Digital Internal Use Only

DOCTOR
User's Guide

Don't Panic

This document describes the functionality of the DOCTOR utility that allows manipulation of several VAX DOCUMENT related input or output files as well as (minimal) conformant PostScript files.

Although its size has grown over the years, it is easy reading and you may only need to read a single chapter if you're only interested in a single feature of DOCTOR.

Revision/Update Information: DOCTOR V3.3

Document release date 3-JAN-1994 17:37:26.09

DOCTOR is proprietary Digital software

Digital Internal Use Only

January 1994

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.

© Digital Equipment B.V. 1990, 1994.

All Rights Reserved.
Processed in the Netherlands

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	digital™

Digital Internal Use Only

Contents

PREFACE vii

CHAPTER 1 DOCTOR AT A GLANCE - QUICK OVERVIEW 1–1

1.1 QUESTIONS 1–1

1.2 ANSWERS 1–3
1.2.1 Add blank pages to a PostScript file 1–3
1.2.2 Extract page range from PostScript file 1–3
1.2.3 Adding diagonal wording to PostScript pages 1–4
1.2.4 Adding footers to PostScript pages 1–4
1.2.5 Adding headers to pages of a PostScript document 1–4
1.2.6 Extract figures from a PostScript document 1–5
1.2.7 Re-arrange pages for saddle-stitch printing 1–5
1.2.8 List of an SDML document composition 1–5
1.2.9 Automate document generation using MMS 1–6
1.2.10 Prepare SDML file for Bookreader 1–6
1.2.11 Produce a list of all <X> index entries 1–6
1.2.12 Produce a list of all symbols defined 1–7
1.2.13 Cross referencing between documents 1–7
1.2.14 Sort a Glossary section 1–8
1.2.15 Count all tags used 1–8
1.2.16 Create SDML <MESSAGE_SECTION> from VMS Message source 1–8
1.2.17 Define dummy symbols 1–9

CHAPTER 2 DOCTOR'S COMPONENTS 2–1

CHAPTER 3 DOCTORING SDML FILES: SORTING GLOSSARY ENTRIES 3–1

3.1 OVERVIEW 3–1

3.2 SORTING PROCESS 3–1

3.3 SORT ORDER 3–2

3.4 OPTIONS 3–3

CHAPTER 4 DOCTORING VMS MESSAGE SOURCE FILES 4–1

4.1 OVERVIEW 4–1

4.2 COMMENTING REQUIREMENTS FOR A MESSAGE FILE 4–2

4.3 INCLUDE RESULT IN DOCUMENTATION 4–5

4.4 DOCTOR/MESSAGE FEATURES ON MESSAGE FILES 4–5

Digital Internal Use Only

Contents

CHAPTER 5 DOCTORING SDML FILES FOR BOOKREADER OUTPUT 5–1

5.1 OVERVIEW 5–1

5.2 EXAMPLE OF ONLINE CONVERSION 5–2

5.3 INCLUDING PREDEFINED SYMBOLS 5–5

5.4 GENERATING ADDITIONAL SYMBOLS 5–5

5.5 THINGS TO REMEMBER 5–6
5.5.1 Using /OUTPUT 5–6
5.5.2 Using /VERSION_1 5–7

5.6 ADDING SYMBOLS 5–7

5.7 ADDING REFERENCES 5–7

5.8 ADDING FIGURE FILES 5–7

5.9 WHAT MUST YOU DO? 5–8

5.10 REMOVING OR RENEWING ADDED SYMBOLS 5–8

CHAPTER 6 DOCTORING POSTSCRIPT FILES 6–1

6.1 OVERVIEW 6–1

6.2 ADDING BLANK PAGES 6–2

6.3 REPLACING THE PROLOG 6–4

6.4 EXTRACTING RANGES 6–7

6.5 REMOVING OR RE-USING FIGURES 6–8
6.5.1 Removing figures 6–9
6.5.2 Extracting figures 6–9

6.6 SADDLE STICH PRINTING 6–10
6.6.1 Using an LPS40 to print the file 6–12
6.6.2 Using an LPS20 to print the file 6–12

6.7 MINIMAL CONFORMANT FILES 6–13

CHAPTER 7 DOCTORING SDML FILES: HIERARCHY OF FILES WITHIN A BOOK 7–1

7.1 OVERVIEW 7–1

7.2 LISTING ALL INCLUDED FILES 7–1

7.3 PRODUCING MMS REBUILD FILES 7–3

7.4 USING THE MMS DESCRIPTION FILE 7–9
7.4.1 What file to build? 7–9
7.4.2 Another doctype than specified 7–10
7.4.3 Another destination than specified 7–10
7.4.4 Using DOCUMENT/GRAPHICS (Rags) 7–10

7.5 RETRIEVE ALL INDEX ENTRIES 7–11

CHAPTER 8 DOCTORING SDML FILES: COUNTING TAGS USED 8–1

8.1 OVERVIEW 8–1

8.2 OPTIONS 8–1

8.3 OUTPUT FORMAT 8–1

8.4 USAGE 8–2

CHAPTER 9 DOCTORING XREF FILES 9–1

9.1 OVERVIEW 9–1

9.2 BUILD SYMBOL LISTINGS 9–1

9.3 BUILD CROSS REFERENCE SYMBOL FILE 9–2

9.4 EXAMPLE OUTPUT 9–3

9.4.1 Symbol file output 9–3

9.4.2 List file output 9–4

9.4.3 Sorted list file output 9–5

COMMAND SECTION

DOCTOR/GLOSSARY 9

DOCTOR/MESSAGE 11

DOCTOR/ONLINE 13

DOCTOR/PS 16

DOCTOR/SDML 22

DOCTOR/TAG_COUNT 26

DOCTOR/XREF 27

APPENDIX A ERROR MESSAGES A–1

A.1 DOCMSG MESSAGES A–1

A.2 DOCTOR MESSAGES A–1

A.3 DTAGCOUNT MESSAGES A–2

A.4 GLOSSARY MESSAGES A–2

A.5 MARFIN MESSAGES A–4

A.6 ONLINE MESSAGES A–6

A.7 PSSCAN MESSAGES A–8

A.8 SADDLE MESSAGES A–13

A.9 XREF MESSAGES A–17

Digital Internal Use Only

Contents

INDEX

EXAMPLES

3–1 Glossary entry for DOCTOR/GLOSSARY 3–2
4–1 Sample Message source file 4–3
4–2 Sample DOCTOR/MESSAGE output 4–4
5–1 Original file 5–2
5–2 Modified file 5–3
6–1 Conformant PostScript skeleton 6–15
7–1 Hierarchy list produced by DOCTOR/SDML 7–4
7–2 Example DEC/MMS description file 7–6
7–3 Output of DOCTOR/SDML/INDEX 7–12
8–1 Sample DOCTOR/TAG_COUNT output file 8–2
9–1 Sample .SDML_XREF output 9–4

FIGURES

6–1 Saddle stiched printing 6–10

TABLES

6–1 FONT keyword values 6–6
6–2 Extracting or removing figures 6–8
7–1 Tags that cause other files to be included 7–2
7–2 DOCTOR/SDML qualifiers for DEC/MMS 7–6
9–1 Alphabetic list of symbols 9–5
9–2 Chapter symbols - in alphabetical order 9–5
9–3 Chapter symbols - in numerical order 9–6

Digital Internal Use Only

Preface

Location and installation

The DOCTOR utility can be found in and copied from

```
NSIC00::PSQ:[PUBLIC.SOURCES.DOCTOR]*.*
```

This includes the `.EXE` executable image and some documentation for a variety of output devices, including bookreader format. This document is also located there. DOCTOR requires VAX/VMS V5.4-2 or higher.

From V3.3-24 onwards, `DOCTOR.EXE` will work on all VAX/VMS systems whether or not DEC/CMS is installed as a version management product. Before this release, a special version `DOCTOR_NOCMS.EXE` is required for sites that do not have DEC/CMS available.

To use DOCTOR properly, it must be invoked as a foreign command. This is achieved by defining a symbol (preferably in your login command procedure so it is always defined when you start a session) as

```
$ DOCTOR :== $disk:[dir]DOCTOR.EXE
```

The *disk:[dir]* part must be the location where you copied the program file into. If many people use this utility, it is advisable to copy the program into the directory pointed to by logical name `DOC$LOCAL_TOOLS`.

If this is all done, you're ready to use the DOCTOR.

Important

Because DOCTOR is a collection of tools under one umbrella, it is important that each invocation starts with the qualifier that indicates which tool you want to use: either `DOCTOR/GLOSSARY`, `DOCTOR/PS`, `DOCTOR/SDML`, `DOCTOR/ONLINE`, `DOCTOR/MESSAGE`, `DOCTOR/TAG_COUNT` or `DOCTOR/XREF`.

A quick overview on how to use DOCTOR and what it can do for you, is provided in Chapter 1, DOCTOR at a glance - Quick Overview.

For problems, QAR's, and questions, please send a mail to Theo de Klerk via either `NSIC00::KLERK` or `Theo de Klerk @UTO`. After mid-December 1993 the E-mail address will be `NSIC00::KLERK`. Because DOCTOR is an unfunded, free-time product, no support is guaranteed or implied. However, the author will try to keep DOCTOR up to date whenever possible. An additional platform for discussions on DOCTOR is its conference notes file on `IJSAPL::DOCTOR`.

Preface

Availability of DOCTOR

This utility is a product of DS/Country Program Office in the Netherlands. It is developed to be used within Digital Equipment Co. As of January 1st, 1993 it is allowed to use this tool on customer sites, provided the following conditions are explicitly communicated to the customer:

1 DOCTOR is Digital proprietary software. It may not be further duplicated, distributed or sold other than through Digital channels.

2 DOCTOR is an unsupported product. Although bug reports, enhancement requests are welcomed, no guarantee is given or implied that any modification will be made to DOCTOR.

Acknowledgements

DOCTOR came into existence initially after some discussions with Mark Devries on the topic of blank pages in PostScript.

The underlay features to include "draft" diagonally across the page was initally written in 1988 as a PostScript routine by Lance McNulty, and later re-created for more general use by Chris Mackay, who also provided the current PostScript code for the `/PS/CHANGE` function.

Will Kohlbrenner provided much of the information required to interpret the XREF files for cross referencing.

The `/ONLINE` code in VAX SCAN finally accepted nested tags after vital help from Barry Rogoff in showing how the TRIGGER EXPOSE and NOTANY had to be combined. Mary Utt provided test files and functioned as tester of /ONLINE while V2.0 was still in engineering phase.

Others who were vital in providing me with some crucial answers to questions raised during the development of DOCTOR are Sheila Huston, Sheila Lawner, David and Tom Parmenter.

Unfortunately, several of these people no longer work for Digital and continued their career outside the company - by choice or due to Digital's restructuring. It was a privilege to work with them.

Technical Changes

Unfortunately, there is a non-upwards compatibility issue for users of DOCTOR V3.0 or earlier and those using V3.1 or higher.

You may need to modify command procedures using DOCTOR, if they use the `DOCTOR /SDML/MMS` utility to produce an MMS file to rebuild the document. All qualifiers related to the specification of a doctype or destination name needed to be reworked, as DCL could not always distinguish between them as more than the first four characters were identical.

Removed are the qualifiers `/LN03`, `/POSTSCRIPT`, `/LINE_PRINTER`, `/MAIL`, `/BOOKREADER`. The same functionality is now made available in a single qualifier with keywords:

```
/DESTINATION=(LN03= , POSTSCRIPT= , LINE_PRINTER= , MAIL= , BOOKREADER= )
```

Default settings when not mentioned remained the same (LN03, PS, LINE, MAIL and BOOK respectively).

Also removed are the qualifiers `/DOCTYPE` and `/ONLINE_DOCTYPE`. They have been replaced with a similar single qualifier:

```
/DOCTYPE=(PAPER= , ONLINE= , MANPAGE=  )
```

From V3.3-22 onwards, there is an incompatibility for the `/ONLINE/DUMMY_SYMBOL` qualifier. Before V3.3-22 this qualifier would accept a file specification. From V3.3-22 onwards, it takes two keywords: `OUTPUT=` which still takes the output file spec and `TEXT="string"` to allow the user to specify a text string as value for each undefined symbol. By default, `/DUMMY_SYMBOL` is equal to `/DUMMY_SYMBOL=(OUTPUT=infile.SDML_DUMMY_SYMBOLS)`, which is the same behaviour of pre-V3.3-22.

From V3.3-24 onwards, there is no need for a special version of DOCTOR that does not depend on DEC/CMS. By not calling the CMS routines directly but through `LIB$FIND_IMAGE_SYMBOL`, DOCTOR will discover itself whether or not DEC/CMS is available for use.

Digital Internal Use Only

1 DOCTOR at a glance - Quick Overview

This chapter will give a short overview to the DOCTOR utility and allows you to find the information required quickly through a set of questions to be answered. This chapter may be the only one you need to read as a new user of DOCTOR.

1.1 Questions

1. What file needs to be used by DOCTOR?

- A `.PS` PostScript file. Goto 3. What modification must be made to the PostScript file?
- A `.SDML` VAX DOCUMENT source file. Goto 4. What must be done with the SDML file?
- A `.XREF` cross reference file produced by a bookbuild run of VAX DOCUMENT. Goto 5. What do you want to know from the XREF file?.
- A `.MSG` VAX Message source file. See Section 1.2.16.
- Don't know but I know what I want to do. Goto 2. What is it you want to do? .

2. What is it you want to do?

- Convert a VAX Message source file into a VAX DOCUMENT source file describing the messages in a <MESSAGE_SECTION>.
- Get an overview of all symbols used in a document written for VAX DOCUMENT. See Section 1.2.12.
- Get an overview of all source files, figure files and other files used and referenced by a VAX DOCUMENT written document. See Section 1.2.8.
- MMS must automate the building of a VAX DOCUMENT written document. See Section 1.2.9.
- Be able to refer to sections in one book from another book, when both books are written in VAX DOCUMENT source code. See Section 1.2.13.
- Sort a VAX DOCUMENT glossary alphabetically although the items are currently unsorted. See Section 1.2.14
- List all tags used in a VAX DOCUMENT written document and their usage count. See Section 1.2.15.
- Add symbols to all VAX DOCUMENT tags in a document to allow the document to process correctly for the BOOKREADER destination. See Section 1.2.10.

Digital Internal Use Only

- Produce a VAX DOCUMENT source file that contains <DEFINE_SYMBOL> tags for all symbols that are referenced but not actually assigned to a specific section, table or figure. See Section 1.2.17.
- Add blank pages to an unbalanced PostScript file. See Section 1.2.1.
- Extract a section from a PostScript file. See Section 1.2.2.
- Add a diagonal faint grey text on all pages of a PostScript document. See Section 1.2.3.
- Add a footer to all pages. See Section 1.2.4.
- Add a header to all pages. See Section 1.2.5.
- Extract figures from a PostScript document. See Section 1.2.6.
- Re-arrange pages in a PostScript file so I can print them as a signature booklet (saddle stitching). See Section 1.2.7.
- Number pages of an unnumbered PostScript document. See Section 1.2.4.

3. What modification must be made to the PostScript file?

- Add blank pages to an unbalanced PostScript file. See Section 1.2.1.
- Extract a section from a PostScript file. See Section 1.2.2.
- Add a diagonal faint grey text on all pages of a PostScript document. See Section 1.2.3.
- Add a footer to all pages. See Section 1.2.4.
- Add a header to all pages. See Section 1.2.5.
- Extract figures from a PostScript document. See Section 1.2.6.
- Re-arrange pages in a PostScript file so I can print them as a signature booklet (saddle stitching). See Section 1.2.7.
- Number pages of an unnumbered PostScript document. See Section 1.2.4.

4. What must be done with the SDML file?

- A list must be produced of all files that make up a document. See Section 1.2.8.
- An MMS description file must be made to automate the rebuild of the document. See Section 1.2.9.
- The document must have symbols added to enable it to be processed to a Bookreader file. See Section 1.2.10.
- A list must be produced of all symbols used in a document. See Section 1.2.12.
- A list of all index entries <X> and <Y> must be produced. See Section 1.2.11.
- The glossary entries must be sorted alphabetically. See Section 1.2.14

Digital Internal Use Only

- A list of all tags used in the document must be given, with their usage count. See Section 1.2.15.

5. What do you want to know from the XREF file?

- A list of all symbols defined in the book. See Section 1.2.12
- A file defining all symbols of one book as symbols to be used in another book for cross referencing between books. See Section 1.2.13.

1.2 Answers

1.2.1 Add blank pages to a PostScript file

To be used only on documents that for some reason omitted to include blank even (lefthand side) pages when a chapter ends on an odd page.

```
DOCTOR/PS /BLANK inputfile.PS [ /OUTPUT=outputfile.PS ]
```

If you have a default blank page defined as PostScript file (e.g. with texts like "Left Blank") you can specify such a blank page file with the `/BLANK=blankpagefile.PS` qualifier.

Example:

```
DOCTOR/PS file.PS /OUTPUT=twosidedfile.ps /BLANK
```

For details, see Section 6.2.

