new pages.
- Each page carries a 2-line running title. The top line is BASIC Statements and the second line is the name of the statement description that is current at the top of the page.

Note that this statement sequence is most appropriate for a series of statement descriptions that occur in a chapter.

1 <STATEMENT_SECTION>(BASIC Statements\\NEWPAGE)
<SET_TEMPLATE_STATEMENT>(BASIC_STATEMENT\NONEWPAGE\DOUBLERUNNINGHEADS)
<BASIC_STATEMENT>(STARTUP_FILE\STF)

The following tag sequence begins a document part in which only statement descriptions occur. The RENUMBER argument on the <ENDPART_PAGE> tag specifies that the first page of the new part (the part page itself) should be numbered page 1. The <STATEMENT_SECTION> tag sets the folio prefix to XYZ; that is, pages will be numbered XYZ-3, XYZ-4, and so on.

No attributes are specified for the statement descriptions produced by <XYZ_STATEMENT>; these statements have default attributes. The first statement will start on a new page. Because RENUMBER was specified on the <ENDPART_PAGE> tag, the first statement will be on the page numbered XYZ-3.

2 <PART>
<PART_PAGE>
<TITLE>(PART II\XYZ Statements)
<ENDPART_PAGE>(RENUMBER)
<STATEMENT_SECTION>(XYZ Statements\XYZ)
<SET_TEMPLATE_STATEMENT>(XYZ_STATEMENT)

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_SUBCOMMAND>

The following default attributes are set in the previous example:

- Each subcommand description that begins with the <QUALIFIER> tag starts on a new page of output.
- Each page carries a 2-line running title. The top line is Qualifiers and the second line is the name of the subcommand description that is current at the top of the page.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_TABLE>

<SET_TEMPLATE_TABLE>

Defines a set of template tags for setting information in 2- or 3-column lists.

SYNTAX

<SET_TEMPLATE_TABLE>(*table tag name*
 \ *default table heading*
 \ *table row tag name*
 \ *column count*
 \ *column widths*
 [\ *table column headings*])

ARGUMENTS

table tag name

Specifies the user-defined name of the tag that begins the user-defined table. This tag name must be a valid tag name less than 28 characters; it must not be the same as an existing SDML tag name.

default table heading

Specifies a default heading to be output over the entire user-defined table.

table row tag name

Specifies the name of the tag to be used to indicate individual table rows in the table being defined. This name must be a valid tag name. For example, if the *table row tag name* argument is specified as SAMP_ROW, the individual table row tag will be <SAMP_ROW>. The user-defined tag created by this argument is similar to the global <TABLE_ROW> tag.

column count

Specifies the number of columns in the user-defined table. The accepted arguments are as follows:

- 2 — Specifies that the table is to have two columns.
- 3 — Specifies that the table is to have three columns.

column widths

Specifies the approximate widths of the table columns. The width of the last table column is determined by VAX DOCUMENT. If you specify a 2-column table, you must specify only a single column width argument, as shown in the following code example:

```
<SET_TEMPLATE_TABLE>(KEYVALS\Keyword Values\KEYVAL\2\10\Keyword\Value)
```

If you specify a 3-column table, you must specify two column-width arguments, as shown in the following code example:

```
<SET_TEMPLATE_TABLE>(KEYVAL_TABLE\Keyword Ranges  
\KEYVAL\3\10\10\Keyword\High\Lower)
```

table column headings

This is an optional argument. It specifies the default headings for each column in the user-defined table. If you specify a 2-column table, you can specify up to two heading arguments. If you specify a 3-column table, you can specify up to three heading arguments.

related tags

- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_PARA>

restrictions

Valid only in the context of a reference template.

DESCRIPTION

The <SET_TEMPLATE_TABLE> tag defines a set of template tags for setting information in 2- or 3-column lists. Tables created using this tag may have either two or three columns. This tag requires five arguments and accepts the optional *table column headings* argument.

If you choose to omit the *table column headings* argument, then the table will not have any column headings and will not output rules in the table.

EXAMPLES

In the following example, the <SET_TEMPLATE_TABLE> tag sets the default heading to be Best Songs, enables a 2-column table, sets the text supplied to the two <45RPM> tags in the table, and then terminates the table.

```
1 <SET_TEMPLATE_TABLE>(RECORDTABLE\Best Songs\45RPM\2\12\Performer\Song Title)
  <RECORDTABLE>
  <45RPM>(Sinatra\Strangers in the Night)
  <45RPM>(Moody Blues\Nights in White Satin)
  <ENDRECORDTABLE>
```

This example produces the following output.

best songs

Performer	Song Title
Sinatra	Strangers in the Night
Moody Blues	Nights in White Satin

The following example shows how to specify the NONE keyword to the <RECORDTABLE> tag.

```
2 <RECORDTABLE>(NONE)
```

This example produces the following output.

best songs

None.

The following example shows how to define a 3-column table with headings. The tags defined are the <OPCODES>, <ENDOPCODES>, and <OPS> tags.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_TABLE>

3 <SET_TEMPLATE_TABLE>(OPCODES\codes\OPS\3\6\6\2-byte\3-byte\4-byte)
<OPCODES>
<OPS>(abc\def\ghi)
<ENDOPCODES>

This example produces the following output.

codes

2-byte	3-byte	4-byte
abc	def	ghi

<SET_TEMPLATE_TAG>

Defines a new reference element tag name to use in the tag template, and specifies formatting attributes for the newly defined tag.

SYNTAX

<SET_TEMPLATE_TAG>(*tag name*
{
 [\ DOUBLE-
 RUNNINGHEADS]
 [\ NONEWPAGE]
 [\ STACK]
 [\ symbol name]
})

ARGUMENTS

tag name

Specifies the name of the template tag being defined. This tag name must be a valid tag name less than 31 characters and must not be the same as an existing tag name other than SDML_TAG (which is the default tag name).

DOUBLERUNNINGHEADS

This is an optional keyword argument. It specifies that the tag descriptions for the *tag name* tag will have two running titles at the top of every page. The top running title is set by the <TAG_SECTION> tag or by the heading of the most recent <CHAPTER> or <APPENDIX> tag. By default, if a doctype does not call for running top titles, only the current tag name prints at the top of each page.

NONEWPAGE

This is an optional keyword argument. It specifies that tag descriptions are not to start on new pages. By default, each *tag name* template tag begins a tag description on a new page.

STACK

This is an optional keyword argument. It specifies that when you give multiple arguments to the *tag name* tag, the arguments are stacked at the beginning of the page.

By default, when you specify multiple arguments, the second and third arguments are assumed to be optional descriptive information, and are output on the same line as the tag name.

symbol name

This is an optional argument for printed output, but is required for using the file in a bookbuild for Bookreader. This argument specifies the name of the symbol used in all references to this tag.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

SOFTWARE Doctype Tag Reference

<SET_TEMPLATE_TAG>

related tags

- <SDML_TAG>
 - <TAG_SECTION>
-

restrictions

Valid only in the context of the <TAG_SECTION> tag.

DESCRIPTION

The <SET_TEMPLATE_TAG> tag defines a new reference element tag name to use in the tag template, and specifies formatting attributes for the newly-defined tag.

This tag also lets you alter the default attributes associated with the <SDML_TAG> tag (or the tag you replaced with the <SDML_TAG> tag using the <SET_TEMPLATE_TAG> tag).

EXAMPLE

The following example shows how to use the <SET_TEMPLATE_TAG> tag to alter the default format of the Tag template.

```
<TAG_SECTION>(Tag Template Tags)
<SET_TEMPLATE_TAG>(LOCAL_TAG\DOUBLERUNNINGHEADS)
<LOCAL_TAG>(LEVEL1)
```

This tag sequence begins a tag section and defines the <LOCAL_TAG> tag for introducing new tag descriptions. These attributes are as follows:

- Each tag description begins on a new page.
- Each page carries a 2-line running title. The top line is Tag Template Tags and the second line is the name of the tag description that is current at the top of the page.

<SIGNATURES>

Begins a list of signatures that are to appear in the front matter portion of a document processed using the SOFTWARE.SPECIFICATION doctype.

SYNTAX

<SIGNATURES>[(NEWPAGE)]

ARGUMENTS

NEWPAGE

This is an optional keyword argument. It specifies that the signature list is to begin on a new page.

related tags

- <AUTHOR>
- <BYLINE>
- The global <FRONT_MATTER> tag

restrictions

Available only in the SOFTWARE.SPECIFICATION doctype following the global <FRONT_MATTER> tag.

DESCRIPTION

The <SIGNATURES> tag begins a list of signatures that are to appear in the front matter portion of a document processed using the SOFTWARE.SPECIFICATION doctype. Each person's name is listed by using the <BYLINE> tag following the <SIGNATURES> tag. The <BYLINE> tag places the name of the person, and additional information about that person (such as his or her title or affiliation), below a line on which the person is to sign.

See the reference description of the <BYLINE> tag for more information on that tag.

EXAMPLE

See the example in the <BYLINE> tag description.

SOFTWARE Doctype Tag Reference

<STATEMENT>

<STATEMENT>

Begins a new statement description.

SYNTAX

<STATEMENT> (*statement name* \ *symbol name*)

ARGUMENTS

statement name

Specifies the name of the statement to be described.

symbol name

Specifies the name of the symbol used in all references to the tag.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <FUNCTION>
- <SET_TEMPLATE_STATEMENT>
- <STATEMENT_SECTION>

DESCRIPTION

The <STATEMENT> tag begins a new statement description. This description is for a single statement in the context of the <STATEMENT_SECTION> tag. This tag has the following default format:

- Each <STATEMENT> tag begins a new page of output.
- Each output page carries a single running title, which is the current statement name.

Use the <SET_TEMPLATE_STATEMENT> tag to replace the <STATEMENT> tag with a tag specific to your task (for example, the <ABC_STATEMENT> tag), or if you wish to change the default attributes of the <STATEMENT> tag. See the description of the <SET_TEMPLATE_STATEMENT> tag for more information.

EXAMPLE

In the following example, the <STATEMENT_SECTION> tag enables the tags for a statement description. By default, the description of the statement OPEN has the following attributes:

- The statement description begins on a new page.
- If the statement carries for more than a page, the name OPEN is carried as a running top title on each page.

```
<STATEMENT_SECTION>  
<STATEMENT>(OPEN)  
<OVERVIEW>  
<ellipsis>
```


SOFTWARE Doctype Tag Reference

<STATEMENT_FORMAT>

- *<FCMD>(statement-keyword\parameter-list)*

This form should be used for statement functions, in which a statement and its parameters are not separated by blank spaces.

Following the formatted statement, the <CONSTRUCT_LIST> and <STATEMENT_LINE> tags can be used to expand on the meanings of variable names specified in the format. Such variable names can be coded using the <VARIABLE> or <KEYWORD> tags.

EXAMPLES

The following two input examples show various uses of the <STATEMENT_FORMAT> tag. Output from these coding examples appears after the last input example.

The following input example shows a statement format section that uses the <FFUNC> tag to present the format of a statement.

```
1 <STATEMENT_FORMAT>
  <FFUNC>(real-vbl\=ABS<VARIABLE>((real-exp)))
  <endstatement_format>
```

The following input example illustrates the use of multiple formats for a single statement, with special headings for each. Note the special use of the <FORMAT_SUBHEAD> tag to introduce each specific format.

```
2 <statement_format>(\multiple)
  <FORMAT_SUBHEAD>(String Variable To Array)
  <FCMD>(CHANGE) <FPARMS>(str-exp <KEYWORD>(TO) num-array)
  <FORMAT_SUBHEAD>(Array to String Variable)
  <FCMD>(CHANGE) <FPARMS>(num-array <KEYWORD>(TO) str-vbl)
  <endstatement_format>
```

These input examples produce the following output:

Format

real-vbl=ABS(real-exp)

Format

String Variable To Array

CHANGE str-exp TO num-array

Array to String Variable

CHANGE num-array TO str-vbl

<STATEMENT_LINE>

Indicates the position of a valid statement line in the context of a statement format or a construct list.

SYNTAX <STATEMENT_LINE>[(*text*[\ *INDENT*])]

ARGUMENTS *text*

This is an optional argument. It specifies the text for a valid statement line. If no text is specified, the default statement line output is as follows:

[*statement*] . . .

INDENT

This is an optional keyword argument. It indicates that the statement line is to be indented from the margin at which the current statement format or construct list is being set.

related tags

- <CONSTRUCT>
- <FCMD>
- <FPARMS>

restrictions

Valid only in the context of the Statement reference template.

DESCRIPTION

The <STATEMENT_LINE>tag indicates the position of a valid statement line in the context of a statement format or a construct list.

EXAMPLES

The following three input examples show various uses of the <STATEMENT_LINE> tag. Output from these coding examples appear after the last input example.

The following input example shows how to use <STATEMENT_LINE> in a formatted statement section.

1 <STATEMENT_FORMAT>
<FCMD>(RECORD) <FPARMS>(rec-nam)
<STATEMENT_LINE>(rec-component)
<ELLIPSIS>
<FCMD>(END RECORD) <FPARMS>([rec-nam])
<ENDSTATEMENT_FORMAT>

The following input example shows how to use the <STATEMENT_LINE> tag in a construct list. Note that the first use of the tag specifies *INDENT* as the second argument.

SOFTWARE Doctype Tag Reference

<STATEMENT_LINE>

```
2 <STATEMENT_FORMAT>
  <CONSTRUCT_LIST>(group-clause:)
  <construct>(group-clause:)
    <keyword>(GROUP) group-nam [ ( int-const, <hellipsis> ) ]
    <statement_line>(rec-component\indent)
    <ellipsis>
    <statement_line>(<keyword>(END GROUP) [ group-nam ])
  <ENDCONSTRUCT_LIST>
  <ENDSTATEMENT_FORMAT>
```

The following input example shows the default output for the <STATEMENT_LINE> tag when it is used in a statement format section.

```
3 <STATEMENT_FORMAT>
  <FCMD>(SUB) <FPARMS>(sub-name [ pass-mech ] [ ( [ formal-param ],
    <hellipsis> ) ])
  <statement_line>
  <FCMD>(<list>(stacked\braces)
    <le>END SUB
    <le>SUBEND<endlist>) <FPARMS>()
  <endstatement_format>
```

These input examples produce the following outputs:

Format

```
RECORD  rec-nam
          rec-component
          .
          .
          .
END RECORD [ rec-nam ]
```

Format

```
group-clause:  GROUP group-nam [ ( int-const, . . . ) ]
                rec-component
                .
                .
                .
                END GROUP [ group-nam ]
```

Format

SUB *sub-name* [*pass-mech*][([*formal-param*], ...)]
[*statement*]...

{ **END SUB** }
{ **SUBEND** }

SOFTWARE Doctype Tag Reference

<STATEMENT_SECTION>

<STATEMENT_SECTION>

Begins a statement reference section, enables tags reserved for use in statement sections, and sets paging attributes.

SYNTAX

<STATEMENT_SECTION>(((*running title*)
[\ *number prefix*]
[\ **NEWPAGE**]))

ARGUMENTS

running title

This is an optional argument. It specifies a top-level running heading to be used throughout the statement section. If you do not specify this argument, the running headings are determined as described in 10.13.

number prefix

This is an optional argument. It specifies a character-string prefix to be used to construct page numbers (folios) and formal figure, table, and example numbers. If you do not specify this argument, the page and formal element numbering are determined as described in 10.13.

NEWPAGE

This is an optional keyword argument. It indicates that the statement section should begin on a new page. This argument is only meaningful in two cases:

- When you have previously entered the <SET_TEMPLATE_STATEMENT> tag with the NONEWPAGE keyword to specify that each new statement in this statement section should not begin on a new page
- When you want to place one or more pages of text between the end of a part page and the beginning of a statement section.

related tags

- <CONSTRUCT>
- <CONSTRUCT_LIST>
- <FCMD>
- <FFUNC>
- <FPARMS>
- <FORMAT_SUBHEAD>
- <FUNCTION>
- <OVERVIEW>
- <SET_TEMPLATE_HEADING>
- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_PARA>

SOFTWARE Doctype Tag Reference

<STATEMENT_SECTION>

- <SET_TEMPLATE_STATEMENT>
- <SET_TEMPLATE_TABLE>
- <STATEMENT>
- <STATEMENT_FORMAT>
- <STATEMENT_LINE>

**required
terminator**

<ENDSTATEMENT_SECTION>

DESCRIPTION

The <STATEMENT_SECTION> tag begins a statement reference section, enables tags reserved for use in statement sections, and sets paging attributes. You can locate a statement section in a chapter or an appendix, or following a part page (that is, in a document section begun with the <PART_PAGE> tag). You code a statement section in a chapter or an appendix in the same manner; statement sections in parts are handled differently.

If your statement section follows a part page, and you include text between the part page and the statement section, specify the NEWPAGE keyword as the third argument to the <STATEMENT_SECTION> tag. This causes the statement section to begin on a new page. The following code fragment shows a statement section that begins on a new page:

```
<STATEMENT_SECTION> (\SD\NEWPAGE)  
<HEAD1> (Statement Format\44_StatementFormat)
```

When you use the <STATEMENT_SECTION> tag in a chapter or an appendix, and you want to place text after the statement section in that chapter or appendix, you must end the statement section with the <ENDSTATEMENT_SECTION> tag and place the text after that tag. By default, this text begins on a new page of output.

Specify the NONEWPAGE argument to the <ENDSTATEMENT_SECTION> tag if you do not want the text to begin on a new page of output. The following code fragment shows the end of a statement section that specifies that the subsequent text not be placed on a new page:

```
<ENDSTATEMENT_SECTION> (NONEWPAGE)
```

When the <ENDSTATEMENT_SECTION> tag is specified in the context of a chapter or appendix, it resets the default running titles to those in effect for the chapter or appendix, so the last page of the last statement description in the statement section may not carry the last statement's name as the running heading. Instead it may carry the running title used by the chapter or appendix.

The <STATEMENT_SECTION> tag can be used more than once in a document. By specifying arguments to that tag, and by using the <SET_TEMPLATE_STATEMENT> tag to specify additional attributes, you can tailor statement sections that meet the specific requirements of your documentation.

SOFTWARE Doctype Tag Reference

<STATEMENT_SECTION>

EXAMPLES

The following example shows how to begin a statement section in a document part.

```
1 <PART>
  <PART_PAGE>
  <TITLE>(Part III\BASIC Statements and Functions)
  <ENDPART_PAGE>(RENUMBER)
  <STATEMENT_SECTION>(Statements and Functions\SF)
  <SET_TEMPLATE_STATEMENT>(BASIC_STATEMENT)

  <BASIC_STATEMENT>(GROUP)
  <OVERVIEW>
  Creates a group in a database.
  <ENDOVERVIEW>
  .
  .
  .
  <ENDSTATEMENT_SECTION>
```

The tags in the previous example perform the following functions:

- The global <PART> tag begins the part.
- The global <PART_PAGE> tag creates a part page.
- The global <TITLE> tag is used in the context of the <PART_PAGE> tag to create a title on the part page.
- The RENUMBER argument to the global <ENDPART_PAGE> tag specifies that the pages should be renumbered beginning with the part page. This causes the first page of text following the part page to be numbered page 3 (page 1 is the unnumbered page the part page title is placed on, page 2 is the back of page 1, and page 3 is the first numbered page after the part page).
- The <STATEMENT_SECTION> tag begins the statement section and specifies Statements and Functions as the running title for the statement section. If the <SET_TEMPLATE_STATEMENT> tag were used with the DOUBLERUNNINGHEADS argument, Statements and Functions would be used as the top running title.

The <STATEMENT_SECTION> tag also specifies that the prefix SF should be used to construct numbers for pages and for formal figures, tables, and examples in the statement section (for example, SF-11, SF-32, Table SF-1, Example SF-2, and so on).

- The <SET_TEMPLATE_STATEMENT> tag specifies that all statement descriptions in this statement section will be identified using the <BASIC_STATEMENT> tag rather than the default tags <FUNCTION> or <STATEMENT>. The <BASIC_STATEMENT> tag created by the <SET_TEMPLATE_STATEMENT> tag will have the default attributes of the default tags <FUNCTION> and <STATEMENT>.

The following example shows how you can create a statement section in which each statement description (begun with a <STATEMENT> or <FUNCTION> tag) is in a separate SDML file, and all these descriptions are included into a primary statement description file. For example, the file MYSTATEMENTS.SDML contains the following SDML tags:

SOFTWARE Doctype Tag Reference

<STATEMENT_SECTION>

```
<INCLUDE>(CLOSE_GROUP.SDML)
<INCLUDE>(OPEN_GROUP.SDML)
<INCLUDE>(READ_GROUP.SDML)
<INCLUDE>(WRITE_GROUP.SDML)
```

Each of the included files contains one statement reference description begun with a <STATEMENT> or <FUNCTION> tag. For these files to process correctly, they must be preceded with the <STATEMENT_SECTION> tag that enables the <STATEMENT> and <FUNCTION> tag. These files can have the necessary tags processed by specifying the /INCLUDE qualifier on the command line to include a startup definition file. This startup file might include the following tags:

```
2 <STATEMENT_SECTION>(Group Statements\GS)
   <SET_TEMPLATE_STATEMENT>(STATEMENT\DOUBLERUNNINGHEADS)
```

If this startup file were named GS_STATEMENT_STARTUP.SDML, it could be included using the DOCUMENT HEAD/INCLUDE qualifier as in the following example:

```
$ DOCUMENT mystatements SOFT.REF LN03-
_ $ /INCLUDE=GS_STATEMENT_STARTUP.SDML
```

When each individual file in MYSTATEMENTS.SDML is processed, the correct sequence of tags is read in to begin the statement section.