1.2.2 Extract page range from PostScript file

Any page range or ranges can be extracted from a PostScript file, provided you specify the correct page values. These are mentioned inside the PostScript file on the lines that start with `%%Page:`. With a DCL command

```
$ SEARCH  inputfile.PS  %%Page:
```

you will see all valid page values specified following the `%%Page:` word.

To extract a range then specify

```
DOCTOR/PS inputfile.PS /OUTPUT=outputfile.PS -
      /EXTRACT=(START=startpage, END=endpage)
```

Or, when a number of pages must be printed following the starting page:

```
DOCTOR/PS inputfile.PS /OUTPUT=outputfile.PS -
      /EXTRACT=(START=startpage, NUMBER=integervalue)
```


Digital Internal Use Only

Example:

```
DOCTOR/PS completefile.PS /OUTPUT=partialfile.PS -
    /EXTRACT=(START=4-6, END=5-2) -
    /EXTRACT=(START=8-1,NUMBER=6) -
    /EXTRACT=(START=INDEX-1, END=INDEX-9)
```

For details, see Section 6.4.

1.2.3 Adding diagonal wording to PostScript pages

To add a faint diagonal text to all pages of a PostScript file, specify

```
DOCTOR/PS inputfile.PS /CHANGE=(DIAGONAL="text string") /OUTPUT=outputfile.PS
```

For details, see Section 6.3.

1.2.4 Adding footers to PostScript pages

To add a footer on all pages of a PostScript document, use the command

```
DOCTOR/PS inputfile.PS /CHANGE=(BOTTOM="text string") /OUTPUT=outputfile.PS
```

To add a counter value on each page footer starting on a certain positive integer value that increments with each page, specify

```
DOCTOR/PS inputfile.PS /CHANGE=(COUNTER=integer) /OUTPUT=outputfile.PS
```

The `BOTTOM=` and `COUNTER=` arguments can be combined. Then the number is appended to the specified footer text.

Example:

```
DOCTOR/PS inputfile.PS /CHANGE=(BOTTOM="text string", COUNTER=50) -
      /OUTPUT=outfilefile.PS
```

For details, see Section 6.3.

1.2.5 Adding headers to pages of a PostScript document

To add a text string at the top of each page, use the command

```
DOCTOR/PS inputfile.PS /CHANGE=(TOP="text string") /OUTPUT=outfilefile.PS
```

For details, see Section 6.3.

Digital Internal Use Only

1.2.6 Extract figures from a PostScript document

When figures in a document are made as encapsulated PostScript files and included in the complete document, you can extract those figures into individual encapsulated PostScript files again by using the command:

```
DOCTOR/PS inputfile.PS /FIGURES=EXTRACT
```

Each figure will get the name it had originally when included into the document. If you want their filenames changed to reflect the page number on which they are found in the complete document, specify

```
DOCTOR/PS inputfile.PS /FIGURES=(EXTRACT, PAGE_NUMBER)
```

Whether a document has included encapsulated PostScript figures can be determined by searching for the lines `%%BeginDocument` inside the input PostScript file.

For details, see Section 6.5.

1.2.7 Re-arrange pages for saddle-stitch printing

To re-order the page order inside a PostScript document to allow for printing four pages at half size on the two sides of a single sheet of paper so that the booklet can be folded as a magazine, you must first ensure whether the document has a balanced set of pages (containing blank pages where needed - see Section 1.2.1 to do this). Then, you can reorder the pages by the command:

```
DOCTOR/PS/SADDLE inputfile.PS /OUTPUT=outputfile.PS
```

Then the booklet is printed on an LPS20 double side PostScript laser printer using:

```
PRINT/QUEUE=lps20_PSqueue/PARAMETER=(SIDE=TUMBLE, NUMBER=2) outputfile.PS
```

For details, see Section 6.6.

1.2.8 List of an SDML document composition

A list of all files (figures, tables, elements etc) used in one document can be produced using the command:

```
DOCTOR/SDML mainfile.SDML  [/OUTPUT=outputfile.SDML]
```

This results in a new version of the main file where the structure of the document is written in a <COMMENT> header at the start of that file. The main file must be the file that is specified on the DOCUMENT command line for processing (i.e. the profile file in bookbuilds).

For details, see Section 7.2.

Digital Internal Use Only

1.2.9 Automate document generation using MMS

To generate an MMS description file that can be used to automate the processing of a VAX DOCUMENT document, use the command

```
DOCTOR /SDML mainfile.SDML /MMS  [/OUTPUT=NL:]
```

This results in a `mainfile.MMS` description. You must specify `/OUTPUT=NL:` if you don't want to generate a new `mainfile.SDML` that contains a listing of the document structure (as described in Section 1.2.8).

You can also specify a specific doctype and/or destination if you don't like the defaults (REPORT, SOFTWARE.ONLINE, MANPAGE and LN03, LINE_PRINTER, PS, MAIL, BOOKREADER, ROFF). For example:

```
DOCTOR/SDML mainfile.SDML /MMS -
    /DOCTYPE=(PAPER=MANUAL,  ONLINE=MANUAL.ONLINE, ) -
    /DESTINATION=(POSTSCRIPT=POST, LN03=MY_OWN_LN03, MAIL=TEXT)
```

For details, see Section 7.3.

1.2.10 Prepare SDML file for Bookreader

A document must have symbols attached to all sectioning tags, tables, examples and figures and these must also be referenced using the <REFERENCE>. DOCTOR can add all this where it is not provided by the author using the command:

```
DOCTOR/ONLINE mainfile.SDML [/SUPERSEDE] /OUTPUT=.SDML
```

This will result in new versions of all files of which the document consists with `\BK_ADDED_nn` symbols added where needed. When a document is modified or has been processed by DOCTOR/ONLINE before, the `/SUPERSEDE` qualifier will remove all old symbols that were added before adding new ones where needed.

When many symbols are predefined in a special symbol file, these can be made known to DOCTOR too using

```
DOCTOR/ONLINE mainfile.SDML /SYMBOL_FILE=filespec /OUTPUT=.SDML
```

For details, see Chapter 5.

1.2.11 Produce a list of all <X> index entries

To produce a text file in which all <X> and <Y> tags are listed, together with their origin (source file and line number) can be produced by the command:

```
DOCTOR/SDML main_file.SDML /INDEX=outputfile_spec
```

This output file can be used to check for typing errors where index entries should be identical and are not. By sorting the output file:

Digital Internal Use Only

```
SORT index_file  index_file
```

all <X> entries are sorted alphabetically and mistyped entries are easily discovered and can be corrected in the original source files.

For details, see Section 7.5.

1.2.12 Produce a list of all symbols defined

When a bookbuild is performed on a profile file, VAX DOCUMENT also produces a cross reference file with the name of the profile file and file type .XREF. This cross reference file can be inspected by DOCTOR and it can generate several tabular lists that describe all symbols and their translation when used with a <REFERENCE> tag. These lists are produced in VAX DOCUMENT source format, so the resulting files from DOCTOR must be processed by VAX DOCUMENT (e.g. using REPORT doctype).

To produce a list of all symbols in alphabetical order, use

```
DOCTOR/XREF profilefile.XREF /LIST  [/FULL]
```

When the /FULL is omitted, the tables only contain the text that would be shown when <REFERENCE>(value) would have been used.

To produce a list of all symbols sorted in alphabetical order, but divided by their type (chapter, section, figure, table etc), use

```
DOCTOR/XREF profilefile.XREF /SORT  [/FULL]
```

To produce a list of all symbols sorted in numerical order (1.1, 1.2, 1.2.1 etc) and by their type, use

```
DOCTOR/XREF profilefile.XREF /SORT /NUMERIC  [/FULL]
```

For details, see Section 9.2.

1.2.13 Cross referencing between documents

When DOCUMENT performs a bookbuild, it produces a cross reference file with the same name as the profile file and with file type .XREF. This file contains all symbols defined in the document and their value. It is possible for DOCTOR to inspect this cross reference file and produce another SDML file with <DEFINE_SYMBOL> tags for each symbol in the file. This way, the resulting SDML file can be included into another book (using the <INCLUDE> or /SYMBOL= qualifier) and the symbols of the first book be used in the second to allow cross referencing between books.

To generate a symbol file from a book after VAX DOCUMENT has produced the required .XREF file, use the command

Digital Internal Use Only

```
DOCTOR/XREF profilefile.XREF /SYMBOL -
      /PREFIX=string -
      /BOOKTITLE="title of the book" -
      [/FULL | /VALUE | /TEXT ]
```

A prefix can be added to all symbols to avoid collision with symbols defined in the second book. A booktitle should be specified to allow it to be named in the second book (e.g. "See 'European Tales', Chapter 5"). Depending on the use of /FULL, /VALUE or /TEXT the symbols will contain only those parts that would be seen when the <REFERENCE> was used with either the argument FULL, VALUE or TEXT. When omitted, /VALUE is assumed.

For details, see Section 9.3.

1.2.14 Sort a Glossary section

When a VAX DOCUMENT source file contains a <GLOSSARY> section, the defined terms can be written in any order and DOCTOR can sort them alphabetically using the <GTERM> tags for sorting. The rest of the document, outside the glossary, remains untouched. To sort the glossary, use the command

```
DOCTOR/GLOSSARY  inputfile.SDML
```

When a specific ordering is wanted, you can additionally specify the /SORT qualifier which has the same options as the /INDEX qualifier of the DOCUMENT command to ensure that both glossary and index are sorted in similar fashion:

```
/SORT=( [LETTER|WORD] [,NONALPHA={AFTER|BEFORE|IGNORE}] )
```

For details, see Chapter 3.

1.2.15 Count all tags used

To obtain a list of all tags used and their usage count, you specify the command

```
DOCTOR/TAG_COUNT mainfile.SDML /OUTPUT=outputfile.TXT
```

This produces a plain text file where all tags are listed alphabetically with the number of times the tag was used. The main file specified is searched as well as any other file that is included by it.

For details, see Chapter 8.

1.2.16 Create SDML <MESSAGE_SECTION> from VMS Message source

To convert a VAX/VMS message utility source file (file type .MSG into documentation for VAX DOCUMENT processing, the message source file needs to be written in a specific format, where each message is followed by a comment block structured as:

```
FNF <File !AS not found>/FAO=1
!+
! Explanation:  text
! User Action:  text
!-
```

The comment block must start with `!+` and end with `!-`. Any comment outside the block is ignored. The explanation and user action text can span multiple lines. When a colon is needed in the text, the <COLON> must be used.

Message source files formatted as described above can be converted into a VAX DOCUMENT `.SDML` file using the command

```
DOCTOR/MESSAGE vmsmessage.MSG [/OUTPUT=outputfile.SDML ]
```

For details, see Chapter 4.

1.2.17 Define dummy symbols

When a VAX DOCUMENT source references symbols that do not (yet/anymore) exist, but the document should process properly without aborting on warnings of undefined symbols, an SDML file can be produced that defines all missing symbols with value "<REFERENCE>(missing_symbol)". This file can then be either included with <INCLUDE> or by the DOCUMENT command qualifier `/SYMBOL`.

To produce such a file with <DEFINE_SYMBOL> tags, use the command:

```
DOCTOR/ONLINE mainfile.SDML /OUTPUT=NL:/DUMMY_SYMBOLS
```

or, when many symbols are predefined in a special symbol file:

```
DOCTOR/ONLINE mainfile.SDML /SYMBOL=symbolfilespec/OUTPUT=NL:/DUMMY_SYMBOLS
```

This will keep track of conditions set and skip blocks that do not match the currently active condition. Using `/CONDITION=` allows you to specify such a condition to DOCTOR, similar to the way you would specify this to DOCUMENT. For details, see Section 5.4.

Digital Internal Use Only

2 DOCTOR's components

The utility consists of several more or less independent parts, each working on some sort of file produced by the VAX DOCUMENT typesetting software. Only the part that works on PostScript files can also be used for other PostScript files, as long as they adhere to the Adobe specified *minimal conformant* file structure. This is defined in Appendix C of the "*Red Book*", the *PostScript Language Reference Manual*, published by Addison & Wesley for Adobe Inc. DOCTOR supports both the Adobe structured comments versions V2 (used by VAX DOCUMENT V1.2B and V2.0-1) and V3 (used by VAX DOCUMENT V2.1-1).

The types of files that can be handled by the DOCTOR are:

- PostScript files
- XREF cross reference files of VAX DOCUMENT
- SDML source files of VAX DOCUMENT
- MSG message files of the VMS Message utility to produce SDML files

These will be discussed in the next chapters.

Digital Internal Use Only

3 DOCTORing SDML files: Sorting glossary entries

3.1 Overview

Using the DOCTOR/GLOSSARY qualifier invokes the glossary item sort utility. It will accept the input of any properly coded SDML source file with <GTERM> entries and sorts these in ASCIIbetical order with some small corrections on its collating sequence. Under a *properly coded* file, it is understood that only a single <GLOSSARY> - <ENDGLOSSARY> block exists in the .SDML file. Multiple glossaries inside a single file are not supported by DOCTOR.

The glossary does not need to be the only part of the file: it may be embedded in a larger SDML file. Only the glossary part is sorted. Text preceding or following the glossary section remains untouched.

Note:

An informational header is produced in the output file. This header is recognized by *GL* markers. When new entries are added to a previously sorted file, DOCTOR/GLOSSARY will remove the original header and replace it by a new one. This can only be done if the header remains untouched. (It can always be manually removed, of course).

Known limitations

The following limitations apply:

1 Only a single <GLOSSARY> block can exist in the input file. Multiple glossaries in one file are not supported.

2 DOCTOR/GLOSSARY ignores any <COMMENT> or <LITERAL> tags inside the glossary. This may lead to wrong results if a comment block includes all or part of a <GTERM> / <GDEF> tag. It may then also cause <COMMENT> and <ENDCOMMENT> to be sorted to different parts of the file. Comment blocks within the range of a single glossary term are no problem. Comment blocks belonging to a glossary term should follow its <GTERM>, not preceed it. Comment blocks should also not span or include <GTERM> tags.

3.2 Sorting process

DOCUMENT/GLOSSARY sorts the input file by looking for the first line that contains a <GTERM> tag. Anything preceding this tag will be copied verbatim into the output file.

Digital Internal Use Only

All text between the first <GTERM> tag and the tag <ENDGLOSSARY> will be sorted according to the textual argument of the <GTERM> tag. A glossary term is considered to be all text from the start of the corresponding <GTERM> tag upto the position of the next <GTERM> (or the final <ENDGLOSSARY>). This means that it is possible to insert other code such as <COMMENT>, <X> or <Y> within this text block - either inside the <GTERM> and associated <GDEF> tag argument, or as a separate tag, not nested within an argument, as Example 3–1 shows.

Example 3–1 Glossary entry for DOCTOR/GLOSSARY

```
<GTERM>(Call frame)
<X>(call frame)
<COMMENT>
  Next definition is taken from Digital Press publication
  "Digital Dictionary".
<ENDCOMMENT>
<GDEF>(Standard data structure built on the stack during a procedure
or function call, starting from the location addressed by the frame
pointer (FP) register, to lower addresses. Same as stack frame.)

<COMMENT>(This comment belongs to the Call Frame entry)

<gterm>(Channel)
<COMMENT>(This comment belongs to the Channel entry)
<x>(I/O<xs>channel)
<x>(channel)
<gdef>(Logical path connecting user process with a physical device unit
 to allow communication to the unit.)
```

3.3 Sort order

The <GTERM> tags can be coded in upper or lowercase or mixed. The glossary entries are all sorted by their uppercase equivalent. However, the output retains the original representation.

The sorting mechanism has been implemented similarly to the `/INDEX` keywords of DOCUMENT. If you don't specify anything there, you don't specify anything with `DOCTOR/GLOSSARY` to keep consistent sorting in index and glossary. The following options are possible, which are invoked by specifying keywords to the `/SORT` qualifier:

/SORT=	Description
WORD	Sorts a term letter by letter (also non-alphabetical characters in the middle of words) and treats spaces and hyphens as significant. Mutual exclusive with LETTER

Digital Internal Use Only

/SORT=	Description
LETTER	Sorts a term letter by letter (also non-alphabetical characters in the middle of words), ignoring spaces and hyphens. This is the default. Mutual exclusive with WORD
NONALPHA=AFTER	Place terms starting with non-alphabetic characters at the end of the glossary (following "Z")
NONALPHA=BEFORE	Place terms starting with non-alphabetic characters at the start of the glossary (preceding "A")
NONALPHA=IGNORE	Ignore nonalphabetic characters. E.g. "69WAYS" is sorted as "WAYS". This is the default.

The default `/INDEX=(LETTER,NONALPHA=IGNORE)` is identical to the DOCUMENT default.

Some attempt is made to translate multinational characters back into "english characters" without punctuation:

Any occurrence in the first string (match string) is replaced by the character in the second string (translation string):

```
'ÄÁÀÅÂÃÆÇÈÉÊËÌÍÎÏÑÒÓÔÕÖ×ØÙÚÛÜÝ'
'AAAAAAACEEEEIIIINOOOOOOOUUUUY'
```

3.4 Options

There are few options available on the sorting mechanism. The qualifier `/OUTPUT` allows to specify another output file. By default the next higher version of the input file is created. A warning that a purge may delete the original file is issued in that case.