You can process multiple files together by using the <INCLUDE> tag to include them into a single master file (such as MYSTATEMENTS.SDML), or you can include them into a bookbuild profile.

Use the <ELEMENT> tags to include multiple files into a profile. For example, the bookbuild profile file GS_STATEPRO.SDML could contain the following tags:

```
<PROFILE>
<ELEMENT>(CLOSE_GROUP.SDML)
<ELEMENT>(OPEN_GROUP.SDML)
<ELEMENT>(READ_GROUP.SDML)
<ELEMENT>(WRITE_GROUP.SDML)
      <COMMENT>(**contains <ENDSTATEMENT_SECTION> tag**)
<ENDPROFILE>
```

Note that the PROFILE file should include the <ENDSTATEMENT_SECTION> tag in the appropriate file, so that the template will be terminated and the bookbuild will process correctly.

SOFTWARE Doctype Tag Reference

<SUBCOMMAND>

<SUBCOMMAND>

Begins a new subcommand description.

SYNTAX **<SUBCOMMAND>** (*subcommand name \ symbol name*)

ARGUMENTS ***subcommand name***

Names the command described in the subcommand section.

symbol name

Specifies the name of the symbol used in all references to the subcommand.

Symbol names must not exceed 31 characters and must only contain alphabetic letters, numbers, or underscores. Do not begin a symbol name with an underscore.

related tags

- <SET_TEMPLATE_SUBCOMMAND>
- <SUBCOMMAND_SECTION>

DESCRIPTION The <SUBCOMMAND> tag begins a new subcommand description. Subcommand sections have these format defaults:

- Each <SUBCOMMAND> tag begins a new page of output.
- The subcommand name appears as the single running title on each page.

To change the name of the <SUBCOMMAND> tag, or to change its default attributes, see the description of the <SET_TEMPLATE_SUBCOMMAND> tag.

EXAMPLES

In the following example, the <SUBCOMMAND_SECTION> tag enables the tags for a subcommand description. The description of the subcommand OPEN, by default, has the following attributes:

- 1 The subcommand description begins on a new page.
- 2 If the subcommand carries for more than a page, the name OPEN is carried as a running top title on each page.

I <SUBCOMMAND_SECTION>
 <SUBCOMMAND> (OPEN)
 <OVERVIEW>

·
·
·

SOFTWARE Doctype Tag Reference

<SUBCOMMAND>

In the following example, the <SUBCOMMAND_SECTION> tag starts a subcommand section and gives it the title File System Subcommands. In this command section, each page of output carries this title at the top of the page, with the name of the current subcommand just below it, until the <ENDSUBCOMMAND_SECTION> tag ends that titling.

```
2 <SUBCOMMAND_SECTION>(File System Subcommands)
  <SUBCOMMAND>(CLOSE)
```

<SUBCOMMAND_SECTION_HEAD>

Specifies the heading for text that precedes a subcommand section.

SYNTAX <SUBCOMMAND_SECTION_HEAD> (*heading*)

ARGUMENTS *heading*
Specifies a heading that precedes introductory text for the subcommand section.

related tags

- <SET_TEMPLATE_SUBCOMMAND>
- <SUBCOMMAND>

restrictions Valid only in the context of the <SUBCOMMAND_SECTION> tag.

DESCRIPTION The <SUBCOMMAND_SECTION_HEAD> tag specifies the heading for text that precedes a subcommand section. This tag lets you provide introductory text for a section of command descriptions that are subordinate to a specific command description.

EXAMPLE The following example illustrates a subcommand section in a command section. The subcommand section is used to describe subordinate commands.

```
<SUBCOMMAND_SECTION>(File System Subcommands\NEWPAGE)
<SUBCOMMAND_SECTION_HEAD>(Subcommand Descriptions)
<p>This section provides information about each of the subcommands
you can enter while you are conversing with the file subsystem.
<SUBCOMMAND>(CLOSE)
<overview>
.
.
.
```

SOFTWARE Doctype Tag Reference

<SYNTAX>

<SYNTAX>

Lets you use special characters to describe language syntaxes.

SYNTAX

<SYNTAX>*[({ heading text [\ WIDE] })]*
WIDE

ARGUMENTS

heading text

This is an optional argument. It specifies a heading. The doctype controls the font used to display the heading. By default, this tag has no heading. You may want to create a heading using the <SYNTAX_DEFAULT_HEAD> tag.

WIDE

This is an optional keyword argument. It specifies that the syntax statement can exceed the normal right margin of the text. If you are using doctype designs that indent the text body, a wide syntax example will extend into the left margin.

related tags

- <DISPLAY>
 - <SYNTAX_DEFAULT_HEAD>
 - The global <CODE_EXAMPLE> tag
 - The global <FORMAT> tag
-

restrictions

You cannot use tab characters, index tags (such as the <X> and <Y> tags), or text element tags (such as <P>, <LIST>, or <NOTE>) in this type of example.

required terminator

<ENDSYNTAX>

DESCRIPTION

The <SYNTAX> tag lets you use special characters to describe language syntaxes. Languages can include programming languages, command languages, application-defined languages, and so forth. This tag also separates the syntax example from the remaining text, retains blank spaces and open lines, and labels the example (if you specified one) using a doctype-specific font different from the current text font.

EXAMPLE

The following example shows how to use the <SYNTAX> tag to describe a language's syntax.

```
<P>The COPY command has the following syntax:  
<SYNTAX>  
COPY input_file output_file  
<ENDSYNTAX>
```

This example produces the following output.

The COPY command has the following syntax:

```
COPY input_file output_file
```

SOFTWARE Doctype Tag Reference

<SYNTAX_DEFAULT_HEAD>

<SYNTAX_DEFAULT_HEAD>

Creates a default heading for the <SYNTAX> tag.

SYNTAX

<SYNTAX_DEFAULT_HEAD>({ *heading text* }
 OFF)

ARGUMENTS

heading text

Specifies a heading to be used by all subsequent <SYNTAX> tags. The doctype controls the font used to display this heading. By default, the <SYNTAX> tag has no heading.

OFF

Disables any heading established by a previous use of the <SYNTAX_DEFAULT_HEAD> tag.

related tags

- <SYNTAX>

DESCRIPTION

The <SYNTAX_DEFAULT_HEAD> tag creates a default heading for the <SYNTAX> tag. Use the <SYNTAX_DEFAULT_HEAD> tag to create the same default heading above each use of the <SYNTAX> tag.

To disable all subsequent default headings, specify the OFF argument to the <SYNTAX_DEFAULT_HEAD> tag.

EXAMPLES

The following example shows how to use the <SYNTAX_DEFAULT_HEAD> tag to create a default heading for all subsequent <SYNTAX> tags.

```
1 <P>
  The COPY command has the following general syntax:
  <SYNTAX>
      COPY input-file output-file
  <ENDSYNTAX>

  <COMMENT>(Set up default headings for syntax statements....)
  <SYNTAX_DEFAULT_HEAD>(What the User Types)
  <P>
  An actual user would type the following:
  <SYNTAX>
  $ COPY MYFILE.TXT NEWFILE.TXT
  <ENDSYNTAX>
```

This example produces the following output:

The COPY command has the following general syntax:

```
COPY input-file output-file
```

SOFTWARE Doctype Tag Reference

<SYNTAX_DEFAULT_HEAD>

An actual user would type the following:

What the User Types

```
$ COPY MYFILE.TXT NEWFILE.TXT
```

The following example shows how to disable the <SYNTAX_DEFAULT_HEAD> tag for all subsequent <SYNTAX> tags.

```
2 <COMMENT>(Set up default headings for syntax statements....)
<SYNTAX_DEFAULT_HEAD>(What the User Types)
<P>
An actual user would type the following:
<SYNTAX>
$ COPY MYFILE.TXT NEWFILE.TXT
<ENDSYNTAX>

<COMMENT>(Disable default headings for syntax statements....)
<SYNTAX_DEFAULT_HEAD>(OFF)

<P>The following is a semantic statement of the COPY operation.
<SYNTAX>
COPY [the existing file specification to] [the new file specification]
<ENDSYNTAX>
```

This example produces the following output:

An actual user would type the following:

What the User Types

```
$ COPY MYFILE.TXT NEWFILE.TXT
```

The following is a semantic statement of the COPY operation.

```
COPY [the existing file specification to] [the new file specification]
```

SOFTWARE Doctype Tag Reference

<TAG_SECTION>

<TAG_SECTION>

Begins a tag reference section, enables tags reserved for use in tag sections, and sets paging attributes.

SYNTAX

<TAG_SECTION>*[[*running title
[\ number-prefix]
[\ NEWPAGE]]

ARGUMENTS

running title

This is an optional argument. It specifies a top-level running heading to be used throughout the tag section. If this argument is not specified, the running headings are determined as described in Using the Template-Enabling Tags (see Section 10.13).

number prefix

This is an optional argument. It specifies a character-string prefix to be used to construct page numbers (folios) and formal figure, table, and example numbers. If this argument is not specified, the page and formal element numbering are determined as described in Using the Template-Enabling Tags (see Section 10.13).

NEWPAGE

This is an optional keyword argument. It indicates that the tag section should begin on a new page. This argument is only meaningful in two cases:

- When you have previously entered the <SET_TEMPLATE_TAG> tag with the NONEWPAGE keyword to specify that each new tag in this tag section should not begin on a new page.
- When you want to place one or more pages of text between the end of a part page and the beginning of a tag section.

related tags

- <DESCRIPTION>
- <EXAMPLE_SEQUENCE>
- <FORMAT>
- <FTAG>
- <OVERVIEW>
- <PARAMDEFLIST>
- <RELATED_ITEM>
- <RELATED_TAG>
- <RELATED_TAGS>
- <RESTRICTIONS>

SOFTWARE Doctype Tag Reference

<TAG_SECTION>

- <RITEM>
- <SDML_TAG>
- <SET_TEMPLATE_HEADING>
- <SET_TEMPLATE_LIST>
- <SET_TEMPLATE_PARA>
- <SET_TEMPLATE_ROUTINE>
- <SET_TEMPLATE_TABLE>
- <SET_TEMPLATE_TAG>
- <TERMINATING_TAG>

**required
terminator**

<ENDTAG_SECTION>

DESCRIPTION

The <TAG_SECTION> tag begins a tag reference section, enables tags reserved for use in tag sections, and sets paging attributes. You can locate a tag section in a chapter or an appendix, or following a part page (that is, in a document section begun with the <PART_PAGE> tag). You code a tag section in a chapter or an appendix in the same manner; tag sections in parts are handled differently.

If your tag section follows a part page, and you include text between the part page and the tag section, specify the NEWPAGE keyword as the third argument to the <TAG_SECTION> tag. This causes the tag section to begin on a new page. The following code fragment shows a tag section that begins on a new page:

```
<TAG_SECTION>(\TD\NEWPAGE)
<HEAD1>(Tag Dictionary\46_TagDictionary)
```

When you use the <TAG_SECTION> tag in a chapter or an appendix, and want to place text after the tag section in that chapter or appendix, you must end the tag section with the <ENDTAG_SECTION> tag and place the text after that tag. By default, this text begins on a new page of output.

Specify the NONENPAGE argument to the <ENDTAG_SECTION> tag if you do not want the text to begin on a new page of output. The following code fragment shows the end of a tag section that specifies that the subsequent text not be placed on a new page:

```
<ENDTAG_SECTION>(NONENPAGE)
```

When the <ENDTAG_SECTION> tag is specified in the context of a chapter or appendix, it resets the default running titles to those in effect for the chapter or appendix, so the last page of the last tag description in the tag section may not carry the last tag's name as the running heading. Instead it may carry the running title used by the chapter or appendix.

SOFTWARE Doctype Tag Reference

<TAG_SECTION>

EXAMPLES

The following example shows how to begin a tag section in a document part.

```
❶ <PART>
  <PART_PAGE>
  <TITLE>(Part III\Tag Dictionary)
  <ENDPART_PAGE>(RENUMBER)
  <TAG_SECTION>(Tag Dictionary\TD)
  <SET_TEMPLATE_TAG>(LOCAL_TAG)

  <LOCAL_TAG>(SITETAG)

  <OVERVIEW>
  This is a site-specific tag.
  <ENDOVERVIEW>
  .
  .
  .
  <ENDTAG_SECTION>
```

The tags in the previous example perform the following functions:

- The global <PART> tag begins the part.
- The global <PART_PAGE> tag creates a part page.
- The global <TITLE> tag is used in the context of the <PART_PAGE> tag to create a title on the part page.
- The RENUMBER argument to the global <ENDPART_PAGE> tag specifies that the pages should be renumbered beginning with the part page. This causes the first page of text following the part page to be numbered page 3 (page 1 is the unnumbered page the part page title is placed on, page 2 is the back of page 1, and page 3 is the first numbered page after the part page).
- The <TAG_SECTION> tag begins the tag section and specifies the running title Tag Dictionary as the running title for the tag section. If the <SET_TEMPLATE_TAG> tag were used with the DOUBLERUNNINGHEADS argument, the title Tag Dictionary would be used as the top running title.

The <TAG_SECTION> tag also specifies that the prefix TD should be used to construct numbers for pages and for formal figures, tables, and examples in the tag section (for example, TD-11, TD-32, Table TD-1, Example TD-2, and so on).

- The <SET_TEMPLATE_TAG> tag specifies that all tag descriptions in this tag section will be identified using the <LOCAL_TAG> tag rather than the default <TAG> tag. The <LOCAL_TAG> tag will have the default attributes of the <TAG> tag.

The following example shows how you can create a tag section in which each tag description (begun with an <SDML_TAG> tag) is in a separate SDML file, and all these descriptions are included into a primary routine description file. For example, the file MYTAGS.SDML contains the following SDML tags:

SOFTWARE Doctype Tag Reference

<TAG_SECTION>

```
<INCLUDE>(CLOSE_FILE.SDML)
<INCLUDE>(OPEN_FILE.SDML)
<INCLUDE>(READ_FILE.SDML)
<INCLUDE>(WRITE_FILE.SDML)
```

Each of the included files contains one tag reference description begun with an <SDML_TAG> tag. For these files to process correctly, they must be preceded with the <TAG_SECTION> tag that enables the <SDML_TAG> tag. These files can have the necessary tags processed before them by specifying the /INCLUDE qualifier on the command line to include a startup definition file. This startup file might include the following tags.

2 <TAG_SECTION>(File Handling Tags\TAGS)
<SET_TEMPLATE_TAG>(SDML_TAG\DOUBLERUNNINGHEADS)

If this startup file were named FILE_TAG_STARTUP.SDML, it could be included using the DOCUMENT /INCLUDE qualifier as in the following example:

```
$ DOCUMENT mytags SOFT.REF LN03 /INCLUDE=FILE_TAG_STARTUP.SDML
```

When each individual file in MYTAGS.SDML is processed, the correct sequence of tags will be read in to begin the tag section.

You can process multiple files together by using the <INCLUDE> tag to include them into a single master file (such as MYTAGS.SDML), or you can include them into a bookbuild profile.

You use the <ELEMENT> tags to include multiple files into a profile. For example, the bookbuild profile file TAGPRO.SDML could contain the following tags:

```
<PROFILE>
<ELEMENT>(CLOSE_FILE.SDML)
<ELEMENT>(OPEN_FILE.SDML)
<ELEMENT>(READ_FILE.SDML)
<ELEMENT>(WRITE_FILE.SDML) <COMMENT>(contains <ENDTAG_SECTION> tag)
<ENDPROFILE>
```

Note that the PROFILE file should include the <ENDTAG_SECTION> tag in the appropriate file, so that the template will be terminated and the bookbuild will process correctly.

SOFTWARE Doctype Tag Reference

<TERMINATING_TAG>

<TERMINATING_TAG>

Specifies the required terminator for a tag.

SYNTAX

<TERMINATING_TAG>({ *tag name*[\ *additional text*] })
NONE

ARGUMENTS

tag name

Specifies the name of the terminating tag.

additional text

This is an optional argument. It specifies additional text you can insert to briefly explain the terminating tag.

NONE

Indicates that there is no terminating tag.

related tags

- <RELATED_TAG>
 - <TAG_SECTION>
-

restrictions

Valid only in the context of the <TAG_SECTION> tag.

DESCRIPTION

The <TERMINATING_TAG> tag specifies the required terminator for a tag. Provide additional information about the terminating tag by specifying this text as the second argument to the <TERMINATING_TAG> tag.

Use the NONE keyword argument to explicitly specify that no terminating tag is needed. This keyword places the text None beneath the heading output by this tag.

EXAMPLES

The following example shows a terminating tag specified with both the *tag name* argument and the *additional text* argument.

1 <TERMINATING_TAG>(ENDRECORDLIST\
<p>
Omit this tag if you use the
NONE keyword with the <TAG>(RECORDLIST) tag.)

This example produces the following output:

required terminator

<ENDRECORDLIST>

Omit this tag if you use the NONE keyword with the <RECORDLIST> tag.

The following example shows the <TERMINATING_TAG> tag used with the NONE keyword.

SOFTWARE Doctype Tag Reference <TERMINATING_TAG>

2 <TERMINATING_TAG> (NONE)

This example produces the following output:

**required
terminator**

None.

Index

A

<ABSTRACT> • 2-2, 2-4, 2-17

description of • 2-17

related tags

<TITLE_SECTION>

<ACKNOWLEDGMENTS> • 2-2, 2-5, 2-18

description of • 2-18

related tags

<BACK_NOTES>

<REF_NOTES>

<ARGDEF> • 10-65

description of • 10-65

related tags

<ARGDEFLIST>

<ARGITEM>

<ARGDEFLIST> • 10-66

description of • 10-66

related tags

<ARGDEF>

<ARGITEM>

<ARGTEXT>

<PARAMDEFLIST>

<QUALDEFLIST>

<SET_TEMPLATE__ARGITEM>

<SET_TEMPLATE_HEADING>

The global <DEFINITION_LIST> tag

<ARGITEM> • 10-69

description of • 10-69

related tags

<ARGDEF>

<ARGDEFLIST>

<ARGTEXT>

<ARGTEXT> • 10-71

description of • 10-71

related tags

<ARGDEF>

<ARGDEFLIST>

<ARGITEM>

<ARGUMENT> • 10-73

description of • 10-73

related tags

<KEYWORD>

<ARGUMENT>

related tags (cont'd)

<VARIABLE>

<ARG_SEP> • 10-74

See also <FTAG>

description of • 10-74

related tags

<FTAG>

ARTICLE doctype • 2-1

abstracts • 2-4

acknowledgments • 2-5

author information • 2-3

back notes • 2-6, 2-7

bibliographies • 2-8

headings • 2-5

improving format of

See Two-column doctype designs • 2-8

page layout of • 2-1

quotations • 2-6

reference notes • 2-6, 2-7

running headings • 2-5

running titles • 2-5

sample SDML file • 2-12

source notes • 2-4

subtitles • 2-3

tags • 2-16

titles • 2-3

<AUTHOR> • 2-2, 2-3, 2-19, 2-20, 9-11, 10-75

description of • 2-19, 2-20, 9-11, 10-75

in ARTICLE doctype • 2-19

in REPORT doctype • 9-3, 9-11

in SOFTWARE doctype • 10-75

related tags

<AUTHOR_ADDR>

<AUTHOR_AFF>

<AUTHOR_LIST>

<BYLINE>

<SIGNATURES>

<SOURCE_NOTE>

<VITA>

<AUTHOR_ADDR> • 2-2, 2-3

description of • 2-20

related tags

<AUTHOR>

<AUTHOR_AFF>

<AUTHOR_LIST>

Index

- <AUTHOR_ADDR>
related tags (cont'd)
 <VITA>
- <AUTHOR_AFF> • 2-2, 2-3, 2-21
description of • 2-21
related tags
 <AUTHOR>
 <AUTHOR_ADDR>
 <AUTHOR_LIST>
 <VITA>
- <AUTHOR_INFO> • 8-2, 8-8
description of • 8-8
related tags
 <INTRO_SUBTITLE>
 <INTRO_TITLE>
- <AUTHOR_LIST> • 2-2, 2-3, 2-22
description of • 2-22
related tags
 <AUTHOR>
 <AUTHOR_ADDR>
 <AUTHOR_AFF>
 <VITA>
- <AUTO_NUMBER> • 8-2, 8-9
description of • 8-9
related tags
 <RUNNING_FEET>
 <SLIDE>
- <BIB_ENTRY> • 2-2, 2-8, 2-27
description of • 2-27
related tags
 <BIBLIOGRAPHY>
- Bookreader • 7-1
- <BOOK_ONLY> • 3-5, 7-2
description of • 3-5, 7-2
related tags
 <ENDBOOK_ONLY>
 <HELP_ONLY>
 <HELP_ONLY>
 <KEEP_HELP_LEVEL>
 <SET_HELP_LEVEL>
 <SET_HELP_LEVEL>
- <BOOK_REF> • 7-3
description of • 7-3
related tags
 <SHELF_CREATE>
 <SHELF_REF>
- <BYLINE> • 9-13, 10-77
description of • 9-13, 10-77
in REPORT doctype • 9-3, 9-13
related tags
 <AUTHOR>
 <SIGNATURES>
 The global <FRONT_MATTER>