Digital Internal Use Only

4 DOCTORing VMS Message source files

4.1 Overview

When a software product is coded and one uses the VAX/VMS error message utility to produce error messages on SYS$ERROR when the software detects an error (like the %DOCTOR messages you get from DOCTOR), the best place to describe the reason of the error message occurring, and what the user could possibly do to prevent it next time, is to write this information immediately when the message is defined. That's when you know best which message is added, why it was added and what you could do about it.

The DOCTOR/MESSAGE utility will format a properly coded source file with VAX/VMS message definitions and embedded comments in such a way that the comments of the file are reformatted into a <MESSAGE_SECTION> section, ready for inclusion into the User's Guide message appendix of the documentation that describes the software to which these error messages belong.

Note:

If you use DOCTOR/MESSAGE/HELP then the produced .SDML file is compatible for processing with the HELP.MESSAGE doctype and the MDFI destination, that is currently under development by CUIP. This will result in a .MDFI help file for the Alpha VMS and Open VMS utility HELP/MESSAGE that gives online access to the message explanation of any base or layered product running on that platform. See note 6.* in the SQM::BLADE_INFO notes file for more details on this utility.

The produced SDML file can be <INCLUDE>d into a VAX DOCUMENT source that is processed for the SOFTWARE family doctype.

The error messages found in Appendix A are made by DOCTOR/MESSAGE from the message file that belongs to the DOCTOR utility sources, and is included into this User's Guide by

```
<APPENDIX>(Error messages\apperror)
<include>(DOCTOR_MSG.SDML)
<ENDAPPENDIX>
```

You can look there to get an impression on the final output.

Digital Internal Use Only

4.2 Commenting requirements for a message file

The VAX Message file must conform to the VAX Message Utility standard. This means that each message definition is one line and upto 255 characters in length. Anything following an exclamation mark ("!") on a line is considered comment or (if part of the message text) a FAO directive.

`DOCTOR/MESSAGE` will use comment lines to allow the programmer to describe the reasons for the error message to be signalled to the user and possible user actions to avoid this situation from happening again. These comment lines must adhere to a certain structure that will be explained next.

`DOCTOR/MESSAGE` will start by reading all the facility definition lines (`.FACILITY`) and sort them alphabetically. Each facility will produce a <HEAD1>(facility name) section in the output file.

Next, it will read all the symbolic message names and also sort them alphabetically.

Finally, it will scan through the comment lines following a message line and reformat it into VAX DOCUMENT. The comments following a message line are considered to be describing that message. They are only recognized by `DOCTOR/MESSAGE` when they are written in a comment block that starts with "`!+`" and ends with "`!-`".

A small example of the message file source code to the `DOCTOR` utility and the resulting message code for this User's Guide is shown in Example 4–1 and Example 4–2.

Some comments to the original message source file and the produced `.SDML` file:

❶ The `.FACILITY` is used to compose a <HEAD1>(facility title\MSG_facility) section tag. They are sorted alphabetically.

❷ The severity level will be printed with each message description as a <MSG_TEXT>(Severity:) entry.

❸ The message ident and string is used to compose the message entry using the <MSG> tag.

❹ The comment block following the message definition is split in comment entries. Each entry starts with a short header. `DOCTOR/MESSAGE` interprets each comment line. If a colon is encountered, the first part of the line is made into a <MSG_TEXT> entry, the remainder of the line (plus possible subsequent lines) form the body of the message entry description.

❺ Same as previous entry. Following the Digital standard for message descriptions, two entries are made per message: an **Explanation** and **User Action** section.

Digital Internal Use Only

Example 4–1 Sample Message source file

```
.TITLE doctor_msg
.IDENT 'V1.0'
.FACILITY doctor,1/PREFIX=doc_                    ❶
.SEVERITY SUCCESS
.BASE 1

.SEVERITY INFORMATIONAL                           ❷
.BASE 100
IDENT    <This is !AS version !AS>/FAO=2           ❸
  !+
  !❹Explanation: Indicates the current version of the utility that is invoked.
  !   This is important when bugs and wishes are expressed to the development
  !   team.
  !
  !❺User Action: Specify number when submitting SPR's or QAR's.
  !-
NOTIMPL  <Functionality !AS not yet implemented>/FAO=1
  !+
  ! Explanation: The specified functionality is not yet implemented, even
  !   though the command interface is already present.
  !
  ! User Action: Wait till the next release or inquire at
  !       NSIC00<colon><colon>KLERK. ❻
  !-
SENDQAR    <Internal error. Please send QAR. Error ID: !AS>/FAO=1
  !+
  ! Explanation: You managed to give a set of input data that caused the
  !   utility program to go to "catch all" statements that should never
  !   be reached.
  !
  ! User Action: Please notify the developers by SPR or QAR, describing
  !   the error and specifying the version of the software used, the
  !   error messages displayed and possibly a sample of the input files
  !   that caused the error to occur, together with the specified command
  !   statement.
  !
  !-
```

❻ If you want to use the colon (:) as part of the text without DOCTOR/MESSAGE interpreting this as an entry heading, use the tag <COLON> which is defined by DOCTOR/MESSAGE at the start of the message section output and undefined at the end of it (so it is *not* a standard DOCUMENT tag!).

❼ Definition of <COLON>

❽ Undefine of <COLON>

Digital Internal Use Only

Example 4–2 Sample DOCTOR/MESSAGE output

```
<HEAD1>(DOCTOR Messages\MSG_DOCTOR)      ❶
<MESSAGE_SECTION>
<DEFINE>(COLON\:\0\0\\\:)  ❼
<MSG>(IDENT\This is <EMPHASIS>(string) version <EMPHASIS>(string))❸
<X>(IDENT)
<MSG_TEXT>(Severity)  ❷
INFORMATIONAL
<MSG_TEXT>(Explanation)      ❹
Indicates the current version of the utility that is invoked.
This is important when bugs and wishes are expressed to the development
team.
<MSG_TEXT>(User Action)  ❺
Specify number when submitting SPR's or QAR's.

<MSG>(NOTIMPL\Functionality <EMPHASIS>(string) not yet implemented)
<X>(NOTIMPL)
<MSG_TEXT>(Severity)
INFORMATIONAL
<MSG_TEXT>(Explanation)
The specified functionality is not yet implemented, even
though the command interface is already present.
<MSG_TEXT>(User Action)
Wait till the next release or inquire at
NSIC00<colon><colon>KLERK.  ❻

<MSG>(SENDQAR\Internal error. Please send QAR. Error ID: <EMPHASIS>(string))
<X>(SENDQAR)
<MSG_TEXT>(Severity)
INFORMATIONAL
<MSG_TEXT>(Explanation)
You managed to give a set of input data that caused the
utility program to go to "catch all" statements that should never
be reached.
<MSG_TEXT>(User Action)
Please notify the developers by SPR or QAR, describing
the error and specifying the version of the software used, the
error messages displayed and possibly a sample of the input files
that caused the error to occur, together with the specified command
statement.

<UNDEFINE>(COLON)  ❽
<ENDMESSAGE_SECTION>
```

The following lines are required by both DOCTOR/MESSAGE and the VAX Message utility to work properly. Note that you should not run DOCTOR/MESSAGE on a source file without having been able to properly compile the source by the MESSAGE command. DOCTOR/MESSAGE does not do the same intensive checking as MESSAGE and partly relies on correct syntax.

Digital Internal Use Only

.FACILITY	Once for each set of facility messages. The facility name is used to create the <HEAD1> heading for that message section. Abbreviated forms of the word .FACILITY are allowed.
.SEVERITY	Required to specify the type of severity of the message described. It is used by DOCTOR/MESSAGE to write this information in the **Severity:** part of the output. Each message can also have its own severity indicated using the /WARNING, /SUCCESS etc. qualifiers attached to the message definition line. Abbreviated forms of the word .SEVERITY are allowed

message line	Formatted as
	`messagename  < text >/qualifiers`
	`messagename  " text "/qualifiers`
	where the *text* message is surrounded by either quotation marks or angle brackets. This line is output literally into the DOCTOR/MESSAGE output file, sorted alphabetically on message name. By default, the FAO directives are replaced by generic strings like *string* or *number*, depending on the FAO directive encountered. Also UIC, time, date and other directives are properly substituted. If this is not wanted, specify DOCTOR/MESSAGE/FAO on the command line.

4.3 Include result in documentation

The resulting message document file can be included into appendices of the user publications or made part of the Detailed Design Specification. Both will then have an identical format.

As such, the following use of the message output is suggested:

```
<APPENDIX>(Error messages)
<INCLUDE>(messagefile.SDML)
    :
  additional message files
    :
<ENDAPPENDIX>
```

4.4 DOCTOR/MESSAGE Features on Message Files

The comments related to a message are recognized by DOCTOR/MESSAGE only if they are written within a special comment block, formatted as

```
!+
!  header text: additional text
!
!-
```

This comment block should *follow* the VMS message definition immediately.

Digital Internal Use Only

The *header text* is output as a boldly written set of words, followed by one or more lines of additional text. The separation between *header text* and the remaining text is made by the colon (":"). As such the additional text should not use this character. If the colon is required as part of the explanatory text, then use the <COLON> tag.

To conform to the Digital standard way of documenting error messages, the following use is suggested:

```
message_name  <error message text> /qualifiers
   !+
   ! Explanation:   text
   !
   ! User Action:   text
   !-
   !  additional comments not meant for DOCTOR/MESSAGE
   !
```

Note that any comment written outside a comment block delimited by the "+" and "–" will not be seen by `DOCTOR/MESSAGE`. In addition, the entire message can be omitted from the SDML output if the message line itself contains a comment trailer that contains either `!*` or `!*Skip*` as text:

```
message_name  <error message text> /qualifiers  !*SKIP*
   !+
   ! Explanation:   text
   !
   ! User Action:   text
   !-
   !  Despite the !+ and !- marked comment block, the entire message
   !  is not copied into the SDML output file due to the !*SKIP* comment
   !  on the message line.
   !
```

5 DOCTORing SDML files for Bookreader output

5.1 Overview

VAX DOCUMENT can be used to build books for printing on paper, but also for using them as online books using the DECwindows Bookreader on either VMS or ULTRIX.

Unfortunately, the Bookreader product imposes a more strict use of some of the tags, available in DOCUMENT, due to its unique features of "pop up" elements (tables, figures, examples) that appear when you click on a "hot spot".

Because any part of the document must be accessible from the table of contents, a symbol must be attached to them so an implicit reference is made from the table of contents to the section pointed at.

DOCTOR/ONLINE will go through your SDML files and add symbols where there are none, and also add references to pop up elements, where none is written in the text. If you specify a profile file, all elements will also be searched, as will any file referenced in a <INCLUDE> tag. Specifying /LOG will show you which files are accessed.

The following tags are modified by DOCTOR/ONLINE:

- <HEAD>, <CHEAD>, <HEADN>, <CHAPTER>, <GLOSSARY>, <PART>, <APPENDIX>, <FRONT_MATTER> - a symbol is added if not present
- <SUBHEADN>, <PREFACE>, <PREFACE_SECTION>, <COMMAND>, <ROUTINE>, <SDML_TAG>, <STATEMENT>, <SUBCOMMAND> - a symbol is added unless /VERSION_1 is specified.
- formal <EXAMPLE>, <TABLE>, <FIGURE> - a symbol is added if not present, a <REFERENCE> to them is created
- <FIGURE_FILE>(book\..\..) is added to supplement existing figures for other destinations.

The following tags are used by DOCTOR/ONLINE:

- <ELEMENT>, <INCLUDES_FILE>, <INCLUDE> - to read these files as well during the processing
- <FIGURE_FILE> - to create a similar one for the BOOK destination.
- <REFERENCE> to track down all references to symbols
- <DEFINE_SYMBOL> which defines additional symbols that may be referenced by <REFERENCE>.
- <DEFINE_BOOK_NAME> which defines a symbol for a user specified book title

Digital Internal Use Only

- <SET_CONDITION>, <CONDITION> and <ENDCONDITION> tags. During Pass 1, when all symbol information is collected, DOCTOR will skip any conditionalized text block whose condition does not match the current conditional setting. It will however, generate symbols for all tags that need one, regardless of condition.

To prepare files for use with DOCUMENT V1.2B you should specify /VERSION_1. No harm done if you forgot: just reprocess those files with /VERSION_1 and any symbol added on tags that should not have a symbol for V1.2B will be removed again.

5.2 Example of ONLINE conversion

An example of original input versus converted output file can be seen in Example 5–1, Original file and Example 5–2, Modified file.

Example 5–1 Original file

```
<chapter>(unsymboled)
<head1>(section head1\s1_1)

<p>
Test for V1.2B and V2 differences: it should not add symbols for V1.2B
<subhead1>(A subheader)

It should not add one if it had one
<subhead1>(A subheader with symbol\subheadsymbol)

Test for informal figures - should add figurefile only
<figure>
<figure_file>(ps\fig.ps\10)
<endfigure>

Test for symbol adding, reference adding and file adding
<figure>(figure caption)
<figure_file>(ln03\ln03fig.six\10)
<endfigure>

Test for same, no reference adding
<reference>(f1_1)
<figure>(figure caption\f1_1)
<figure_file>(ln03\ln03fig.six\10)
<endfigure>

Test for same, no reference adding, no file adding
<reference>(f1_2)
<figure>(figure caption\f1_2)
<figure_file>(ln03\ln03fig.six\10)
<figure_file>(book\bookfig.fse\10)
<endfigure>
```

Example 5–1 (continued on next page)

Digital Internal Use Only

Example 5–1 (Cont.) Original file

```
Test for same, reference adding only
<figure>(figure caption\f1_3)
<figure_file>(ln03\ln03fig.six\10)
<figure_file>(book\bookfig.fse\10)
<endfigure>

Test for same, reference adding only again
<figure>(figure caption\f1_4)
<FIGURE_SPACE>(42\left blank)
<endfigure>

Test for same, reference adding only again
<figure>(figure caption\f1_5)
<line_art>
   something
<endline_art>
<endfigure>

Add symbol and reference to table.
<table>(small table)
<table_setup>(2\10)
<table_row>( 1 \ the only row)
<endtable>

<head1>(section head1\s1_2)

<head1>(section head1)
```

Example 5–2 Modified file

```
<chapter>(unsymboled\BK_ADDED_1)
<head1>(section head1\S1_1)

<p>
Test for V1.2B and V2 differences: it should not add symbols for V1.2B
<subhead1>(A subheader\BK_ADDED_2)

It should not add one if it had one
<subhead1>(A subheader with symbol\SUBHEADSYMBOL)

Test for informal figures - should add figurefile only
<figure>
<figure_file>(ps\fig.ps\10)
<FIGURE_FILE>(BOOK\FIG.fse\10)
<endfigure>
```

Example 5–2 (continued on next page)

Digital Internal Use Only

Example 5–2 (Cont.) Modified file

```
Test for symbol adding, reference adding and file adding
<figure>(figure caption\BK_ADDED_3)
<figure_file>(ln03\ln03fig.six\10)
<FIGURE_FILE>(BOOK\LN03FIG.fse\10)
<endfigure>

<CP><REFERENCE>(BK_ADDED_3\FULL)

Test for same, no reference adding
<reference>(f1_1)
<figure>(figure caption\F1_1)
<figure_file>(ln03\ln03fig.six\10)
<FIGURE_FILE>(BOOK\LN03FIG.fse\10)
<endfigure>

Test for same, no reference adding, no file adding
<reference>(f1_2)
<figure>(figure caption\F1_2)
<figure_file>(ln03\ln03fig.six\10)
<figure_file>(book\bookfig.fse\10)
<endfigure>

Test for same, reference adding only
<figure>(figure caption\F1_3)
<figure_file>(ln03\ln03fig.six\10)
<figure_file>(book\bookfig.fse\10)
<endfigure>

<CP><REFERENCE>(F1_3\FULL)

Test for same, reference adding only again
<figure>(figure caption\F1_4)
<FIGURE_SPACE>(42\left blank)
<endfigure>

<CP><REFERENCE>(F1_4\FULL)

Test for same, reference adding only again
<figure>(figure caption\F1_5)
<line_art>
   something
<endline_art>
<endfigure>

<CP><REFERENCE>(F1_5\FULL)

Add symbol and reference to table.
<table>(small table\BK_ADDED_4)
<table_setup>(2\10)
<table_row>( 1 \ the only row)
<endtable>

<CP><REFERENCE>(BK_ADDED_4\FULL)

<head1>(section head1\S1_2)
```

Example 5–2 (continued on next page)

Digital Internal Use Only

Example 5–2 (Cont.) Modified file

```
<head1>(section head1\BK_ADDED_5)
```

5.3 Including predefined symbols

In many cases, a book also uses a set of predefined symbols, that are made known to DOCUMENT through the DOCUMENT/SYMBOL=symbolfile command qualifier. These symbols will also be taken into account by DOCTOR if the /SYMBOL_FILE qualifier is also used on the DOCTOR command line:

```
DOCTOR/ONLINE/SYMBOL=symbolfilespec inputfile.SDML
```

This way it is avoided that those predefined symbols are listed as unknown in the list file that is produced if /LIST is also specified.