B

- Back notes
 in ARTICLE doctype • 2-6
- <BACK_NOTE> • 2-2, 2-23
description of • 2-23
related tags
 <BACK_NOTES>
 <REF_NOTE>
- <BACK_NOTES> • 2-2, 2-25
description of • 2-25
related tags
 <BACK_NOTE>
- <BIBLIOGRAPHY> • 2-2, 2-8, 2-26
description of • 2-26
related tags
 <BIB_ENTRY>
 <REF_NOTE>
 <REF_NOTES>

C

- <CC> • 4-2, 4-9
description of • 4-9
related tags
 <CCLIST>
- <CCLIST> • 4-2, 4-11
description of • 4-11
related tags
 <CC>
 <DISTLIST>
- <CLOSING> • 4-2, 4-12
description of • 4-12
related tags
 <FROM_ADDRESS>
 <MEMO_FROM>
 <MEMO_TO>
 <SALUTATION>
 <TO_ADDRESS>

Code fragments

See SOFTWARE doctype

<CODE_EXAMPLE> • 6–18

definition of • 6–19

description of • 6–18, 6–19

related tags

the global <INTERACTIVE> tag

the global <LINE_ART> tag

the global <VALID_BREAK> tag

<COLUMN> • 2–2, 2–28, 9–15

description of • 2–28, 9–15

in REPORT doctype • 9–3, 9–15

related tags

the global tag <BIBLIOGRAPHY>

<COMMAND> • 10–79

description of • 10–79

related tags

<COMMAND_SECTION>

<SET_TEMPLATE_COMMAND>

Command template

See SOFTWARE doctype

<COMMAND_SECTION> • 10–81

description of • 10–81, 10–82

related tags

<COMMAND>

<DESCRIPTION>

<EXAMPLE_SEQUENCE>

<FCMD>

<FORMAT>

<FPARM>

<FPARMS>

<OVERVIEW>

<PARAMDEFLIST>

<PROMPTS>

<QUALDEFLIST>

<RESTRICTIONS>

<SET_TEMPLATE_COMMAND>

<SET_TEMPLATE_HEADING>

<SET_TEMPLATE_LIST>

<SET_TEMPLATE_PARA>

<SET_TEMPLATE_TABLE>

<STATEMENT_LINE>

<CONSTRUCT> • 10–85

description of • 10–85

related tags

<CONSTRUCT_LIST>

<STATEMENT_LINE>

<CONSTRUCT_LIST> • 10–87

description of • 10–87

<CPOS> • 10–90

description of • 10–90

related tags

<KEY>

<KEY_SEQUENCE>

Creating reference templates

in SOFTWARE doctype • 10–27

Creating template tables

in SOFTWARE doctype • 10–28

D

Data Item Description documents (DID)

See MILSPEC Doctype

Default online topic • 7–30

<DELETE_KEY> • 10–91

description of • 10–91

related tags

<GRAPHIC>

<KEY>

<DESCRIPTION> • 10–92

description of • 10–92

related tags

<OVERVIEW>

<SET_TEMPLATE_HEADING>

<THE GLOBAL <INTERACTIVE> TAG>

<THE GLOBAL <VALID_BREAK> TAG>

DID documents

See MILSPEC Doctype

<DISPLAY> • 10–94

description of • 10–94

related tags

<SYNTAX>

<DISTLIST> • 4–2, 4–13

description of • 4–13

Doctype

ARTICLE • 2–1

LETTER • 4–1

list of • 1–1

MANUAL • 5–1

MILSPEC • 6–1

ONLINE • 7–1

OVERHEADS • 8–1

REPORT • 9–1

SOFTWARE • 10–1

Index

Doctype

two-column

See Two-column doctype

using • 1–2

Doctype(xs)HELP • 3–1

Doctype-specific tags • 1–1

using • 1–2

<DOCUMENT_ATTRIBUTES> • 2–30, 6–20, 9–17, 10–96

description of • 2–30, 6–20, 6–22, 9–17, 10–96

in ARTICLE doctype • 2–2, 2–5

in MILSPEC.SECURITY doctype • 6–22

in MILSPEC doctypes • 6–20

in REPORT doctype • 9–3, 9–17

in SOFTWARE doctype • 10–96

related tags

<RUNNING_FEET>

<RUNNING_TITLE>

<EXI> • 10–102

description of • 10–102

related tags

<EXAMPLE_SEQUENCE>

<EXTENSION> • 7–4

description of • 7–4

related tag

<ENDEXTENSION>

<EXTEXT> • 10–104

description of • 10–104

related tags

<EXAMPLE_INTRO>

<EXC>

<EXI>

Extracts

See <QUOTATION>

E

<ENDBOOK_ONLY> • 3–5

<ENDHELP_ONLY> • 3–6, 7–9

<ENDKEEP_HELP_LEVEL> • 3–7, 7–10

End notes

See <BACK_NOTES>

in ARTICLE doctype • 2–6

<EXAMPLES_INTRO> • 10–100

description of • 10–100

<EXAMPLE_SEQUENCE> • 10–98

description of • 10–98

related tags

<EXAMPLES_INTRO>

<EXAMPLE_SEQUENCE>

<EXC>

<EXI>

<EXTEXT>

<EXC> • 10–101

description of • 10–101

related tags

<EXAMPLES_INTRO>

<EXAMPLE_SEQUENCE>

<EXC>

<EXI>

<EXTEXT>

The global <CODE_EXAMPLE> tag

The global <INTERACTIVE> tag

The global <S> tag

The global <U> tag

F

<FARG> • 10–105

description of • 10–105

related tags

<FARGS>

<FFUNC>

<FORMAT>

<FRTN>

<ROUTINE_SECTION>

<FARGS> • 10–107

description of • 10–107

related tags

<FARG>

<FARGS>

<FRTN>

<FCMD> • 10–109

description • 10–109

in Statement template • 10–109

related tags

<COMMAND_SECTION>

<FORMAT>

<FPARMS>

<STATEMENT_FORMAT>

<FFUNC> • 10–112

description of • 10–112

in Statement template • 10–112

related tags

<STATEMENT_FORMAT>

Footers
 in SOFTWARE.SPECIFICATION doctype • 10–33
 <FORMAT> • 10–114
 description of • 10–114
 in Routine template • 10–114
 related tags
 <FARG>
 <FARGS>
 <FFUNC>
 <FRTN>
 <FORMAT_SUBHEAD> • 10–116
 description of • 10–116
 related tags
 <FCMD>
 <FPARMS>
 <STATEMENT_FORMAT>
 <FPARM> • 10–117
 description of • 10–117
 in Statement template • 10–117
 related tags
 <FCMD>
 <FPARMS>
 <STATEMENT_FORMAT>
 <STATEMENT_SECTION>
 <FPARMS> • 10–119
 description of • 10–119
 in Command and Statement templates • 10–119
 related tags
 <COMMAND_SECTION>
 <FCMD>
 <FORMAT>
 <FPARM>
 <STATEMENT_FORMAT>
 <FROM_ADDRESS> • 4–2, 4–14
 description of • 4–14
 related tags
 <MEMO_FROM>
 <TO_ADDRESS>
 <FRTN> • 10–121
 description of • 10–121
 related tags
 <FARG>
 <FARGS>
 <FFUNC>
 <FTAG> • 10–123
 description of • 10–123
 related tags
 <ARG_SEP>
 <FORMAT>

<FUNCTION> • 10–125
 description of • 10–125
 related tags
 <SET_TEMPLATE_STATEMENT>
 <STATEMENT>
 <STATEMENT_SECTION>

G

<GRAPHIC> • 10–126
 description of • 10–126
 related tags
 <KEY>

H

Headers
 in SOFTWARE.SPECIFICATION doctype • 10–33
 <HEADN> • 6–23
 description of • 6–23
 related tags
 the global <CHEAD> tag
 the global <SUBHEAD1> tag
 the global <SUBHEAD2> tag
HELP
 using • 3–1
HELP doctype • 3–1
 tags • 3–4
 <HELP_ONLY> • 3–6, 7–9
 description of • 3–6, 7–9
 related tags
 <BOOK_ONLY>
 <BOOK_ONLY>
 <ENDHELP_ONLY>
 <KEEP_HELP_LEVEL>
 <SET_HELP_LEVEL>
 <SET_HELP_LEVEL>
 <SET_HELP_ONLY>
 <HIGHEST_SECURITY_CLASS> • 6–25
 description of • 6–25
 related tags
 <SECURITY>
 <SET_PAGE_SECURITY>
 <HOTSPOT> • 7–6
 description of • 7–6

Index

<HOTSPOT> (cont'd)

related tags

<REFERENCE>

<SET_ONLINE_TOPIC>

Hotspots

using the <ONLINE_POPUP> tag • 7–25

I

Informal online topic

popping it up • 7–24

<INTRO_SUBTITLE> • 8–2, 8–10

description of • 8–10

related tags

<INTRO_TITLE>

<SUBTITLE>

<INTRO_TITLE> • 8–2, 8–11

description of • 8–11

related tags

<INTRO_SUBTITLE>

<SLIDE>

<TITLE>

K

<KEEP_HELP_LEVEL> • 3–7, 7–10

description of • 3–7, 7–10

related tags

<BOOK_ONLY>

<ENDBOOK_ONLY>

<ENDHELP_ONLY>

<ENDKEEP_HELP_LEVEL>

<HELP_ONLY>

<KEEP_HELP_LEVEL>

<SET_HELP_LEVEL>

<KEY> • 10–127

description of • 10–127

related tags

<GRAPHIC>

<KEY_NAME>

<KEY_PLUS>

<KEY_SEQUENCE>

<KEYPAD> • 10–129

description of • 10–129

<KEYPAD>

related tags

<KEYPAD_ENDROW>

<KEYPAD_ROW>

<KEYPAD_SECTION>

<KEYPAD_ENDROW> • 10–132

description of • 10–132

related tags

<KEYPAD>

<KEYPAD_ROW>

<KEYPAD_SECTION>

<KEYPAD_ROW> • 10–133

description of • 10–133

related tags

<KEYPAD>

<KEYPAD_ENDROW>

<KEYPAD_SECTION>

<KEYPAD_SECTION> • 10–134

description of • 10–134

related tags

<KEYPAD>

<KEYPAD_ENDROW>

<KEYPAD_ROW>

<KEY_NAME> • 10–137

description of • 10–137

related tags

<CPOS>

<GRAPHIC>

<KEY>

<KEY_SEQUENCE>

<KEY_PLUS> • 10–138

description of • 10–138

related tags

<KEY>

<KEY_SEQUENCE>

<KEY_TYPE>

<KEY_SEQUENCE> • 10–139

description of • 10–139

related tags

<CPOS>

<GRAPHIC>

<KEY>

<KEY_PLUS>

<KEY_TYPE>

<KEY_TYPE> • 10–141

description of • 10–141

related tags

<GRAPHIC>

<KEY_PLUS>

<KEY_TYPE>

related tags (cont'd)

<KEY_SEQUENCE>

L

Language extensions • 7-4

Layout, page

of ARTICLE doctype design • 2-1

of LETTER doctype design • 4-1

of MANUAL doctype

MANUAL.GUIDE design • 5-2

MANUAL.PRIMER design • 5-2

MANUAL.REFERENCE design • 5-2

of MILSPEC doctype • 6-2

of OVERHEADS.35MM doctype design • 8-2

of OVERHEADS doctype design • 8-1

of REPORT.TWOCOL doctype • 9-2

of REPORT doctype • 9-1

of SOFTWARE.BROCHURE doctype • 10-3

of SOFTWARE.GUIDE doctype • 10-3

of SOFTWARE.HANDBOOK doctype • 10-4

of SOFTWARE.POCKET_REFERENCE doctype •
10-4

of SOFTWARE.REFERENCE doctype • 10-4

of SOFTWARE.SPECIFICATION doctype • 10-5

LETTER doctype • 4-1

characteristics of • 4-1

page layout of • 4-1

sample SDML file

to create a letter • 4-6

to create a memo • 4-4

tags • 4-8

<LEVEL> • 9-3, 9-19

description of • 9-19

related tags

<OUTLINE>

<SHOW_LEVELS>

License Management Facility • 7-12, 7-15, 7-16,
7-18, 7-19, 7-20, 7-21

<LMF> • 7-12

description of • 7-12

related tags

<LMF_ALTNAME>

<LMF_INFO>

<LMF_PRODUCER>

<LMF_PRODUCT>

<LMF_RELEASE_DATE>

<LMF_VERSION_NUMBER>

<LMF_ALTNAME> • 7-15

description of • 7-15

related tags

<ENDLMF>

<LMF>

<LMF_INFO>

<LMF_PRODUCER>

<LMF_PRODUCT>

<LMF_RELEASE_DATE>

<LMF_VERSION_NUMBER>

<LMF_INFO> • 7-16

description of • 7-16

related tags

<LMF>

<LMF_ALTNAME>

<LMF_PRODUCER>

<LMF_PRODUCT>

<LMF_RELEASE_DATE>

<LMF_VERSION_NUMBER>

<LMF_PRODUCER> • 7-18

description of • 7-18

related tags

<ENDLMF>

<LMF>

<LMF_ALTNAME>

<LMF_INFO>

<LMF_PRODUCT>

<LMF_RELEASE_DATE>

<LMF_VERSION_NUMBER>

<LMF_PRODUCT> • 7-19

description of • 7-19

related tags

<ENDLMF>

<LMF>

<LMF_ALTNAME>

<LMF_INFO>

<LMF_PRODUCER>

<LMF_RELEASE_DATE>

<LMF_VERSION_NUMBER>

<LMF_RELEASE_DATE> • 7-20

description of • 7-20

related tags

<ENDLMF>

<LMF>

<LMF_ALTNAME>

<LMF_INFO>

<LMF_PRODUCER>

<LMF_PRODUCT>

Index

<LMF_RELEASE_DATE>
 related tags (cont'd)
 <LMF_VERSION_NUMBER>
<LMF_VERSION_NUMBER> • 7–21
 description of • 7–21
 related tags
 <LMF>
 <LMF_ALTNAME>
 <LMF_INFO>
 <LMF_PRODUCER>
 <LMF_PRODUCT>
 <LMF_RELEASE_DATE>
 <LMF_VERSION_NUMBER>

M

MANUAL doctype • 5–1, 5–4
 page layout of • 5–2
 sample input file • 5–4
 sample output file • 5–4
 sample SDML file • 5–4

Memo

 See LETTER doctype

<MEMO_DATE> • 4–2, 4–15
 description of • 4–15
 related tags
 <FROM_ADDRESS>
 <MEMO_LINE>
 <MEMO_TO>
 The global <DATE> tag
<MEMO_FROM> • 4–2, 4–17
 description of • 4–17
 related tags
 <FROM_ADDRESS>
 <MEMO_TO>
<MEMO_HEADER> • 4–2, 4–18
 description of • 4–18
 related tags
 <MEMO_FROM>
 <MEMO_TO>
<MEMO_LINE> • 4–2, 4–19
 description of • 4–19
 related tags
 <FROM_ADDRESS>
 <MEMO_DATE>
 <MEMO_FROM>
 <MEMO_TO>

<MEMO_TO> • 4–3, 4–21
 description of • 4–21
 related tags
 <MEMO_FROM>
 <TO_ADDRESS>

Messages

 See SOFTWARE doctype

<MESSAGE_FACILITY>
 related tags
 <MESSAGE_SECTION>
 <MSG>
 <MSGGS>
 <MSG_ACTION>
 <MSG_TYPE>
<MESSAGE_SECTION> • 10–142
 description of • 10–142
 related tags
 <MESSAGE_TYPE>
 <MSG>
 <MSGGS>
 <MSG_ACTION>
 <MSG_FACILITY>
 <MSG_SEVERITY>
 <MSG_TEXT>
<MESSAGE_SEVERITY>
 related tags
 <MESSAGE_SECTION>
 <MESSAGE_TYPE>
 <MSG>
 <MSGGS>
 <MSG_ACTION>
 <MSG_FACILITY>
 <MSG_TEXT>
<MESSAGE_TYPE> • 10–145
 description of • 10–145
 related tags
 <MESSAGE_SECTION>
 <MSG>
 <MSGGS>
 <MSG_TEXT>

Military specifications

 See MILSPEC Doctype
MILSPEC.DRAFT doctype
 sample output file • 6–5
 sample SDML file • 6–4, 6–5
MILSPEC.SECURITY doctype
 sample output file • 6–5
 sample SDML file • 6–4, 6–5

MILSPEC doctype • 6–1, 6–45
 DOD-STD-2167 documents • 6–12, 6–16
 MIL-STD-490A documents • 6–11, 6–12
 page layout of • 6–2
 tags • 6–17, 6–45
 templates • 6–15

MIL-STD-490A documents
 See MILSPEC doctype

Modifying reference templates
 in SOFTWARE doctype • 10–30

Modifying template defaults
 in SOFTWARE doctype • 10–30

<MSG> • 10–146
 description of • 10–146
 related tags
 <MESSAGE_SECTION>
 <MESSAGE_TYPE>
 <MSGS>
 <MSG_TEXT>

<MSGS> • 10–148
 description of • 10–148
 related tags
 <MESSAGE_SECTION>
 <MESSAGE_TYPE>
 <MSG>
 <MSG_TEXT>

MSG_ACTION • 10–150

<MSG_ACTION> • 10–150
 description of • 10–150
 related tags
 <MESSAGE_SECTION>
 <MESSAGE_TYPE>
 <MSG>
 <MSGS>
 <MSG_FACILITY>
 <MSG_SEVERITY>
 <MSG_TEXT>

MSG_FACILITY • 10–151

<MSG_FACILITY> • 10–151
 description of • 10–151

MSG_SEVERITY • 10–152

<MSG_SEVERITY> • 10–152
 description of • 10–152

<MSG_TEXT> • 10–153
 description of • 10–153

N

Notes

back notes • 2–6
 See <BACK_NOTES>
 end notes • 2–6
 footnotes • 2–6
 reference notes • 2–6
 See <REF_NOTES>
 source notes
 See <SOURCE_NOTE>

O

ONLINE doctype • 7–1
 tags • 7–1

Online topic • 7–30

<ONLINE_CHUNK> • 7–22
 description of • 7–22

<ONLINE_CHUNK>
 related tags
 <ONLINE_POPUP>
 <ONLINE_TITLE>

<ONLINE_POPUP> • 7–24
 description of • 7–24
 related tags
 <ONLINE_CHUNK>
 <ONLINE_TITLE>

<ONLINE_TITLE> • 7–26
 description of • 7–26
 related tags
 <ONLINE_CHUNK>
 <ONLINE_POPUP>

<OUTLINE> • 9–3, 9–20
 description of • 9–20
 related tags
 <LEVEL>
 <SHOW_LEVELS>

OVERHEADS.35MM doctype
 page layout of • 8–2

OVERHEADS doctype • 8–1
 page layout of • 8–1
 sample SDML file • 8–3
 tags • 8–7

Overhead slide
 See OVERHEADS doctype

Index

<OVERVIEW> • 10–154
description of • 10–154
related tags
 <DESCRIPTION>

P

Page layout

of ARTICLE doctype design • 2–1
of LETTER doctype design • 4–1
of MANUAL doctype
 MANUAL.GUIDE design • 5–2
 MANUAL.PRIMER design • 5–2
 MANUAL.REFERENCE design • 5–2
of MILSPEC doctype • 6–2
of OVERHEADS.35MM doctype design • 8–2
of OVERHEADS doctype design • 8–1
of REPORT.TWOCOL doctype • 9–2
of REPORT doctype • 9–1
of SOFTWARE.BROCHURE doctype • 10–3
of SOFTWARE.GUIDE doctype • 10–3
of SOFTWARE.HANDBOOK doctype • 10–4
of SOFTWARE.POCKET_REFERENCE doctype •
 10–4
of SOFTWARE.REFERENCE doctype • 10–4
of SOFTWARE.SPECIFICATION doctype • 10–5

<PARAMDEF> • 10–155
description of • 10–155
related tags
 <PARAMDEFLIST>
 <PARAMITEM>

<PARAMDEFLIST> • 10–156
description of • 10–156
related tags
 <ARGDEFLIST>
 <PARAMDEF>
 <PARAMITEM>
 <QUALDEFLIST>
 <SET_TEMPLATE_HEADING>