See also Section 5.4 for additional information on the combination of predefined symbols and additionally generated ones.

5.4 Generating additional symbols

Apart from providing all necessary section tags with symbols where the author did not specify these, it is also possible that a number of <REFERENCE> tags are used that refer to undefined sections. Either because there was a typing error, or because a section was removed or because the section has yet to be written.

When DOCUMENT processes a source file that contains more than 30 undefined references (or any other warning message), it will abort after the 30th message. To avoid this, you can instruct DOCTOR/ONLINE to generate an additional file that produces values for all these undefined symbols: specify the /DUMMY_SYMBOL qualifier.

It then produces a file with <DEFINE_SYMBOL> tags where the value of each symbol is simply set to the value <TAG>(reference) tag. When the source document is now processed with DOCUMENT, the final text will print "<REFERENCE>(*undefined_symbol*)". If this text is unwanted, you may define your own *undefined_symbol* text using the /DUMMY_SYMBOL=(TEXT="*string*") keyword.

The processing of the entire document is now possible, despite the undefined symbol references by the command sequence:

```
DOCTOR/ONLINE/DUMMY_SYMBOL=(OUTPUT=dummy_symbol_file, TEXT="string") -
      inputfile.SDML
DOCUMENT/SYMBOL=dummy_symbol_file inputfile.SDML doctype destination
```

To generate such a symbol file, the qualifier /DUMMY_SYMBOLS is required. When no keyword OUTPUT= is specified, it defaults to input_file_name.SDML_DUMMY_SYMBOLS.

Digital Internal Use Only

During the collection of defined symbols (in Pass 1), DOCTOR obeys the setting of conditional tags. If a conditionalized block of text does not match the current condition, it is ignored and all symbols defined in that block are also not seen, i.e. remain undefined and hence will be output to the dummy symbol file. You can set a specific condition through either the <SET_CONDITION> inside the SDML file or through the DOCTOR /ONLINE /CONDITION= qualifier.

When the /SYMBOL qualifier is also specified, the symbols defined in that symbol file are not copied into the dummy symbol file. If you want to include a <INCLUDE>(symbol_file_spec) into the generated dummy symbol file to be sure you include both, then the keyword /DUMMY=(INCLUDE_SYMBOLS) is required. This is useful if you want to process the final document using both the predefined symbols and the generated dummy ones. Because only a single /SYMBOLS= qualifier is allowed on the DOCUMENT command line, you can specify the generated dummy symbol file, which now automatically references the predefined symbols through a <INCLUDE> tag.

5.5 Things to remember

5.5.1 Using /OUTPUT

Using the /OUTPUT= qualifier allows you to let DOCTOR create modified output files with a different name or extension than the orginals. This allows you to keep the original files untouched. One of three approaches is suggested:

1 Specify /OUTPUT=*disk:[dir]* only. This will move all modified output files into a separate directory. You can attempt to build the online books from there. Once they are coded the way you want it, you can delete the originals. This is the prefered way.

2 Specify /OUTPUT=*.filetype* only. This will create the modified files in the same directory, but they all have a different file extension, e.g. .SDML_BOOK. Once all files are coded to your liking, you can simply RENAME them into normal SDML files. Processing all files with modified file types may cause problems, as the <INCLUDE> and <ELEMENT> tags inside the files still refer to the original files.

3 Specify /OUTPUT=[]. This will create new versions of the files in your current directory with the same name and file extension as the originals. If originals are also in your default directory then obviously the new created ones supersede the originals.

If you specify /OUTPUT=.SDML the output will always be in the original directory of the sources and the new versions will supersede the old ones.

Note:

When no /OUTPUT is specified, DOCTOR/ONLINE will use the file type .SDML_ONLINE, to prevent unintentional loss of the originals by a purge.

Digital Internal Use Only

5.5.2 Using /VERSION_1

There is a distinct difference in processing between VAX DOCUMENT V1.2B and V2.0 or higher. The V2+ allows a symbol on the <SUBHEAD*>, <PREFACE>, <PREFACE_SECTION>, <COMMAND>, <ROUTINE>, <SDML_TAG>, <STATEMENT> and <SUBCOMMAND> tags, whereas V1.2B does not. If you specify `/VERSION_1`, no symbols are added. In fact, if any is encountered, it is removed and placed inside a <COMMENT> tag. Therefore, if by accident you processed files for V2 (the default) any symbols that should not be added for V1.2B can be removed by reprocessing the same file with `/VERSION_1`.

5.6 Adding symbols

Each section tag such as the <HEAD*>(header) tag, without a symbol will be given one. This is done by adding a unique symbol `BK_ADDED_nnn` where *nnn* is a generated number. Also files processed before can be re-processed as `DOCTOR/ONLINE` first looks for the presence of any of those tags, and will generate new ones which start from the highest found number *nnn+1*. These symbols are generated for all tags that require one, regardless of whether they are embedded in a conditional text block whose condition is not met.

5.7 Adding references

While the DOCTOR searches through the files, it registers all symbols found and tries to match them with the occurence of at least one <REFERENCE> tag. If the document is not ready for Bookreader destination processing, DOCTOR will have found some symbols that have no references. If this is the case with either a formal table, example or figure, these elements cannot be "popped up" by clicking on a hotspot, as these only exist when a <REFERENCE> is coded.

To resolve this, DOCTOR will add a <REFERENCE> to these pop up elements immediately following the <ENDFIGURE>, <ENDEXAMPLE> or <ENDTABLE>. It will not attempt to embed this reference in a sentence: you may still want to do this by yourself. The list file that can be generated contains the instructions of where manual intervention may be required.

5.8 Adding figure files

Because figures may have been specified for known printer destinations, but not for online books, the DOCTOR will also add a <FIGURE_FILE> within a figure block if no figure for the `BOOK` destination is found. Obviously, DOCTOR will not generate such a figure. You will need to create it yourself or try to convert the existing figure into the FSE format required by the Bookreader. RAGS and UTOX are two tools that can assist you in this process. RAGS can make figures for all valid destination types of VAX DOCUMENT, UTOX can convert several figure file types into others. However, if a sixel of PostScript file is all you got of a figure, you will have to re-draw it. Use RAGS.

Digital Internal Use Only

5.9 What must you do?

If you specify the /LIST qualifier to DOCTOR/ONLINE, you will get a list file that contains all the actions you still must perform after the documents have been converted.

5.10 Removing or renewing added symbols

Occassionally it will happen that you want to remove all automatically generated symbols. This may be the case when an existing piece of text needs to be incorporated in another document. Or when a book is created of individual elements which at some point in time were all processed individually. In both cases chances are that the document elements contain identical symbols that were generated.

DOCTOR allows you two options to correct this situation:

1 DOCTOR/ONLINE/REMOVE removes all symbols generated by any previous processing of a file with DOCTOR/ONLINE. It removes the symbols during Pass 1 processing and then stops.

2 DOCTOR/ONLINE/SUPERSEDE also removes all generated symbols during Pass 1 but continues with Pass 2 to generate new ones, that are consistent throughout the document. In this process files without generated symbols are created during Pass 1 in the specified output directory (/OUTPUT=) and these are then used to scan for adding new symbols during Pass 2. Therefore you will end up with two new versions of the original files: the newest with new symbols, the one but newest without any generated symbols.

Note:

Due to the fact that VAX SCAN is unable to retrieve the specific file version it creates it was decided not to delete this intermediate file without symbols automatically as DOCTOR can never be deadsure it deletes the right version always. And we'd rather be safe than sorry.

6 DOCTORing PostScript files

6.1 Overview

PostScript files are produced by a wide variety of products, such as VAX DOCUMENT, DECwrite, DECpaint or MS-Word. With the arrival of laser printers lots of documentation is produced in the PostScript format and printed. Several problems can then occur:

- A file print is halfway ready when the system crashes or the job gets aborted for all the right and wrong reasons.
- Only a section of a document is interesting to print.
- Some products, like VAX DOCUMENT, do not produce blank pages when a section ends on an odd page. This causes double sided printers to skip a page and print the following section on the wrong side of the page (recto pages on verso pages and vice versa).
- The same file is needed, but one would like a different standard page layout (e.g. with DRAFT written over the page) as defined in the prolog of the PostScript file.
- The PostScript file contains figures that were inserted as *encapsulated PostScript* and you would like to remove those from the file or to create them as separate files to re-use those figures.
- A PostScript document should be re-ordered in page output to allow for saddle stiching of double sided printed sheets (4 pages/sheet, like a magazine with pages folded in the middle).

The DOCTOR utility allows to do all this in an easy manner, provided the PostScript file that must be inspected adheres to Adobe's minimal conformant coding style, as described in Section 6.7. This is assumed to be true if the first line of the PostScript file contains `%!PS-Adobe`, possible followed by a version number.

DOCTOR also adheres to the coding standard itself. It prefixes the file output with the following standard banner, that follows the `%!PS-Adobe` header line:

Digital Internal Use Only

```
%!PS-Adobe-2.0
%%Creator:   VAX DOCUMENT V2.0 modified by DOCTOR V3.0
%%
%%CreationDate: 20-JUL-1992 13:21:58.92
%%
%% Command issued: DOCTOR/PS DOCTOR.PS/OUT=[]/BLANK
%% File used:      PSQ:[PUBLIC.SOURCES.DOCTOR]DOCTOR.PS;4
%% QAR's to:       Theo de Klerk @UTO,  NSIC00::KLERK
%%
%%     *** Digital Internal Use Only ***
%%
```

Because this header preceeds any header that is copied from the original input file, the above entries overrule identical header entries. The original header line indicating its creator has been kept, but appended with the DOCTOR version. This is to allow other tools to still retrieve from this line the origin of the PostScript code.

Important:

It is advised always to use the /OUTPUT qualifier to specify the output file specification. If omitted, the DOCTOR will make the next higher version of the input file. This is potentially dangerous as one might accidentially delete this original file through purging.

6.2 Adding blank pages

Using the /BLANK qualifier will allow you to insert additional pages into the PostScript file to balance the number of odd/even pages. This is especially important for VAX DOCUMENT files that, by default, do not include any blank pages if a chapter ends on an odd page.

This will make sure that the output document, when printed on a double sided printing device will have all recto and verso pages printed in the correct orientation. This is also important on single side printing devices, where the final output will be stacked into a double sided printing photocopier machine for further duplication. The inserted blank page is indeed blank: no page numbers or anything is output to the page, as DOCTOR does not know how the page layout is composed.

Restriction for update pages:

DOCTOR/BLANK inspects the %%Page: comments in the PostScript file and determines from the last numeric part of the folio number (such as "Glossary-10") whether or not a page needs to be inserted. It can only make this decision properly for normally formatted books. If the book is built using update pages, the end result may be incorrect, as DOCTOR cannot make the right decision on page jumps like 4-3, 4-3.1 or 4-3.1, 4-4. The last digits are all odd, assuming that a blank even page needs to be inserted. Depending on your own feelings, this may or may not be desired.

If you plan to print the PostScript file in a reduced format (with two pages on one side of a sheet of paper), you should also use the /LEADING_BLANK qualifier. This inserts a blank page before the first page of the document output, to ensure that the positioning of recto and verso pages is not changed: the first page of a document is a right hand

side page, but would occupy the left half of the page when /LEADING_BLANK had not been used.

When the /LOG qualifier is also enabled, the DOCTOR will tell you when a page is inserted. For example:

```
$ DOCTOR/PS DOW.PS/BLANK/EXTRACT=(START=7-1,NUMBER=4)/OUT=TEMP.PS/LOG
%DOCTOR-I-IDENT, This is DOCTOR version V3.0
%DOCTOR-I-CONFORM, File assumed to conform to %!PS-Adobe-3.0
%DOCTOR-I-INSERTED, Page 7-2  inserted
%DOCTOR-I-INSERTED, Page 8-2  inserted
%DOCTOR-I-INSERTED, Page A-2  inserted
%DOCTOR-I-INSERTED, Page B-2  inserted
%DOCTOR-S-CREATED, Created PSQ:[DOCTOR]TEMP.PS;1
```

where the chapters 7 and 8 only had a single page (7-1 and 8-1), as had the two appendices A and B.

Each inserted blank page only results in the insertion of the following PostScript code for VAX DOCUMENT generated files (Adobe V3 output):

```
%%
%%Page: (7-2) 2   (the added folio page and ordinal number)
%%     INSERTED BLANK PAGE
%%PageFonts: (atend)
%%PageProcessColors: (atend)
%%PageCustomColors: (atend)
PaperHeight PaperWidth PM 0 0 XY
EP PP
%%PageTrailer
%%PageFonts: Helvetica-Bold
%%PageProcessColors:
%%PageCustomColors: (BLACK)
%
```

The same file, made with Adobe V2 standards by DOCUMENT V2.0 or before, results in

```
%%Page: 7-2 2   (the added folio page and ordinal number)
%%     INSERTED BLANK PAGE
%%BeginPageSetup
%%EndPageSetup
%%PageFonts: (atend)
%%PageCustomColors: (atend)
1000 BP PaperHeight PaperWidth PM 0 0 XY
%%BeginCustomColor: 0_BLACK
%%EndCustomColor: 0
6 PP EP
%%PageTrailer
%%PageFonts: Helvetica-Bold
%%PageCustomColors: 0_BLACK
%
```

whereas any other PostScript file produced by a product other than VAX DOCUMENT will result in

Digital Internal Use Only

```
%%Page: 7-2 2
%%     INSERTED BLANK PAGE
showpage
%
```

Note:

The Adobe comment standard also allows for pages to be marked as `%%Page: text 5` where *text* can be any string with no white space. It need not have a numeric part therefore. In this case obviously DOCTOR is not able to determine where a new blank page is needed or not, since there is no page number information available to determine whether a page is skipped.

User specified blank pages

The `/BLANK` qualifier allows the specification of a separate file specification, that can contain user-specified instructions to create an additional page. This would overrule the default blank page output as specified in the previous section.

The user specified file must contain all the relevant PostScript instructions to create the blank page to be inserted. This could include instructions to typset *This page has been left blank intentionally* or something similar. The instructions should contain the `showpage` command to output the page during printing. It should not contain a page header in the form of `%%Page:` as the internal page folio numbering and ordinal numbering is performed by the DOCTOR itself.

When a part of a document is extracted, the `/BLANK` may result in an extra blank page preceeding the extracted section (if it starts on an even page) or one following the section if it ended on an odd page.

The only time this may cause two blank pages is where two consecutive extracted ranges both have a page appended and preceeded respectively. This ensures different sections are always on different physical pages.

Note that the `/LEADING_BLANK` *page will not use the user specified file for blank pages, as this leading page is not considered part of the document, but rather a placeholder before the document.*

6.3 Replacing the prolog

The prolog of each PostScript file is supposed to follow the initial header section and either starts at the first line that does not start with `%!` or `%%` or after the first line that contains `%%EndComments`. It continues until there is an explicit `%%EndProlog` line. This line must be present, even if no prolog is specified.

The prolog contains definitions and a general page layout applicable to all pages. Hence, you can add or replace certain features to give the printed pages a different look, without modifying any of the text that is part of the document itself. You can create your own prolog, either based on the original one, or entirely homewritten, to replace the prolog that comes with the input file. This way you can produce effects as writing "DRAFT" diagonally across all pages, or print a faint logo on each page or border the text in a frame.

The user specified prolog file must end with the %%EndProlog statement.

To allow some simple modification, the qualifier /CHANGE_PROLOG has been implemented. You can specify three items on each page:

- A bold printed text at the top of each page (a header)
- A bold printed text at the bottom of each page (a footer), with or without an additional (page)number added
- A faintly grey printed text diagonally across the page. The grey scale can be set manually.

These three text items do not interfere with the original text in the PostScript file. You could imagine it as printing this file on pre-printed paper that had those three items on it already. The text on the diagonal line is dynamically adjusted in size so that the entire text fits on a single line running from the bottom lefthand corner to the top righthand corner.

The /CHANGE_PROLOG comes with twelve keywords, all of which are optional, but at least one must be specified:

```
/CHANGE_PROLOG=(TOP="text", -
                BOTTOM="text", COUNTER=integer, -
                DIAGONAL="text", -
                BORDER, -
                FONT=font_type, SIZE=number, GRAYSCALE=number, -
                OUTPUT=file_spec, -
                HSIZE=pointsize, VSIZE=pointsize -
                PAGESIZE=papertype)
```

The first four arguments specify the text to be used for the header, footer or diagonal line. If a space occurs in this text, the entire text must be enclosed within quotation marks. The specified texts are used for the prolog in the output PostScript file.

The COUNTER argument allows you to specify an integer value that will be printed on the bottom line, following the specified BOTTOM text. The counter will increment with each page. It allows you to number pages that would otherwise be unnumbered and you can specify the starting value of the page numbering sequence.

The GRAYSCALE allows to specify how dark the grey diagonal text must be. It defaults to 95%, which is light grey. A value of 00 would make it ink-black and 99 almost invisibly white. The diagonal text will auto-size to fit the entire length of the diagonal. By default the text is printed in Helvetica-Bold, and the top and bottom lines are sized to 15 points. However, you may specify a different font using the FONT= keyword and the SIZE= keyword. Although any size (in point units) is accepted, values above 20 points seem (no pun intended) pointless. The fonts that can be specified are listed in Table 6–1.

Digital Internal Use Only

Table 6–1 FONT keyword values

Note that all PostScript font specification hyphens are replaced by underscores for the DCL keyword
- Times_Roman
- Times_Bold
- Times_BoldItalic
- Times_Italic
- NewCenturySchlbk_Roman
- NewCenturySchlbk_Bold
- NewCenturySchlbk_BoldItalic
- NewCenturySchlbk_Italic
- Helvetica
- Helvetica_Bold
- Helvetica_BoldOblique
- Helvetica_Oblique
- Courier
- Courier_Bold
- Courier_BoldOblique
- Courier_Oblique
- AvantGarde_Book
- AvantGarde_Demi
- AvantGarde_DemiOblique
- AvantGarde_BookOblique
- LubalinGraph_Book
- LubalinGraph_Demi
- LubalinGraph_DemiOblique
- LubalinGraph_BookOblique
- Souvenir_Light
- Souvenir_Demi
- Souvenir_DemiItalic
- Souvenir_LightItalic

A border may be printed around the page specifying the `BORDER` keyword. By default, no such rectangular border is drawn (Note that PostScript printers can produce the same if with printing the command `PRINT /PARA=(NUMBER=1)` is specified).

Digital Internal Use Only

The OUTPUT keyword allows the specification of a file specification. In that case, in addition to using those header, footer and diagonal line texts in the output PostScript file, a separate prolog file is created containing those same definitions for header, footer and diagonal. This created prolog file can later be used with the /PROLOG qualifier to replace the standard prolog with this newly created one.

The two keywords HSIZE and VSIZE are only needed if the prolog must be made to fit a specific page size. By default A4 size is used. Any other dimension can be specified in point units.

Alternatively, you can use the keyword PAGESIZE that will recognizethe following standard paper sizes: LETTER, LEDGE R, LEGAL, EXECUTIVE, 7X9, 35MM, A5, A4, A3, B5, B4, C6, C5, C4. The keyword PAGESIZE is mutually exclusive to both HSIZE and VSIZE.

6.4 Extracting ranges

Using the /EXTRACT qualifier you can specify to copy only a range of pages from the input file into the output file. The first page to copy is indicated by the START=folio keyword, the last page either through END=folio keyword or the NUMBER=integer keyword. The specified folio numbers must exist in the file, as otherwise DOCTOR cannot synchronize on them.

Specifying an invalid start page means the entire section is skipped, specifying an invalid end page means the remainder of the document is included while DOCTOR is looking for the end page to match the invalid specified folio number. When both the END and NUMBER are specified and they indicate a different range, the first match is used, resulting in the shortest range between the first and last page of the extracted section.

You can specify several /EXTRACT qualifiers to allow for several ranges to be included in the same output file. The ranges should not overlap. In its search to end one range, the DOCTOR will read past the beginning of the next section and thereby will never encounter the starting page again of the section that overlapped.

The /LOG qualifier will indicate when a range is found and whether in the end some ranges were skipped.

The valid entries for START= and END= can be found by doing a

```
$ SEARCH filespec.PS %%Page:
```

command at DCL level. This will reveal all the page header lines of the available pages within the PostScript document. For Adobe V3 comment standards, the page numbers are enclosed within parentheses. You do not specify these parentheses in the START= and END= keywords of the /EXTRACT qualifier.

Important:

Many documents consist of text and graphics, made by different products and somehow combined into the final document. As each product produces its own %%Page: entries, some entries may seem out of order from the usual incremental numbering of the page sequence number (ordinal). When this is

Digital Internal Use Only

the case, DOCTOR ignores those page comments and assumes it is some sort of included graphic and not really a page. Therefore you cannot select one of those page numbers to select a range boundary.

Note:

The Adobe comment standard also allows for pages to be marked as `%%Page: text 5` where *text* does not contain a numeric part. In this case obviously DOCTOR is not able to determine what page ranges are available to extract.

6.5 Removing or re-using figures

A PostScript file contains text and possibly figures. Those figures are very often made separately (using a drawing package) and then inserted into the text body by the text formatting tool such as VAX DOCUMENT's <FIGURE_FILE> tag or DECwrite's "LINK TO PICTURE" option.

Occassionally there is a need to either remove the figures from the PostScript file (to allow the remainder to be converted back to plain ASCII text file) or to re-use those figures in other documents.

The extracting or removal of figures is based on the assumption that the input PostScript file contains the lines

```
%%BeginDocument:  figure_filespec
 ... code of the encapsulated PostScript figure ...
%%EndDocument
```

to mark the begin and end of each included encapsulated figure.

In Table 6–2 all possible combinations are summarized.

Table 6–2 Extracting or removing figures

Qualifier	Result
/FIGURE	Default. Leaves figures untouched
/FIGURE=EXTRACT	Extract figures into separate files by their original names. Figures also remain in input document
/FIGURE=(EXTRACT,PAGE_NUMBER)	Extract figures into separate files. These files have the same name as the input file spec, but appended to it is the page number on which the figure occurred.
/NOFIGURE	Removes figures from the input file.
/NOFIGURE=EXTRACT	Removes figures from the input file. Extract figures into separate files by their original names.
/NOFIGURE=(EXTRACT,PAGE_NUMBER)	Removes figures from the input file. Extract figures into separate files. These files have the same name as the input file spec, but appended to it is the page number on which the figure occurred.

Digital Internal Use Only

6.5.1 Removing figures

For some time now several tools are available to convert a PostScript file back into its plain ASCII text file. This can come handy if the sources to produce the PostScript file are lost or part of the document could be used in another file.

Within Digital there is a PS2TEXT utility to enable you to do this on several PostScript files, amongst which VAX DOCUMENT. However, if such a document contains figures made by some graphics package, these should be removed from the PostScript file before a conversion is attempted. Again DOCTOR comes to the rescue by allowing the /NOFIGURES qualifier that will remove all <FIGURE_FILE> included figures.

When the /LOG qualifier is specified, it outputs the names of the figure files that are removed from the output file.

6.5.2 Extracting figures

When you want to extract figures from a document into individual figure files, you can also use the /FIGURE qualifier. In this case, you need to add a keyword to it: EXTRACT. This will result in DOCTOR scanning the PostScript source file and to extract each included figure to become a separate .EPS file. This file can then be used in other documents or presentations.

There are two ways to specify a file name for the extracted figures:

- They get their original name back. This name is specified in the PostScript input file on the comment lines that start with

  ```
  %%BeginDocument: filespec
  ```

  The extracted figure files will be created in the current default directory, or the one that is indicated by the /OUTPUT qualifier. This manner may be useful for the original author of the document who has lost the original figure files.

  This behaviour is obtained by specifying /FIGURES=EXTRACT.

- They all get the name of the original input PostScript file, but appended to it is the page number on which the figure occurs. This makes it easy for those that want to use the original document as a reference to where the figure is printed.

  This behaviour is obtained by specifying /PAGES=(EXTRACT,PAGE_NUMBER).

If /LOG is also specified, the created figure file names are shown.

When the /EXTRACT qualifier is also used, only the figures in the selected page ranges are extracted. When the /NOFIGURE qualifier is used in combination with the keywords EXTRACT and/or PAGE_NUMBER, the figures are extracted into their individual files, and at the same time removed from the input document.

Digital Internal Use Only

6.6 Saddle stich printing

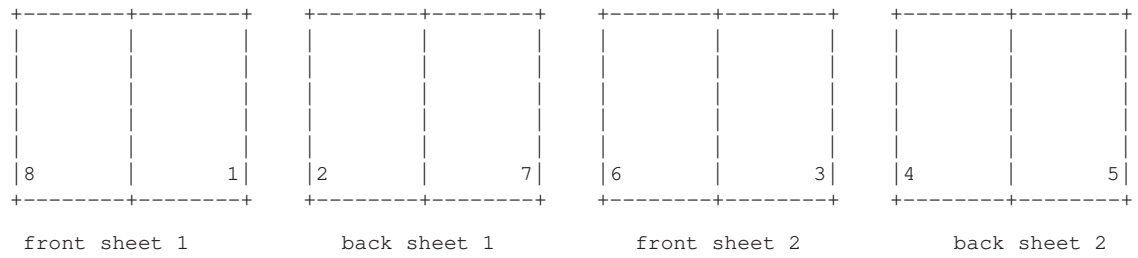
When one wants to use a printer that allows for printing on both sides of a sheet of paper, it may be advantageous to print the pages on half format (allowing two pages on a single sheet side) and then fold the pages in the middle to make a signature (like a magazine), ready for saddle stiching.

When the book is very thick, one may decide to divide the book into several of these signatures and then stich them together.

In both cases this requires the output order of the pages in the PostScript file to be modified. Rather than the usual sequential order of page 1, 2, 3 etc, we now need the first and the last page to be printed after each other (and on the same sheet side if printing is setup for two pages/sheet). And then page 2 and the one-but-last page on the backside of the sheet.

As a small example: suppose 8 pages A4 must be printed in reduced format and folded as a brochure of size A5. This results in 2 sheets of paper as illustrated in Figure 6–1.

Figure 6–1 Saddle stiched printing



To rearrange the order of the pages in a PostScript file the command `DOCTOR/PS /SADDLE` must be used.

Note:

It is important to note that the two qualifiers `/PS/SADDLE` *must* be specified together and in that order.

The `/SADDLE` qualifier cannot be used in combination with any of the other PostScript manipulation qualifiers (like `/EXTRACT, /CHANGE` etc.).

Finally, `/SADDLE` only works for printing 4 pages on a sheet (two on each side). There is no support for 4 or more pages printed on a single side of a sheet of paper.

Digital Internal Use Only

You must indicate how many pages will compose a signature. Obviously this must always be a multiple of 4. It doesn't matter if your PostScript document page number is not an exact multiple of this signature size: DOCTOR will append the necessary final pages to complete the final signature. If the document only has 7 pages, DOCTOR will create a blank 8th page to team up with page 1 on the same sheet.
The number of pages to gather for each signature is specified with `/GATHER=`. If you do not specify this qualifier, the entire book is considered to become a single signature. This will be convenient for typical books of less than 32 pages (8 sheets). If the book is thicker, you should specify a gather value yourself to allow for saddle stiching the individual signatures later. Alternatively, if `/GATHER` is not specified for a thick book, the entire book will be a single signature that can be used if the pages are cut in the middle to produce two stacks of the half-sized sheets that are then glued together in a perfect bind (or by thermoglue or spiral binding).

Additional qualifiers that can be specified are:

- `/LOG` for additional information during processing
- `/OUTPUT=` to specify an output file specification. By default the next higher version of the input file is created (and a warning is issued that purging will delete the original).
- `/2UP`. This will cause DOCTOR to process the file produced with `/SADDLE` once more and modify some page output PostScript code to allow you to print the resulting file on A3 sheets with two A4 pages printed on a side (rather than reducing each page to fit two pages on an A4 sheet). Because the code inserted is very specific PostScript for VAX DOCUMENT V2.1 output, this qualifier is ignored if specified for any other PostScript document and a warning message is issued.

Although it is possible, it usually adds to the confusion and frustration to try and reorganize a PostScript file that was reorganized before. It also shows common sense to first manipulate the PostScript file in all other ways (like adding blank pages, modifying prolog etc) before trying to re-order the pages. The re-ordering should be the last thing you do on a PostScript file before sending it to a printer.

Note:

Because several products that produce PostScript code redefine the definition of the `showpage` command, which is used by DOCTOR to insert additional blank pages, the resulting output file may print as a set of blank pages. Currently the following PostScript output has shown to print correctly (or DOCTOR has special features built in to produce the desired result): VAX DOCUMENT V1.2 or higher, DECwrite V2.0 and higher, DECpresent V1.0, All MS-Windows V3 PostScript files produced through the MS-Windows `"Windows PSCRIPT"` converter (e.g. MS-Powerpoint, MS-Word).

Digital Internal Use Only

6.6.1 Using an LPS40 to print the file

The LPS40 is only capable of single sided printing. This means that you will get two pages on a sheet and a blank back side. Those pages can serve as master copy before using them in copying machines for further duplication.

The command to give is

```
PRINT/QUEUE=lps40_queue file.PS /PARAMETER=(DATA=POSTSCRIPT, NUMBER=2)
```

By default all pages are bordered. If you don't want this, you need to make sure that the file LPS$NOBORDERNUP.LUP exists in a system directory that is pointed to by logical name LPS$LAYUP. In this directory the Printserver software will look for any layup information to position the output on the printed pages.

The contents of the file LPS$LAYUP:LPS$NOBORDERNUP.LUP should contain the following lines:

```
! LPS$NOBORDERNUP.LUP
! specifies a variation for n-up printing.  A larger left margin
! is specified to allow for hole punching.  This file is for single sided
! printing. It does not draw borders around the pages

no borders
margins = 19, 19, 60, 19
```

It is then activated through the print command:

```
 PRINT/QUEUE=LPS40_queue file.PS -
       /PARAMETER=(DATA=POSTSCRIPT, NUMBER=2, LAYUP=LPS$NOBORDERNUP)
```

6.6.2 Using an LPS20 to print the file

If single side printing is required, the same operations are applicable as for the LPS40 printing as outlined in the previous section.

If double sided printing is required to produce a ready to use booklet or double sided master for further reproduction, the following command is required:

```
PRINT/QUEUE=LPS20_queue  file.PS -
   /PARAMETER=(DATA=POSTSCRIPT, NUMBER=2, SIDE=TUMBLE)
```

The TUMBLE keyword ensures that all pages are printed in the right orientation and not upside down, which is the default for double sided printing (to allow easy flipping through, but that's unwanted now).

When no borders are wanted around the pages, the same LAYUP keyword can be specified as described for the LPS40.

Digital Internal Use Only

6.7 Minimal conformant files

Adobe Inc. defined a "Document Structuring Conventions Specification" for PostScript files that utilities such as VAX DOCUMENT and DECwrite should adhere to. Only then other utilities such as DOCTOR can inspect those files and manipulate them. DOCTOR's output is also conformant to these styles, so one could use DOCTOR on its own files recursively.

A complete description of the conformant rules can be found in the *PostScript Reference Manual, 2nd edition* written by Adobe Inc. and published by Addison & Wesley.

Electronic copy:

An electronic copy of this part of the manual can be obtained via the mail file server of Adobe System Inc. This is a three step approach, where in each case a simple MAIL message is sent (no headers etc - just a plain message the way VAXmail works) to `DECWRL::"adobe!ps-file-server"` with no subject title[1].

0 Initially you may send a simple mail with only the word "help". In response, the mail server will send you instructions on what are valid requests to it.

1 The first request consists of only the word "index". This results in an index of categories of documents that are available.

2 The second request is asking for the list of files available within such a category. For example: "index Documents". This produces a list of files available from the category (folder) Documents.

3 The final request, based on the previous list of information, is asking for the files to be sent over. Sometimes a single file is so large, that mail gateways will refuse it. So you can ask it in parts, and will need to edit them together afterwards (removing all heading and trailing information added by the mail gateways). Each part must be a separate mail request (otherwise the combined request still exceeds the maximum byte size of the allowed mail size).
The request is made by sending the words `send` *`category filename`*.
For example, for the structuring conventions:

```
send Documents DSC.ps.A
```

and send additional messages for the parts `.B`, `.C`, `.D`, `.E` and `.F` (as was the case in September 1992).

It is also known informally as the *Red Book* because of its cover colour. Appendix C of that book describes the style. There are two important aspects to conformancy:

1 The comment blocks have a particular format

[1] Thanks to Kevin Manderson to point this out to me

Digital Internal Use Only

2 The prolog contains all definitions, font setups etc, so that each page description depends only on itself and the definitions in the prolog.

Currently there are two versions of structured comments defined by Adobe, known as PS-Adobe-V2 and PS-Adobe-V3. Although both have much in common, there are certain incompatibilities. DOCTOR attempts to interpret both correctly.

In a conformant file, the descriptions of the individual pages are self-contained. They do not depend on what's written on other pages. For this reason, any definition of customized PostScript commands are not embedded inside the description of a single page, but they are all collected at the front of the document, the prolog part. Hence VAX DOCUMENT PostScript files allows printing of some pages, as long as the extracted file contains the prolog, the selected page descriptions, and the trailer. The prolog can also be enhanced (add certain features applicable to all pages), as long as no definition already present is removed.