<PARAMITEM> • 10–159
description of • 10–159
related tags
 <PARAMDEF>
 <PARAMDEFLIST>

Pop-up • 7–24

<PROMPT> • 10–161
description of • 10–161

<PROMPT>
related tags
 <PROMPTS>

<PROMPTS> • 10–163
description of • 10–163
related tags
 <COMMAND_SECTION>
 <PROMPT>
 <SET_TEMPLATE_HEADING>

Q

<QPAIR> • 10–165
description of • 10–165
related tags
 <QUAL_LIST>
 <QUAL_LIST_HEADS>

<QUALDEF> • 10–166
description of • 10–166
related tags
 <QUALDEFLIST>
 <QUALITEM>

<QUALDEFLIST> • 10–167
description of • 10–167
related tags
 <ARGDEFLIST>
 <PARAMDEFLIST>
 <QUALDEF>
 <QUALITEM>
 <SET_TEMPLATE_HEADING>

<QUALITEM> • 10–169
description of • 10–169
related tags
 <QUALDEF>
 <QUALDEFLIST>

<QUAL_LIST> • 10–171
description of • 10–171
related tags
 <QPAIR>
 <QUAL_LIST_DEFAULT_HEADS>
 <QUAL_LIST_HEADS>

<QUAL_LIST_DEFAULT_HEADS> • 10–175
description of • 10–175
related tags
 <QPAIR>
 <QUAL_LIST>
 <QUAL_LIST_HEADS>

<QUAL_LIST_HEADS> • 10–177
 description of • 10–177
 related tags
 <QPAIR>
 <QUAL_LIST>
 <QUOTATION> • 2–2, 2–32
 description of • 2–32
 related tags
 <CODE_EXAMPLE>
 <SAMPLE_TEXT>

R

Reference notes
 in ARTICLE doctype • 2–6

Reference template
 See SOFTWARE doctype

<REF_NOTE> • 2–2, 2–7, 2–33
 description of • 2–33
 related tags

 <BACK_NOTE>
 <BIBLIOGRAPHY>
 <REF_NOTES>

<REF_NOTES> • 2–2, 2–7, 2–35
 description of • 2–35
 related tags

 <ACKNOWLEDGEMENTS>
 <BACK_NOTES>
 <REF_NOTES>

<RELATED_ITEM> • 10–178
 description of • 10–178
 related tags

 <RELATED_TAG>
 <RELATED_TAGS>

<RELATED_TAG> • 10–179
 description of • 10–179
 related tags

 <RELATED_ITEM>
 <RELATED_TAGS>

<RELATED_TAGS> • 10–180
 description of • 10–180
 related tags

 <RELATED_ITEM>
 <RELATED_TAG>

REPORT.TWOCOL doctype
 improving format of

REPORT.TWOCOL doctype
 improving format of (cont'd)
 See Two-column doctype designs • 2–8
 page layout of • 9–2
 sample output file • 9–7
 sample SDML file • 9–7

REPORT doctype • 9–1
 improving format of REPORT.TWOCOL
 See Two-column doctype designs • 9–3
 page layout of • 9–1
 sample output file • 9–4
 sample SDML file • 9–3
 tags • 9–10

<RESTRICTIONS> • 10–182
 description of • 10–182
 in Tag template • 10–182
 related tags

 <RITEM>
 <TAG_SECTION>

<RETTEXT> • 10–184
 description of • 10–184
 related tags

 <RETURNS>

<RETURNS> • 10–185
 description of • 10–185
 related tags

 <RETTEXT>

<RETURN_VALUE> • 10–187
 description of • 10–187

<RITEM> • 10–188
 description of • 10–188
 in Command template • 10–188
 related tags

 <RESTRICTIONS>

<ROUTINE> • 10–189
 description of • 10–189
 related tags

 <ROUTINE_SECTION>

Routine template
 See SOFTWARE doctype

<ROUTINE_SECTION> • 10–191
 description of • 10–191
 related tags

 <ARGDEF>
 <ARGDEFLIST>
 <ARGITEM>
 <ARGTEXT>
 <DESCRIPTION>
 <EXAMPLE_SEQUENCE>

Index

<ROUTINE_SECTION>
 related tags (cont'd)
 <FARG>
 <FARGS>
 <FFUNC>
 <FORMAT>
 <FRTN>
 <OVERVIEW>
 <RETTEXT>
 <RETURNS>
 <ROUTINE>
 <RSDEFLIST>
 <RSITEM>
 <SET_TEMPLATE_HEADING>
 <SET_TEMPLATE_LIST>
 <SET_TEMPLATE_PARA>
 <SET_TEMPLATE_TABLE>
<RSDEFLIST> • 10–196
 description of • 10–196
 related tags
 <RSITEM>
<RSITEM> • 10–198
 description of • 10–198
 related tags
 <RSDEFLIST>
<RUNNING_FEET> • 2–36, 6–26, 8–12, 9–22, 10–199
 description of • 2–36, 6–26, 8–12, 9–22, 10–199
 in ARTICLE doctype • 2–2, 2–5, 2–36
 in MILSPEC doctype • 6–26
 in OVERHEADS doctype • 8–2, 8–12
 in REPORT doctype • 9–3, 9–22
 in SOFTWARE doctype • 10–199
 related tags
 <AUTO_NUMBER>
 <CHAPTER>
 <RUNNING_TITLE>
 <SET_FOOTERS>
 <SLIDE>
<RUNNING_TITLE> • 2–37, 6–27, 8–14, 9–23, 10–200
 description of • 2–37, 6–27, 8–14, 9–23, 10–200
 in ARTICLE doctype • 2–2, 2–5, 2–37
 in MILSPEC.SECURITY doctype • 6–27
 in OVERHEADS doctype • 8–2, 8–14
 in REPORT doctype • 9–3, 9–23
 in SOFTWARE doctype • 10–200
 related tags
 <DOCUMENT_ATTRIBUTES>
 <RUNNING_FEET>
 <SLIDE>

S

<SALUTATION> • 4–3, 4–22
 description of • 4–22
 related tags
 <FROM_ADDRESS>
 <MEMO_FROM>
 <MEMO_TO>
 <TO_ADDRESS>
<SDML_TAG> • 10–202
 description of • 10–202
 related tags
 <SET_TEMPLATE_TAG>
 <TAG_SECTION>
<SECTION> • 9–3, 9–25
 description of • 9–25
<SECURITY> • 6–28
 description of • 6–28
 related tags
 <HIGHEST_SECURITY_CLASS>
 <SET_CONTENTS_SECURITY>
 <SET_PAGE_SECURITY>
 <SET_SECURITY_CLASS>
<SET_APPENDIX_NUMBER> • 6–31
 description of • 6–31
 related tags
 the global <APPENDIX> tag
 the global <SET_APPENDIX_LETTER> tag
 the global <SET_CHAPTER_NUMBER> tag
<SET_APPENDIX_NUMBER> tag • 6–3
<SET_CONTENTS_SECURITY> • 6–33
 description of • 6–33
 related tags
 <SECURITY>
 <SET_PAGE_SECURITY>
 <SET_SECURITY_CLASS>
<SET_HELP_LEVEL> • 3–9, 7–28
 description of • 3–9, 7–28
 related tags
 <BOOK_ONLY>
 <BOOK_ONLY>
 <HELP_ONLY>
 <HELP_ONLY>
 <KEEP_HELP_LEVEL>
 <KEEP_HELP_LEVEL>
 <SET_HELP_LEVEL>

- <SET_ONLINE_TOPIC> • 7–30
 - description of • 7–30
- <SET_PAGE_SECURITY> • 6–35
 - description of • 6–35
 - related tags
 - <SECURITY>
 - <SET_CONTENTS_SECURITY>
 - <SET_SECURITY_CLASS>
- <SET_SECURITY_CLASS> • 6–37
 - description of • 6–37, 6–38
 - in MILSPEC.SECURITY doctype • 6–37, 6–38
 - related tags
 - <HIGHEST_SECURITY_CLASS>
 - <SECURITY>
 - <SET_CONTENTS_SECURITY>
 - <SET_PAGE_SECURITY>
- <SET_TEMPLATE_ARGITEM> • 10–203
 - description of • 10–203
 - related tags
 - <ARGDEFLIST>
 - <ARGITEM>
 - <ROUTINE_SECTION>
- <SET_TEMPLATE_COMMAND> • 10–206
 - description of • 10–206
 - related tags
 - <COMMAND>
 - <COMMAND_SECTION>
- <SET_TEMPLATE_HEADING> • 10–209
 - description of • 10–209
 - related tags
 - <SET_TEMPLATE_LIST>
 - <SET_TEMPLATE_PARA>
 - the global <SET_TEMPLATE_HEADING> tag
- <SET_TEMPLATE_LIST> • 10–211
 - description of • 10–211
 - related tags
 - <SET_TEMPLATE_HEADING>
 - <SET_TEMPLATE_PARA>
 - the global <LIST> tag
- <SET_TEMPLATE_PARA> • 10–213
 - description of • 10–213
 - related tags
 - <SET_TEMPLATE_HEADING>
 - <SET_TEMPLATE_LIST>
 - <SET_TEMPLATE_TABLE>
- <SET_TEMPLATE_ROUTINE> • 10–215
 - description of • 10–215
- <SET_TEMPLATE_ROUTINE>
 - related tags
 - <ROUTINE>
 - <ROUTINE_SECTION>
- <SET_TEMPLATE_STATEMENT> • 10–218
 - description of • 10–218
 - related tags
 - <FUNCTION>
 - <STATEMENT>
 - <STATEMENT_SECTION>
- <SET_TEMPLATE_SUBCOMMAND> • 10–220
 - description of • 10–220
 - related tags
 - <COMMAND_SECTION>
 - <SUBCOMMAND>
 - <SUBCOMMAND_SECTION>
- <SET_TEMPLATE_TABLE> • 10–222
 - description of • 10–222
 - related tags
 - <SET_TEMPLATE_LIST>
 - <SET_TEMPLATE_PARA>
- <SET_TEMPLATE_TAG> • 10–225
 - description of • 10–225
 - related tags
 - <SDML_TAG>
 - <TAG_SECTION>
- <SHELF_CREATE> • 7–33
 - description of • 7–33
 - related tags
 - <BOOK_REF>
 - <SHELF_REF>
- <SHELF_REF> • 7–35
 - description of • 7–35
 - related tags
 - <BOOK_REF>
 - <SHELF_CREATE>
- <SHOW_LEVELS> • 9–3, 9–26
 - description of • 9–26
 - related tags
 - <LEVELS>
 - <OUTLINE>
- <SIGNATURES> • 9–28, 10–227
 - description of • 9–28, 10–227
 - in REPORT doctype • 9–3, 9–28
 - in SOFTWARE doctype • 10–227
 - related tags
 - <AUTHOR>
 - <BYLINE>