For DOCTOR, the only important aspects are that each PostScript file contains at least the parts shown in Example 6–1. It needs to be able to find the end and start of the header, prolog, main body and trailer section of the document. Within the main body, it must find all the page headers in case an extract is required. Each conformant file needs to start with `%!PS-Adobe` on the first line of the file. DOCTOR will check on this and aborts if it did not find that line. If the line is found, it assumes the remainder of the file also conforms to the coding standards.

You can always do a

```
$ SEARCH  PostScript_file %%Page:
```

to see what page folio numbers are available within the PostScript file if you want to use the `/EXTRACT` qualifier option. When these page numbers are surrounded by parentheses (as done with Adobe V3 comment standards), you do not specify these parentheses.

It is important to realize that many PostScript documents consist of contributions made by different packages: a type setting system such as VAX DOCUMENT and graphical packages such as MacDraw or RAGS or single page documents from DECwrite or DECpresent. Each of these create their own PostScript files that are somehow combined into the final complete document file. A scan for all the `%%Page:` entries will then quickly reveal deviations from the ordinary page numbering. An example list of this from one of the VAX DOCUMENT documentation files on using graphics reveals this:

Digital Internal Use Only

Example 6–1 Conformant PostScript skeleton

```
%!PS-Adobe
%%
%%CreationDate:  date
%%
%%  ...other header comments...
%%
%%EndComments   (or a line not starting with %% or %!)

%%
%%  ...prolog commands...
%%
%%EndProlog

%%Page: folionr  ordinal

   ...main body of text...

%%Page: folionr  ordinal

   ...etc for all pages...

%%BeginDocument: (filespec)
   ...encapsulated PostScript code of included figure...
%%EndDocument

%%Trailer
```

```
%%Page: (2-12) 33
%%Page: (2-13) 34
%%Page: (2-14) 35
%%Page: 1 1                  <--  included figure
%%Page: (2-15) 36
%%Page: "Only" 1             <--  included figure
%%Page: (2-16) 37
%%Page: (2-17) 38
%%Page: (2-18) 39
```

The DOCTOR recognizes this as a sudden disruption of the incremental change of the ordinal page number that indicates the number of the page sheet within the document. If such a distruption occurs, DOCTOR does not consider the disrupting `%%Page:` entry a real page and treats it as an ordinary line.

Digital Internal Use Only

7 DOCTORing SDML files: hierarchy of files within a book

7.1 Overview

A complex document written for VAX DOCUMENT processing can consist of many elements. The text may all be written in a single source file, or distributed over many others, that are all <INCLUDE>d into the final printable document. Besides text, other tools and utilities can have provided additional graphical or tabular data. Often a complex document is built using a *profile file* where this profile specifies the names of the other SDML files that are part of the complete document.

When several people work on such a compound document, it is easy to loose track of the number of files that are referenced from the main, the root, source file.

Here, the `DOCTOR/SDML` utility, also known as MARFIN (MARkup Files INcluded), can assist you in the process. Given any top level root file that is coded in VAX DOCUMENT, it will produce one or several of:

- An organizational hierachy of how the document is composed of individual elements (that in turn may also be composed of smaller elements).
- A DEC/MMS description file that lists all the dependencies between the individual components in order to rebuild the final printable document in any of the supported destinations: LN03, PostScript, Bookreader, Line_printer or Mail.
- A list of all occurrences of the <X> and <Y> tags that are written in those VAX DOCUMENT source files, annotated with the exact line number of the file in which they were found.

The following sections show you how to use that information.

7.2 Listing all included files

When a document is processed to the final output, the VAX DOCUMENT command line indicates the top level source SDML file to be processed. When one of VAX DOCUMENT's three components (Tag Translator, Text Processor and Device Converter) encounter any of the tags (or their equivalents) as specified in Table 7–1, the specified files will be opened and included into the final printable output.

The tags available in VAX DOCUMENT to include material into the document are listed in Table 7–1.

Digital Internal Use Only

Table 7–1 Tags that cause other files to be included

tag	description
<ELEMENT>(filespec)[1]	An SDML file specified in a profile file
<INCLUDE>(filespec)[1]	An SDML file specified inside another SDML file.
<INCLUDES_FILE>(logical-name\file-spec)[1]	A reference in the profile that one of the <ELEMENT> files contains a <INCLUDE> with a logical name that must be translated into the specified file spec.
<TABLE_FILE>(file-spec)[1]	Within a table, inserts a table.
<FIGURE_FILE>(target-device\file-or-space\vertical-size)[3]	Specified within a <FIGURE> environment, and specifies a binary graphical file to be included in the document.
<INCLUDE_TEX_FILE>(file-spec)[2]	Specifies that the Tag Translator should insert the specified file with TEX macros at this spot. It is a `.TEX` file, not an `.SDML` file.
<ICON_FILE>(target-device\file-spec\vertical-size)[3]	As a figure file, but within an <ICON> context.
<EXAMPLE_FILE>(file-spec)[1]	Allows a text file to be inserted as part of an <EXAMPLE>.
<MEMO_FILE>(dvi-file-spec)[3]	Allows another `.DVI_`*dest* file to be included by the Device Converter at this position in the output file.

[1]Tag Translator does inclusion

[2]Text Processor does inclusion

[3]Device Converter does inclusion

By specifying the `DOCTOR/SDML` for an `.SDML` file, the DOCTOR will open that file and follow every lead if one of the above tags is encountered. Only when those files are VAX DOCUMENT source files themselves, the DOCTOR opens those files and recursively inspects those too. It will ignore any of these included file tags if they are coded inside a <LITERAL> and/or <COMMENT> block. If you do want to take into account these blocks, you must specify the `/IGNORE=(COMMENTS,LITERALS)` qualifier.

The final result is a new version of the specified file, but with a comment header that illustrates the nested composition of the parts the document is build of. Rather than the next higher version, you can specify the output file yourself through `/OUTPUT`, or if you don't want another version of the source file, `/OUTPUT=NL:` produces no output. This is useful only when some other output is required, e.g. the MMS description file.

Digital Internal Use Only

It is also highlighted which files were supposed to be there, but could not be found. By specifying DOCTOR/SDML/CMS you instruct the DOCTOR also to inspect any DEC/CMS libraries that are currently defined through the CMS SET LIBRARY command. Normally it will simply look in the CMS library and inspect any of the highest generations of the CMS elements. However, by specifying /CMS=*generation* only elements are looked up that are a member of that particular generation.

An example output is given in Example 7–1.

This example illustrates that the header that is produced always starts with *MF*. You should not remove this, if you wish to run DOCTOR/SDML over the same source again later. These *MF* are recognized by the utility as being an older hierarchy list and it would replace it by an updated one, if the same SDML file is scanned a second time.

The annotations in the example indicate

❶ The version of the utility that produced the listing. This is important if bug fixes are communicated to the developers.

❷ The utility was invoked as DOCTOR/SDML/CMS to allow searching though the active DEC/CMS libraries that are listed here.

❸ This file is not found in either the current directory or any other CMS library

❹ This file was found only as element in the DEC/CMS library

❺ The remainder of the file is unmodified and an exact copy of the specified input file. In the example, we used a *profile* file that includes many *elements* that in turn reference several graphical figures (.PS, .EPS, .SIX) as well as program examples in Pascal (.PAS).

Using the DOCTOR/SDML utility provides you with an easy tool to make sure all elements that are needed to rebuild your document are present (or not...).

7.3 Producing MMS rebuild files

Because once the hierarchy list is made, the DOCTOR also knows which files depend on which other ones, it is an easy task to produce a description file that can be read by DEC/MMS to rebuild the document if one of the elements is modified.

Because DEC/MMS does not know how to build an LN03 or PostScript file from a SDML file, the DOCTOR/SDML also inserts all the required MMS rules and suffixes to allow DEC/MMS to retrieve the sources from a CMS library and to rebuild it using a proper VAX DOCUMENT command.

Digital Internal Use Only

Example 7–1 Hierarchy list produced by DOCTOR/SDML

```
<COMMENT>*MF*
*MF*    Created by DOCTOR V3.0 at 20-JUL-1992 13:51:10.98  ❶
*MF*
*MF*    Command issued: DOCTOR/SDML PASCAL_DECPRESS.SDML/CMS
*MF*    QAR's to:        Theo de Klerk @UTO,  NSIC00::KLERK
*MF*
*MF*         *** Digital Internal Use Only ***
*MF*
*MF*    CMS library = PSQ:[EXAMPLE.CMSLIB]
*MF*
*MF*    The following files are referenced by this SDML file:
*MF*
*MF*     FRONT.SDML
*MF*     VMS.SDML
*MF*          PROCESS.PS
*MF*          PSPACE.PS
*MF*          PSPACE.SIX --- file not found ---  ❷
*MF*          SCHEDULE.PS
*MF*          SCHEDULE.SIX --- file not found ---
*MF*     IMAGEACTIVATOR.SDML
*MF*          SOURCEFILE.PS
*MF*     PASCAL_ENHANCE.SDML
*MF*          VARYING.PAS  (CMS library)          ❸
*MF*          STRING_SCHEMA.PAS  (CMS library)
*MF*          CONFORMANT.PAS
*MF*     ENVIRONMENT.SDML  (CMS library)
*MF*     STARLET.SDML
*MF*          DESCFIG1.EPS
*MF*     STORAGE.SDML
*MF*          VAL_VAR_CAL.PS
*MF*          NOPICTURE.TXT
*MF*          FOREIGN.PS
*MF*     CASE.SDML  (CMS library)
*MF*     BIBLIOGRAPHY.SDML  (CMS library)
*MF*     GLOSSARY.SDML
*MF*<ENDCOMMENT>

<PROFILE>
    <ELEMENT>(FRONT.SDML)                          ❹
    <ELEMENT>(VMS.SDML)
    <ELEMENT>(IMAGEACTIVATOR.SDML)
    <ELEMENT>(PASCAL_ENHANCE.SDML)
    <ELEMENT>(ENVIRONMENT.SDML)
    <ELEMENT>(STARLET.SDML)
    <ELEMENT>(STORAGE.SDML)
    <ELEMENT>(CASE.SDML)
    <ELEMENT>(BIBLIOGRAPHY.SDML)
    <ELEMENT>(GLOSSARY.SDML)
    <INDEX_FILE>
<ENDPROFILE>
```

In order to produce such a description file for DEC/MMS you must specify the /MMS qualifier. If you specify a filespec to it, that filename will be used. Otherwise a description file is produced with the same filename as the input SDML file, but with file extention .MMS. If you don't want to replace the header of the original source file also, specify /OUTPUT=NL: to ensure only an .MMS file is produced, and no new .SDML file.

The DOCTOR/SDML will generate an MMS description file that has build instructions for the following destinations. If nothing is specified explicitly by the user, a default choice is made for the name of the doctype and the destination:

- BOOKREADER (default: SOFTWARE.ONLINE BOOKREADER)
- LN03 (default: REPORT LN03)
- POSTSCRIPT (default: REPORT POSTSCRIPT)
- LINE_PRINTER (default: REPORT LINE)
- MAIL (default: REPORT MAIL)

It is possible to specify a different doctype name or destination, if your local site does not use the above (default DOCUMENT installation) names:

- Use the /DOCTYPE to specify another doctype for either paper or online destinations:

```
/DOCTYPE=(PAPER=name, ONLINE=name)
```

- Use the /DESTINATION to specify another destination name for one or several of the possible output destinations:

```
/DESTINATION=(LN03=LN03_destination,
              POSTSCRIPT=PostScript_destination,
              LINE_PRINTER=LINE_PRINTER_destination,
              MAIL=MAIL_destination,
              BOOKREADER=BOOKREADER_destination
             )
```

It is possible to order DOCTOR to only inspect element generations that belong to a specific CMS class. In that case, the /CMS=*generation* must be specified. This also results in an additional MMS macro, CMSFLAGS=/GEN=*generation* to be specified in the MMS description file.

Finally, some sites use special logical names, additional DEC/MMS rules etc. To allow for this, you can specify the /INCLUDE= qualifier. It specifies a file that contains valid DEC/MMS instructions. Its result on the produced description file by the DOCTOR/SDML /MMS is that an .INCLUDE filespec directive is written into the description file. This included file can contain your own set of suffix definitions, action rules and MMS macro symbols.

Digital Internal Use Only

Normally DOCTOR/SDML/MMS provides a set of suffixes and action rules to retrieve files from DEC/CMS libraries. If you use an .INCLUDE file, you may want to define all your own suffixes and rules and not use those provided by DOCTOR. In that case, you should specify the /MMS/NORULES. This qualifier will still produce a minimum set of rules that enable MMS to rebuild the final target files from the .SDML source files.

The valid qualifiers for DOCTOR/SDML are listed in Table 7–2.

Table 7–2 DOCTOR/SDML qualifiers for DEC/MMS

/MMS file	Produces a new version of SDML and a standard MMS description file.
/MMS=descfile file	As above, but with descfile used for the MMS output file
/MMS=descfile/NORULES file	As above, but without the generation of default MMS rules and suffixes
/MMS /DOCTYPE=(PAPER=MANUAL) /DESTINATION=(POST=MY_QUEUE)	produces an MMS file for making a MANUAL style based document for paper destinations, and using the MY_QUEUE name for PostScript destinations.
/MMS /CMS=generation	produces an MMS description file that contains only the relations to rebuild the document consisting of elements that are part of the specified CMS generation (class).

Because the doctype and destination as well as the VAX DOCUMENT commmand are defined as MMS symbols, you can at any time overrule them by specifying the same symbols as DCL command symbols and invoke the description file through the command MMS/OVERRULE.

Specific additional DOCUMENT qualifiers may be specified using /DOCQUALIFIERS= where the string of DOCUMENT qualifiers are specified between quotes, e.g. /DOCQUALIFIERS= ("/CONTENTS/INDEX"). By default, /CONTENTS /INDEX /BATCH=(NOPRINT,NOTIFY) is used.

An example output is given in Example 7–2.

Example 7–2 Example DEC/MMS description file

Example 7–2 (continued on next page)

Digital Internal Use Only

Example 7–2 (Cont.) Example DEC/MMS description file

```
!
!  Created by DOCTOR V3.1 at 24-SEP-1992 13:53:23.58      ❶
!
!  Command issued: /SDML PASCAL_DECPRESS.SDML/OUT=NL: /MMS=X.MMS
!  QAR's to:       Theo de Klerk @UTO,  NSIC00::KLERK
!
!      *** Digital Internal Use Only ***
!
.SUFFIXES    ❷
.SUFFIXES  .DECW$BOOK .PS .LN03 .LINE .TXT  .FDL .CLD .MSG .FOR .PAS  -
   .PSART .EPS .FSE .SIX .SDML .GRA .DECW$BOOK~ .PS~ .LN03~ .LINE~ .TXT~  -
   .FDL~ .CLD~ .MSG~ .FOR~ .PAS~ .PSART~ .EPS~ .FSE~ .SIX~  -
   .SDML~ .GRA~

DOCUMENT=DOCUMENT     ❸
DOCTYPE=REPORT
ONLINE_DOCTYPE=SOFTWARE.ONLINE
LINE_DEST=LINE
MAIL_DEST=MAIL
POSTSCRIPT_DEST=POSTSCRIPT
LN03_DEST=LN03
ONLINE_DEST=BOOKREADER
DOCQUALIFIERS=/CONTENTS/INDEX/BATCH=(NOTIFY,NOPRINT)

.DECW$BOOK~.DECW$BOOK : ❹
    IF "$(MMS$CMS_LIBRARY)" .NES. "" THEN  DEFINE/USER CMS$LIB $(MMS$CMS_LIBRARY)
    $(CMS) FETCH $(MMS$CMS_ELEMENT) /OUTPUT=$(MMS$TARGET_NAME).DECW$BOOK $(CMSFLAGS) $(CMSCOMMENT)

.PS~.PS :
    IF "$(MMS$CMS_LIBRARY)" .NES. "" THEN  DEFINE/USER CMS$LIB $(MMS$CMS_LIBRARY)
    $(CMS) FETCH $(MMS$CMS_ELEMENT) /OUTPUT=$(MMS$TARGET_NAME).PS $(CMSFLAGS) $(CMSCOMMENT)

    ... etcetera ...

.SIX~.SIX :
    IF "$(MMS$CMS_LIBRARY)" .NES. "" THEN  DEFINE/USER CMS$LIB $(MMS$CMS_LIBRARY)
    $(CMS) FETCH $(MMS$CMS_ELEMENT) /OUTPUT=$(MMS$TARGET_NAME).SIX $(CMSFLAGS) $(CMSCOMMENT)

    ... etcetera ...

.SDML~.SDML :
    IF "$(MMS$CMS_LIBRARY)" .NES. "" THEN  DEFINE/USER CMS$LIB $(MMS$CMS_LIBRARY)
    $(CMS) FETCH $(MMS$CMS_ELEMENT) /OUTPUT=$(MMS$TARGET_NAME).SDML $(CMSFLAGS) $(CMSCOMMENT)

.SDML.DECW$BOOK :
    $(DOCUMENT) $(MMS$TARGET_NAME) $(ONLINE_DOCTYPE) $(ONLINE_DEST) $(DOCQUALIFIERS)

.SDML.PS :
    $(DOCUMENT) $(MMS$TARGET_NAME) $(DOCTYPE) $(POSTSCRIPT_DEST) $(DOCQUALIFIERS)

.SDML.LN03 :
    $(DOCUMENT) $(MMS$TARGET_NAME) $(DOCTYPE) $(LN03_DEST)       $(DOCQUALIFIERS)

.SDML.LINE :
    $(DOCUMENT) $(MMS$TARGET_NAME) $(DOCTYPE) $(LINE_DEST)       $(DOCQUALIFIERS)
```