Index

<SIGNATURE_LINE> • 6–39
description of • 6–39
related tags
 <SIGNATURE_LIST>
 <SPECIFICATION_INFO>
 <SPEC_TITLE>
 <SUBTITLE>
the global <FRONT_MATTER> tag
the global <TITLE_PAGE> tag

<SIGNATURE_LIST> • 6–40
description of • 6–40
related tags
 <SIGNATURE_LINE>
 <SPECIFICATION_INFO>
 <SPEC_TITLE>
 <SUBTITLE>
 <TITLE_PAGE>

<SLIDE> • 8–2, 8–16
description of • 8–16
related tags
 <AUTHOR_INFO>
 <AUTO_NUMBER>
 <RUNNING_FEET>
 <RUNNING_TITLE>
 <SUBTITLE>
 <TEXT_SIZE>
 <TITLE>
 <TOPIC>

Slides
See OVERHEADS doctype

SOFTWARE.BROCHURE doctype
page layout of • 10–3

SOFTWARE.GUIDE doctype
page layout of • 10–3

SOFTWARE.HANDBOOK doctype
page layout of • 10–4

SOFTWARE.POCKET_REFERENCE doctype
page layout of • 10–4

SOFTWARE.REFERENCE doctype
page layout of • 10–4

SOFTWARE.SPECIFICATION doctype
headers and footers • 10–33
page layout of • 10–5
using template-enabling tags • 10–33

SOFTWARE doctype
arguments, parameters and qualifiers • 10–18
code fragments • 10–13
creating reference templates • 10–27
creating template tables • 10–28

SOFTWARE doctype (cont'd)
interactive or code examples • 10–22
messages
 See software messages • 10–14
modifying reference templates • 10–30
modifying template defaults • 10–30
reference templates
 Command template • 10–37
 sample output file • 10–41
 sample SDML file • 10–39
 Routine template • 10–45
 sample output file • 10–49
 sample SDML file • 10–47
 Statement template • 10–53
 sample output file • 10–55
 sample SDML file • 10–54
 Tag template • 10–58
 sample output file • 10–61
 sample SDML file • 10–60
software messages • 10–14
tags • 10–64
 in all of doctype • 10–64
terminal keys and keypads • 10–6
using reference templates • 10–23
using template-enabling tags • 10–32

Software messages
See SOFTWARE doctype

Software specifications
See MILSPEC doctype
See SOFTWARE doctype

<SOURCE_NOTE> • 2–2, 2–4, 2–39
description of • 2–39
related tags
 <AUTHOR>
 <VITA>

Specifications
for military
 See MILSPEC doctype
for software
 See SOFTWARE doctype

<SPECIFICATION_INFO> • 6–42
description of • 6–42, 6–43
related tags
 <SIGNATURE_LINE>
 <SIGNATURE_LIST>
 <SPEC_TITLE>
 <SUBTITLE>
<SPECIFICATION_INFO> tag • 6–3

- <SPEC_TITLE> • 6–44
 - description of • 6–44
 - related tags
 - <ONLINE_TITLE>
 - <SIGNATURE_LINE>
 - <SIGNATURE_LIST>
 - <SPECIFICATION_INFO>
 - <SUBTITLE>
- <STATEMENT> • 10–228
 - description of • 10–228
 - related tags
 - <FUNCTION>
 - <SET_TEMPLATE_STATEMENT>
 - <STATEMENT_SECTION>
- Statement template
 - See SOFTWARE doctype
- <STATEMENT_FORMAT> • 10–229
 - description of • 10–229
 - related tags
 - <CONSTRUCT_LIST>
 - <FCMD>
 - <FFUNC>
 - <FORMAT_SUBHEAD>
 - <FPARMS>
 - <STATEMENT_LINE>
- <STATEMENT_LINE> • 10–231
 - description of • 10–231
 - related tags
 - <CONSTRUCT>
 - <FCMD>
 - <FPARMS>
- <STATEMENT_SECTION> • 10–234
 - description of • 10–234
 - related tags
 - <CONSTRUCT>
 - <CONSTRUCT_LIST>
 - <FCMD>
 - <FFUNC>
 - <FORMAT_SUBHEAD>
 - <FPARMS>
 - <FUNCTION>
 - <OVERVIEW>
 - <SET_TEMPLATE_HEADING>
 - <SET_TEMPLATE_LIST>
 - <SET_TEMPLATE_PARA>
 - <SET_TEMPLATE_STATEMENT>
 - <SET_TEMPLATE_TABLE>
 - <STATEMENT>
- <STATEMENT_SECTION>
 - related tags (cont'd)
 - <STATEMENT_FORMAT>
 - <STATEMENT_SECTION>
- <SUBCOMMAND> • 10–238
 - description of • 10–238
 - related tags
 - <SET_TEMPLATE_SUBCOMMAND>
 - <SUBCOMMAND_SECTION>
- <SUBCOMMAND_SECTION> • 10–240
 - description of • 10–240
 - related tags
 - <COMMAND_SECTION>
 - <SET_TEMPLATE_SUBCOMMAND>
 - <SUBCOMMAND>
 - <SUBCOMMAND_SECTION_HEAD>
- <SUBCOMMAND_SECTION_HEAD> • 10–241
 - description of • 10–241
 - related tags
 - <SET_TEMPLATE_SUBCOMMAND>
 - <SUBCOMMAND>
- <SUBJECT> • 4–3, 4–23
 - description of • 4–23
 - related tags
 - <FROM_ADDRESS>
 - <MEMO_FROM>
 - <MEMO_LINE>
 - <MEMO_TO>
- <SUBTITLE> • 2–40, 6–45, 8–3, 8–18
 - description of • 2–40, 6–45, 8–18
 - in ARTICLE doctype • 2–2, 2–3, 2–40
 - in MILSPEC doctype • 6–3, 6–45
 - in OVERHEADS doctype • 8–18
 - related tags
 - <SIGNATURE_LINE>
 - <SIGNATURE_LIST>
 - <SPECIFICATION_INFO>
 - <SPEC_TITLE>
 - <TITLE>
 - <TITLE_SECTION>
- <SYNTAX> • 10–242
 - description of • 10–242
 - related tags
 - <DISPLAY>
 - <SYNTAX_DEFAULT_HEAD>
- <SYNTAX_DEFAULT_HEAD> • 10–244
 - description of • 10–244
 - related tags
 - <SYNTAX>

Index

T

Tags, doctype_specific
using • 1–2

Tag template

See SOFTWARE doctype

<TAG_SECTION> • 10–246

description of • 10–246

related tags

<DESCRIPTION>

<EXAMPLE_SEQUENCE>

<FORMAT>

<FTAG>

<OVERVIEW>

<PARAMDEFLIST>

<RELATED_ITEM>

<RELATED_TAG>

<RELATED_TAGS>

<RESTRICTIONS>

<RITEM>

<SDML_TAG>

<SET_TEMPLATE_HEADING>

<SET_TEMPLATE_LIST>

<SET_TEMPLATE_PARA>

<SET_TEMPLATE_ROUTINE>

<SET_TEMPLATE_TABLE>

<SET_TEMPLATE_TAG>

<TERMINATING_TAG>

Technical manual

for military specifications

See MILSPEC doctype

for software

See SOFTWARE doctype

general purpose

See MANUAL doctype

Technical report

See REPORT doctype

Templates

See MILSPEC doctype

See SOFTWARE doctype

<TERMINATING_TAG> • 10–250

description of • 10–250

related tags

<RELATED_TAG>

<TAG_SECTION>

Text formatter memory • 7–22

running out of with long informal text elements •
7–24

<TEXT_SIZE> • 8–3, 8–19

description of • 8–19

related tags

<SLIDE>

<TITLE> • 2–41, 8–3, 8–21

description of • 2–41, 8–21

in ARTICLE doctype • 2–2, 2–3, 2–41

in OVERHEADS doctype • 8–21

related tags

<SUBTITLE>

<TITLE_SECTION>

<TITLE_SECTION> • 2–2, 2–42

description of • 2–42

related tags

<SUBTITLE>

<TITLE>

<TOPIC> • 8–3, 8–22

description of • 8–22

related tags

<SLIDE>

<TEXT_SIZE>

Topics

default online • 7–30

<TO_ADDRESS> • 4–3, 4–24

description of • 4–24

related tags

<FROM_ADDRESS>

<MEMO_TO>

Transparency

See OVERHEADS doctype

Two-column doctype designs

improving format of • 2–8

U

Using reference templates

in SOFTWARE doctype • 10–23

Using template-enabling tags

in SOFTWARE.SPECIFICATION doctype • 10–33

in SOFTWARE doctype • 10–32

V

<VITA> • 2-2, 2-3, 2-43
description of • 2-43

related tags

- <AUTHOR>
- <AUTHOR_ADDR>
- <AUTHOR_AFF>
- <AUTHOR_LIST>
- <SOURCE_NOTE>

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	_____	Local Digital subsidiary or approved distributor
Internal ¹	_____	USASSB Order Processing - WMO/E15 <i>or</i> U.S. Area Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

¹For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

VAX DOCUMENT
Using Doctypes and
Related Tags
AA-JT86B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less _____

What I like best about this manual is _____

What I like least about this manual is _____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

I am using **Version** _____ of the software this manual describes.

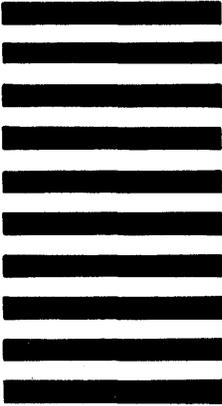
Name/Title _____ Dept. _____
Company _____ Date _____
Mailing Address _____
Phone _____

--- Do Not Tear - Fold Here and Tape ---

digital™



No Postage
Necessary
if Mailed
in the
United States



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987



--- Do Not Tear - Fold Here ---

Reader's Comments

VAX DOCUMENT
Using Doctypes and
Related Tags
AA-JT86B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

I would like to see more/less _____

What I like best about this manual is _____

What I like least about this manual is _____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

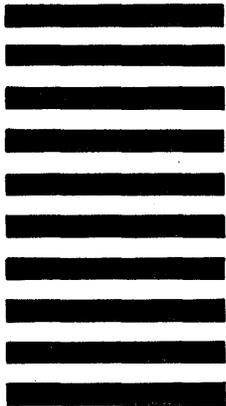
Phone _____

— Do Not Tear - Fold Here and Tape

digitalTM



No Postage
Necessary
if Mailed
in the
United States



BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987



--- Do Not Tear - Fold Here