Example 7–2 (continued on next page)

Example 7–2 (Cont.) Example DEC/MMS description file

```
.SDML.TXT :
    $(DOCUMENT) $(MMS$TARGET_NAME) $(DOCTYPE) $(MAIL_DEST)       $(DOCQUALIFIERS)

.GRA.SDML :
    $(DOCUMENT)/GRAPHICS=RENDER $(MMS$TARGET_NAME).GRA -
         /TYPE=(PS,BRF,MONOSIX)/OUTPUT=$(MMS$TARGET_NAME)/NOBACKGROUND/SDML=FIGURE

.DEFAULT :  ❺
    ! No action for $(MMS$TARGET)

PROFILE.DECW$BOOK DEPENDS_ON  PROFILE.SDML, -
        BOOKREADER_FIX.SDML, -
        FRONT.SDML, -
        PART10.SDML, -
        VMS.SDML, -
        PROCESS.FSE, -
        PSPACE.FSE, -

    ... etcetera...

PROFILE.LN03 DEPENDS_ON  PROFILE.SDML, - ❻
        BOOKREADER_FIX.SDML, -
        FRONT.SDML, -
        PART10.SDML, -
        VMS.SDML, -
        PROCESS.SIX, -
        PSPACE.SIX, -

    ... etcetera...

PROFILE.PS DEPENDS_ON  PROFILE.SDML, -
        BOOKREADER_FIX.SDML, -
        FRONT.SDML, -
        PART10.SDML, -
        VMS.SDML, -
        PROCESS.PS, -
        PSPACE.PS, -

   ...etcetera for all other targets...

! If the next dependency is used, one should not submit the final
! DOCUMENT jobs in batch mode as they are not allowed to run in
! parallel. The MMS macro DOCQUALIFIERS should be defined as
! DOCQUALIFIERS=/CONTENTS/INDEX
ALL_DOCUMENTS DEPENDS_ON PROFILE.DECW$BOOK, -     ❼
         PROFILE.PS, PROFILE.LN03, -
         PROFILE.LINE, PROFILE.TXT
      ! All are up to date
```

Some notes by the MMS description file as shown in this example:

❶ The version number of the utility is once again mentioned.

❷ The order in which file types depend on one another is redefined. It includes all the file types the DOCTOR encountered while building the hierarchy list. This list will be suppressed if `/NORULES` is specified.

❸ The DOCUMENT command as well as the doctype, destination and qualifiers are all defined as symbols to allow overrule through DCL symbols or a `MMS/MACRO=` command.

❹ For each of the file types found, rules are made to allow MMS to fetch them from a DEC/CMS library. These are suppressed if the qualifier `/NORULES` is specified.

❺ When no update can be made for a certain file element, by default it notifies the user and continues.

❻ The first target to build is the entire document. Dependency lines for all supported output file types are created in a similar way.

❼ If you want all destinations to be processed one after the other, you must invoke MMS in this example, using the command

```
MMS/DESCRIPTION=PROFILE.MMS ALL_DOCUMENTS
```

It will then execute this annotated line which in turn will invoke each of the DOCUMENT commands to produce the Bookreader, PostScript, LN03, Line_ printer and Mail files respectively. This command may not result in a batch job, however. That would cause parallel execution of DOCUMENT on the same set of files, causing a conflict on the access to the `.XREF` cross reference file of the book: it must be executed in sequence.

7.4 Using the MMS description file

7.4.1 What file to build?

The generated MMS description file can be used in several ways. As you can see from Example 7–2, a dependency rule is generated for each of the supported destinations.

If you run MMS using all its defaults, it will always generate a dependency line for the Bookreader destination only.

As can be seen from Example 7–2 this is the first dependency rule specified in the generated description file, and by default this is the only action performed by MMS.

If you want any specific destination to be used, you must invoke MMS as

```
$ MMS/DESCRIPTION=file.MMS  targetfile
```

where the *targetfile* is the output file you want to build. From the Example 7–2 this could be either `PROFILE.PS` or `PROFILE.LN03` or any of the others.

Digital Internal Use Only

7.4.2 Another doctype than specified

Once the MMS description file is generated, a value is assigned to the MMS macro DOCTYPE and ONLINE_DOCTYPE. Either these are the defaults REPORT and SOFTWARE.ONLINE or those that were specified using the qualifier /DOCTYPE=(PAPER= ,ONLINE=)

If for some reason you still want another doctype, there are three ways of obtaining this:

1 Edit the MMS description file and change the values specified for DOCTYPE and ONLINE_DOCTYPE macros as specified at the start of the file.

2 Invoke MMS using the /MACRO qualifier:

```
MMS/MACRO=("DOCTYPE=other_name","ONLINE_DOCTYPE=other_name")
```

3 Define a DCL symbol with the same name as the MMS macro before you invoke MMS (not using the /MACRO qualifier):

```
$  DOCTYPE = "other_name"
$  ONLINE_DOCTYPE = "other_name"
```

7.4.3 Another destination than specified

If another destination name is required than that specified in the MMS description file, you can proceed the same way as specified in Section 7.4.2, replacing the macros:

```
ONLINE_DEST
POSTSCRIPT_DEST
LN03_DEST
LINE_DEST
MAIL_DEST
```

Change of the destination is normally only required if the local DOCUMENT system has been set up to know different names than those specified by default. This is especially the case with sites that have DOCUMENT installed from V1.0 onwards or that have enabled automatic printing after processing. Those started at V2.0 normally don't have different names and will not have the need to change the destination names.

7.4.4 Using DOCUMENT/GRAPHICS (Rags)

A special mention must be made for the use of DOCUMENT/GRAPHICS produced graphics. This utility, that is delivered with VAX DOCUMENT V2.0 onwards, is a DECwindows oriented graphics editor, that will produce one or several different output graphics: PostScript, sixel or BRF-bookreader files. In addition it can output another .SDML file that only contains <FIGURE_FILE> tags. This enables the author to completely decouple

text and graphics components of the document. The only thing written in the text file will be

```
<FIGURE>(figure caption\figure_symbol)
 <INCLUDE>(figurefile.SDML)
 <ENDFIGURE>
```

whereas the figurefile.SDML will be created by the DOCUMENT/GRAPHICS editor and contains the required <FIGURE_FILE> tags with the correct size of the figure, which may differ for each destination.

When DOCTOR/SDML/MMS is required to produce an MMS file, it will also add the action rule to invoke the graphics editor DOCUMENT/GRAPHICS=RENDER to recreate any such .SDML file from a graphics meta-file with file type .GRA. However, this will only work, if the MMS description file is invoked by MMS while running under a DECwindows environment, as the DOCUMENT/GRAPHICS utility will only work when it has the DECwindows environment available—even if it will not display a single window. For this reason, the MMS file must either be invoked from a DECterm window on your workstation or, if submitted to batch made, the LOGIN.COM file needs to contain a DCL command to create a display on some active workstation:

```
$ SET DISPLAY/CREATE/NODE=nodename /TRANSPORT=(LAT|LOCAL|DECNET)
```

where the specified node name is a workstation in which the Session Manager has been told (via the Security option) to accept displays from the node on which the batch queue is running and from the username for which the batch job has been submitted. If the security setting does not allow this, the MMS job will abort because DOCUMENT /GRAPHICS will return a fatal error for "unable to create display".

Instead of the LOGIN.COM file, you can also specify another MMS file to contain the SET DISPLAY command. This MMS file is added to the generated one through the DOCTOR/SDML/MMS/INCLUDE qualifier.

7.5 Retrieve all index entries

When a document is written and it contains many index entries, your chances are good that some index entries that were supposed to be identical, but on different pages, are actually printed as two entries in the index. Usually because you mistyped an entry of the <X> or <Y> tag.

Finding those typo's is tedious: you need to get the book printed, then look in the index, find the errors, look at that page, determine which file is comes from and finally go into the editor and find the entry.

The DOCTOR allows all this to be done much easier, without the overhead of rebuilding the entire book each time you think you found all the errors. (Usually the fixing of index entries take several repeated loops of correcting the tag entries).

By specifying DOCTOR/SDML/INDEX the DOCTOR produces the hierarchy list, but, while scanning through all the SDML source files, it also copies each and every occurrence of the <X> and <Y> tags into a separate file, specified by the /INDEX= qualifier.

The result looks something like Example 7–3. This may look odd, but it is very useful.

Example 7–3 Output of DOCTOR/SDML/INDEX

```
<X>(DOCTOR<XS>installation)       <comment>(    34 DOCTOR.SDML)
<X>(DOCTOR<xs>invocation)       <comment>(    58 DOCTOR.SDML)
<X>(CONFORM)       <comment>(    16 DOCTOR_MSG.SDML)
<X>(CREATED)       <comment>(    25 DOCTOR_MSG.SDML)
<Y>(IDENT - See DOCTOR)       <comment>(    34 DOCTOR_MSG.SDML)
```

The fact that all lines start with tag <X> or <Y>, allows you to use the DCL command SORT to sort the file into alphabetical order. (It is produced in order of the files referenced by the DOCTOR).

```
$ SORT  input_file   output_file
```

This will position typo's near each other and easy to spot when the sorted file is printed. It is also easy to load all the required SDML source files into your editor buffers once, and then move from buffer to buffer[1] to correct the errors. Since you do not add any lines, but simply re-type an entry, the line numbers that preceed the file specification that is given in the <COMMENT> on the same line, allows you to use a *goto line* command[2] to quickly move to the indicated line to find the incorrect tag entry.

Once all these entries have been corrected in the SDML files, you can run DOCTOR /SDML/INDEX once more on it and inspect the new output to spot any typo's you forgot. If there are none, you can use the produced index file to make a dummy book by processing the index file by itself. Of course the final output will contain non-sense page numbers, but at least it is processed and build in substantially less time than redoing the entire document. When the processed and printed index also looks correct, then you reprocess the entire book. If you're still not satisfied, you should make additional changes to the incorrect tag entries.

Note:

You may run into %TEX-MEMEXCEEDED errors when there are many entries in the produced index file. Because there is no real text, the Text Processor can write all the index entries on a single page. It stores them in memory, waiting for a new page to start (which never comes). Therefore, it is a good idea to use an editor and insert dummy text between the lines of index tags, as well as <PAGE> tags. If you supply the qualifier /RANDOM_TEXT, then DOCTOR will add this text for you. Obviously, you need only to print or inspect the resulting *_INDEX.* files.

[1] "GOTO BUFFER name" for LSEDIT

[2] LINE for LSEDIT

Digital Internal Use Only

8 DOCTORing SDML files: Counting tags used

8.1 Overview

This is a very simple utility that counts all the tags encountered in an SDML file. A tag is defined as any contiguous text string consisting of alphabetic, numeric or underscore characters, surrounded by the opening and closing angle brackets: <LIKE_THIS>. For this reason, the utility also detects and counts all user defined tags.

The utility is invoked through DOCTOR/TAG_COUNT, and takes a single SDML file as input parameter. It automatically also searches through all files referenced by the input file specified: if a profile is given, all elements of the book are also searched, as are the <INCLUDE> files.

Note:

This utility counts any tag-like construct, as it does not interpret or validate them. This also means it counts all tags within <COMMENT> blocks and/or <LITERAL> blocks. When within these blocks <INCLUDE> or <ELEMENT> or <INCLUDES_FILE> tags are encountered, DOCTOR/TAG will attempt to lookup the files referenced by these tags. It will give a warning if those files do not exist, but continues processing.

8.2 Options

Because of its simplicity, there are few options with this utility. You can specify your own output file specification using the /OUTPUT qualifier. If omitted, it defaults to the input file specification, but with file type .TAG_COUNT. Only /LOG can be specified to keep track of the number of files sofar processed in case several nested SDML included files are encountered.

8.3 Output format

The output file is a simple ASCII text file, formatted as indicated in Example 8–1, which was made against a version of the source file of this document.

Notice that the number of occurrences is given in the first columns 1–7, whereas the tag names are given from column 9 onwards. They are sorted in alphabetical order, but if you need them in numerical order, the DCL SORT command can quickly do this for you:

```
SORT /KEY=(POSITION=1,SIZE=7) infile outfile
```


Digital Internal Use Only

Example 8–1 Sample DOCTOR/TAG_COUNT output file

```
 Generated by DOCTOR/TAG_COUNT.  Digital Internal Use Only

Tags (and their frequency) found in SDML document rooted in DOCTOR.SDML:

      2 <ABSTRACT>
      4 <APPENDIX>
     10 <CALLOUT>
      3 <CALLOUTS>
      2 <CENTER_LINE>
     10 <CHAPTER>
    412 <CODE_EXAMPLE>
      7 <COMMAND>
      1 <COMMAND_SECTION>
     23 <COMMENT>
         :
         :
    177 <TAG>
      6 <TEX>
      3 <TITLE>
      2 <TITLE_PAGE>
      4 <U>
      3 <VALID_BREAK>
    369 <X>
    104 <XS>
     13 <Y>
```

8.4 Usage

This utility was created to make a quick estimate on the amount of conversion effort required if an SDML encoded file needs to be converted into some other markup language, such as SGML, Runoff or a TEX-flavour such as LaTEX or ZzTEX.

9 DOCTORing XREF files

9.1 Overview

When a large document is built using DOCUMENT, often references are used to point to other parts of the document. After a while, and certainly if several authors work on a document, you start losing track of what symbols are defined for what tables, sections and other referable items. Even if a naming convention is used.

For this reason DOCTOR/XREF has been built. It will work on any document of which the cross reference .XREF file exists. This is always true for documents built through a profile. Single file documents do not have an .XREF file - the Tag Translator keeps all data gathered in Pass 1 in memory and uses it to resolve references during Pass 2: no need for a file. You must break up this file into parts and add a profile file to it, in order to use DOCTOR/XREF.

Using DOCTOR/XREF on a .XREF file will allow you to do several things:

- Build an SDML file with all symbols listed in a <TABLE> structure. By processing this SDML file (e.g. using the REPORT doctype) you get a nicely printed listing of all symbols and their textual contents. Those lists can be alphabetic on symbol name or sorted numerically by type (e.g. all table symbols sorted by table number). The /SORT or /LIST qualifiers are required for this.
- Build an SDML symbol file containing <DEFINE_SYMBOL> tags of all symbols defined in the .XREF file. This allows you to <INCLUDE> that file into some other book and thereby enables you to cross reference between books. The /SYMBOL_FILE qualifier is required for this.

Note:

In some cases you want to ensure all used <REFERENCE> tags will have proper definitions. Sometimes however those symbols do not (yet) exist. If you need a file where all undefined symbols can be temporarily defined using a <DEFINE_SYMBOL>, then use the DOCTOR/ONLINE/DUMMY qualifier, as described in Section 5.4.

9.2 Build symbol listings

DOCTOR/XREF allows you to produce a formatted listing of all symbols defined in a document. The listings are made in <TABLE> format and can be processed with VAX DOCUMENT using any doctype available that supports table tags. The listings can be made in alphabetical order or the symbol names but also in numerical order of the section or object numbers.

Digital Internal Use Only

To obtain these lists, you need to use the command

```
DOCTOR/XREF file.XREF /LIST=filespec
```

if you want an alphabetical list of all symbols - a mixture of tables, sections, examples, chapters, appendices and others. If you don't specify the filespec with the /LIST qualifier, it defaults to the same file name as the .XREF file, but with file type .SDML_SYMBOL_LIST.

To produce a list of symbols sorted by type of symbol (table, chapter, example, user defined and others) the qualifier /SORT is needed.

```
DOCTOR/XREF file.XREF /SORT=filespec
```

By default all symbols are then sorted alphabetically per type. However, in many cases it may be more useful if they are sorted by their numeric value (if applicable) so that you can quickly look up the symbol for Table 5-6. If this is required, the /NUMERIC qualifier is also required. To have both alphabetic and numerically sorted lists of symbols, use /ALPHABETIC/NUMERIC.

You can decide whether the listings just show the brief information about the symbol or the full text, similarly to the working of the <REFERENCE> tag, where arguments FULL, TEXT, VALUE are allowed. Use the appropriate qualifier /FULL, /TEXT, /VALUE for this.

9.3 Build cross reference symbol file

To build a file with <DEFINE_SYMBOL> tags that define the symbols plus their reference translation, you need to specify the .XREF file to scan and the /SYMBOL_FILE=*filespec* for the resulting output file.

```
DOCTOR/XREF file.XREF /SYMBOL_FILE=filespec -
            /BOOKTITLE="title text" /PREFIX=prefix
```

If you omit the file specification, the file name of the .XREF file is taken, and file type defaults to .SDML_XREF to avoid it to overrule the profile file of the book, which otherwise would have the same file specicification, but a totally different contents that you would not want to lose by an accidental purge.

If the symbols defined are going to be used in another book, a potential collision may occur if the other book uses the some identical names for symbols. To prevent this, a prefix can (and should) be attached to all symbols derived from the .XREF file to ensure their uniqueness. Within Corporate Publishing, the bookcode could be used for this. If /PREFIX is not specified, a warning is issued, but processing continues.

Note:

Make sure that symbol name + prefix never exceeds the VAX DOCUMENT limit of 31 characters maximum length. DOCTOR/XREF will warn you if this is the case and truncate the symbol name to maximum allowable length

Digital Internal Use Only

In addition to specifying a prefix, you should also specify the title of the book, using the /BOOKTITLE= qualifier. This will result in the definition of a symbol *prefix*BOOK which translates into the specified title of the book. (The *prefix* part is specified through /PREFIX or omitted if this is not specified). This way you can perform a cross reference to the book from another book by writing:

```
For more information see <reference>(prefixBOOK)
in its <reference>(prefixOVERVIEWCHAPTER)
```

Which could then translate into

... For more information see *The Art of Zen* in its Chapter 1, How It All Began ...

You can decide whether the listings just show the brief information about the symbol or the full text, similarly to the working of the <REFERENCE> tag, where arguments FULL, TEXT, VALUE are allowed. Use the appropriate qualifier /FULL, /TEXT, /VALUE for this.

In order to use the file with all the symbols defined, you must either

- Specify /SYMBOLS=file.SDML_XREF on the DOCUMENT command line for the book that references the defined symbols
- Specify a <INCLUDE>(file.SDML_XREF) in the profile of the book that references the defined symbols

9.4 Example output

This section gives three examples on the final output produced by DOCTOR/XREF when the .XREF file is scanned that is produced in processing the *Hitch Hiker's Guide to VAX DOCUMENT*, another publication of the author.

The command given was

```
DOCTOR/XREF ENG_COOKBOOK.XREF /LIST/SORT/ALPHA/NUMERIC -
  /SYMBOL/PREFIX=HHG_/BOOKTITLE="Hitch Hiker's Guide to VAX DOCUMENT" -
  /FULL
```

The results are presented below in the next subsections. Only small parts of the complete table are printed - enough to get a feeling of what DOCTOR/XREF produces.

9.4.1 Symbol file output

The output as shown in Example 9–1 has been <INCLUDE>d into this document. Therefore, we can now cross reference to the *Hitch Hiker's Guide to VAX DOCUMENT* book by using the <REFERENCE> to any particular part.

Digital Internal Use Only

Example 9–1 Sample .SDML_XREF output

```
<CHECK_FOR_INCLUSION>(ENG_COOKBOOK_XREF)
<COMMENT> ****************************************
 Created by DOCTOR V3.0
 Creation Date:    21-JUL-1992 11:37:28.78

 Command issued:   /XREF [COOKBOOK]ENG_COOKBOOK /SORT/ALPHA/NUME/FULL/LIS/SYMB/PREF=HHG_/BOOKT
ENT"
 XREF file used:   PSQ:[PRIVATE.THEO.COOKBOOK]ENG_COOKBOOK.XREF;3
 QAR's to:        Theo de Klerk @UTO,  NSIC00::KLERK

    *** Digital Internal Use Only ***

****************************************<ENDCOMMENT>

<DEFINE_SYMBOL>(HHG_BOOK\Hitch Hiker's Guide to VAX DOCUMENT)
<DEFINE_SYMBOL>(HHG_ALIGNCHAR\<_italic>(<REFERENCE>(HHG_BOOK))
\Sectionname 4.8, Aligned text)
<DEFINE_SYMBOL>(HHG_CHAP1\<_italic>(<REFERENCE>(HHG_BOOK))
\Chaptername 1, Basic commands for DOCUMENT)
<DEFINE_SYMBOL>(HHG_CHAP10\<_italic>(<REFERENCE>(HHG_BOOK))
\Chaptername  10, Using the Language Sensitive Editor LSE)
<DEFINE_SYMBOL>(HHG_CHAP3\<_italic>(<REFERENCE>(HHG_BOOK))
\Chaptername 3, Lists in Document)

  ... etcetera...

<DEFINE_SYMBOL>(HHG_THUNDERBIRDSSECTION\<_italic>(<REFERENCE>(HHG_BOOK))
\Sectionname 2.5.1, The origin of the Thunderbirds)
<DEFINE_SYMBOL>(HHG_XREFBOOKREADERSECTION\<_italic>(<REFERENCE>(HHG_BOOK))
\Sectionname 2.3, Cross references and Bookreader files)
<DEFINE_SYMBOL>(HHG_XREFSECTION\<_italic>(<REFERENCE>(HHG_BOOK))
\Sectionname 9.5.3, XREF)
<ENDCHECK_FOR_INCLUSION>
```

For example, using <REFERENCE>(hhg_chap10) would result in: "*Hitch Hiker's Guide to VAX DOCUMENT* Chapter 10, Using the Language Sensitive Editor LSE".

9.4.2 List file output

Rather than showing the VAX DOCUMENT constructs produced in the .SDML_SYMBOL_LIST file, we will show a small sample of the final output, when the <TABLE> has been processed by VAX DOCUMENT. This table is shown in Table 9–1.

Symbols in PSQ:[COOKBOOK]ENG_COOKBOOK.XREF;1

Book title: <HHG_BOOK> = Hitch Hiker's Guide to VAX DOCUMENT

All data specified in table is used for output.

Table 9–1 Alphabetic list of symbols

Symbol name	Type	Number	Text
HHG_ALIGNCHAR	Section	4.8	Aligned text
HHG_CHAP1	Chapter	1	Basic commands for DOCUMENT
HHG_CHAP10	Chapter	10	Using the Language Sensitive Editor LSE
HHG_CHAP3	Chapter	3	Lists in Document
		... etcetera...	
HHG_TB1SECTION	Section	2.5.1.1.1	Thunderbird 1
HHG_ THUNDERBIRDSSECTION	Section	2.5.1	The origin of the Thunderbirds
HHG_XREFBOOKREADERSECTION	Section	2.3	Cross references and Bookreader files
HHG_XREFSECTION	Section	9.5.3	XREF

9.4.3 Sorted list file output

When the output is sorted by its symbol type, the following tables occur. Only a small sample is reproduced here:

- Table 9–2 shows the list of chapter symbols in alphabetical order
- Table 9–3 shows the list of chapter symbols in numerical order

Similar lists are also produced for sections, figures, examples, tables, user defined symbols, and other tags that allow symbolic references.

Symbols in PSQ:[COOKBOOK]ENG_COOKBOOK.XREF;1

Book title: <HHG_BOOK> = Hitch Hiker's Guide to VAX DOCUMENT

All data specified in table is used for output.

Table 9–2 Chapter symbols - in alphabetical order

Symbol name	Number	Text
HHG_CHAP1	1	Basic commands for DOCUMENT
HHG_CHAP10	10	Using the Language Sensitive Editor LSE
HHG_CHAP3	3	Lists in Document

Digital Internal Use Only

Table 9–2 (Cont.) Chapter symbols - in alphabetical order

Symbol name	Number	Text
		...etcetera...
HHG_LAYOUTCHAPTER	2	Document layout and cross references
HHG_TABLECHAPTER	4	Tables in Document

Table 9–3 Chapter symbols - in numerical order

Number	Symbol name	Text
1	HHG_CHAP1	Basic commands for DOCUMENT
2	HHG_LAYOUTCHAPTER	Document layout and cross references
3	HHG_CHAP3	Lists in Document
		...etcetera...
9	HHG_CHAP9	Extra local additions
10	HHG_CHAP10	Using the Language Sensitive Editor LSE

Digital Internal Use Only

Command Section

Digital Internal Use Only

DOCTOR/GLOSSARY—Sort glossary items in SDML file

The DOCTOR/GLOSSARY utility will sort all <GTERM> tags and associated <GDEF> tags within an SDML source file. The glossary section may be part of a larger file. There may only be a single glossary.

FORMAT

DOCTOR/GLOSSARY *filespec [/qualifiers]*

Command Qualifiers	Defaults
/OUTPUT	*/OUTPUT=input_filespec*
/LOG	*/NOLOG*
/SORT	*/SORT=(LETTER, NONALPHA=IGNORE)*

PARAMETERS

filespec

A valid VMS file specification of an SDML file that contains a <GLOSSARY> section. No wildcards are allowed.

restrictions

- Input file must have <GLOSSARY> and <ENDGLOSSARY> if non-glossary text preceeds and follows the glossary section.
- If no <ENDGLOSSARY> tag is available, the remainder of the input file, starting at the first <GTERM> occurrence, is considered part of the glossary.
- No <INCLUDE> tags are evaluated within the glossary section: all glossary items to sort must be present in the source input file.
- Tags for comment blocks and/or literal blocks are ignored. This may lead to unexpected results if comment blocks include or span one or more <GTERM> tags. Comments on a glossary term should follow the <GTERM>, not preceed it.

prompts

File: filespec

DESCRIPTION

DOCTOR/GLOSSARY scans through the input file and will sort the file part that starts with the first <GTERM> tag occurrence, and ends with the <ENDGLOSSARY> tag. All text before this section or following it, remains untouched.

Digital Internal Use Only

The glossary terms are sorted in ASCII collating sequence, where only the character ranges "A"-"Z" and "0"-"9" are evaluated. Other characters are ignored for the sorting process. Internally all glossary terms are converted to uppercase. Output remains the same as specified in the input file.

QUALIFIERS

/LOG

This qualifier result in additional messages on the file being processed. By default no such messages are issued (`/NOLOG`).

/OUTPUT=filespec

This qualifier specifies the output file to generate from the input file. When omitted, the output file take will be the next higher version of the input file. A warning message will then be generated that the original file may be lost if a purge is done.

/SORT=keywords

This qualifier determines the way the sort is performed. One can either specify LETTER or WORD. With LETTER all terms are sorted with spaces and hyphens as not significant. This is the default. WORD does treat those as significant.

In addition one can specify the keyword NONALPHA with one of the following values:

- NONALPHA=AFTER - all terms starting with non alphabetic characters are positioned at the end of the glossary.
- NONALPHA=BEFORE - all terms starting with non alphabetic characters are positioned at the start of the glossary
- NONALPHA=IGNORE - all terms are sorted on their first alphabetic character. This is the default.

Digital Internal Use Only

DOCTOR/MESSAGE—Create SDML file from VMS Message definitions

The DOCTOR/MESSAGE utility can create a VAX DOCUMENT SDML file with explanation of error messages, from a properly coded VAX/VMS Message source file.

FORMAT

DOCTOR/MESSAGE *filespec [/qualifiers]*

Command Qualifiers	Defaults
/FAO	*/NOFAO*
/HELP	*/NOHELP*
/LOG	*/NOLOG*
/OUTPUT	*/OUTPUT=.SDML*

PARAMETERS

filespec

A valid VMS file specification of a VMS Message definition file. Default file type is .MSG. No wildcards are allowed.

restrictions

- It is assumed the input file has proper VAX/VMS Message Utility syntax and will correctly compile.
- Comments describing the message should follow the message definition line.

prompts

File: filespec

DESCRIPTION

DOCTOR/MESSAGE reads through the message definition file and produces a SDML file from the comments that describe the messages. The SDML file can be included in any SDML source file that is processed for the SOFTWARE family doctypes. It will then produce a <HEAD1> section in that document, preferably as part of an appendix. For full details see Chapter 4.

Digital Internal Use Only

QUALIFIERS

/FAO

When this qualifier is specified, the message string is written into the output `.SDML` file with all specified FAO arguments unchanged. By default all FAO arguments are replaced by generic strings, such as *string* for a `!AS` FAO argument.

/HELP

This qualifier will produce an `.SDML` file that can be processed for the `HELP.MESSAGE` doctype that is currently being developed by CUIP to produce input files for the `HELP/MESSAGE` utility for Open VMS and Alpha VMS systems. The default is `/NOHELP`, resulting in a normal `.SDML` message file for insertion into a paper or online Bookreader user's guide.

/LOG

This qualifier result in additional messages on files being processed and output files being created. By default no such messages are issued (/NOLOG).

/OUTPUT=filespec

This qualifier specifies the output file to generate from the input file. When omitted, the output file take the same values as the input file, except for the file type, which becomes `.SDML`. It is allowed to specify only a part of a filespec. Missing fields default from the input fields (except the type part).

Digital Internal Use Only

DOCTOR/ONLINE—Modify SDML files for Bookreader

The DOCTOR/ONLINE scans through the specified SDML source file and the files it references, for correct syntax to be processed for the BOOKREADER destination. This implies that symbols are added to all unsymboled sections, tables, examples and figures.

FORMAT

DOCTOR/ONLINE *filespec [/qualifiers]*

Command Qualifiers	Defaults
/CONDITION	*/NOCONDITION*
/DUMMY_SYMBOLS	*/NODUMMY_SYMBOLS*
/LIST	*/NOLIST*
/LOG	*/NOLOG*
/OUTPUT	*/OUTPUT=.SDML_ONLINE*
/REMOVE	*/NOREMOVE*
/SUPERSEDE	*/NOSUPERSEDE*
/SYMBOL_FILE	*/NOSYMBOL_FILE*
/VERSION_1	*/NOVERSION_1*

PARAMETERS

filespec

A valid VMS file specification of a VAX DOCUMENT source file. This can be a single file or the profile file of a book. No wildcard allowed.

restrictions

- It is assumed the input files have proper VAX DOCUMENT syntax and will correctly process for Mail, LN03 or PostScript output.

prompts

File: filespec

DESCRIPTION

DOCTOR/ONLINE scans through the specified source file and its referenced files to ensure that symbols are attached to all sectioning tags as well as the Bookreader "pop up" items: table, example and figure. Use the /VERSION_1 qualifier if the document is processed with VAX DOCUMENT V1.2B. A text file explaining what actions must be taken after processing can be obtained by specifying /LIST.

Digital Internal Use Only

QUALIFIERS

/CONDITION=name

This qualifier instructs DOCTOR to ignore any conditional text block whose condition does not match the one specified. This will result in ignoring any symbol defined in those blocks (and which hence will show up in the `/LIST` or `/DUMMY_SYMBOL` specified files. It does not affect generation of symbols to tags that require one. All of those will get one, regardless of any condition set or removed.

/DUMMY_SYMBOLS[=(keywords)]

This qualifier generates a file that contains <DEFINE_SYMBOL> tags for each <REFERENCE> tag that is used in the document that has no definition. This way the document can be processed successfully using `DOCUMENT /SYMBOL=symbol_file` until the references are written up by the authors or the reference itself removed.

If a symbol file for the document already exists, then the dummy symbol file generated can include a reference to this file by inserting a <INCLUDE>(symbol_filespec) tag in the generated dummy symbol file. This way you can use `DOCUMENT/SYMBOL=dummyfile` to process the document, using both the predefined symbols and the generated symbols.

There are three keywords that can be specified:

OUTPUT=*filespec*	Allows to specify the output file spec with the symbol definitions. When omitted, the output file name is identical to the input file name and file type `.SDML_DUMMY_SYMBOLS`.
TEXT="*text string*"	Allows to specify the replacement text for all undefined references. When omitted, it defaults to <TAG>(reference\undefined_symbol_name).
INCLUDE_SYMBOLS	When specified, the resulting dummy symbols file will also contain a line with <INCLUDE>(symbol_filespec). This is useful when the document is processed with `DOCUMENT /SYMBOL=dummy_filespec` because this will read both the predefined symbols as well as the generated ones.

/LIST [=filespec]

When this qualifier is specified, a list file is produced, containing instructions of what actions the author must undertake to make the document truely ready for Bookreader processing. Actions are instructions to create additional figure files, or adding sentences around created <REFERENCE> tags. By default no list is created. When no file specification is given, it defaults to the input file spec, but with the file type `.ONLINE_ERRORS`.

Digital Internal Use Only

/LOG

This qualifier result in additional messages on files being processed and output files being created. By default no such messages are issued (/NOLOG).

/OUTPUT=filespec

This qualifier specifies the output files to generate for the modified input. When omitted, the output files take the same values as the input file, except for the file type, which becomes .SDML_ONLINE. It is allowed to specify only a part of a filespec. Missing fields default from the input fields (except the type part). It is advised either to specify only the disk and directory or the file type. It is discouraged to specify a file name, as all output will then result in multiple versions of the same file name.

/REMOVE

This qualifier allows you to remove any generated symbols added by DOCTOR/ONLINE during previous processing. It executes only Pass 1 of DOCTOR/ONLINE and does not generate any new symbols. This qualifier is mutually exclusive with /SUPERSEDE.

/SUPERSEDE

This qualifier allows you to remove any DOCTOR/ONLINE generated symbols from the document and re-issue new ones. This enables you to combine document elements that were previously processed individually and therefore might have identical symbols defined in the sources which may lead to multiple symbol definition errors during DOCUMENT processing. This qualifier is mutually exclusive to /REMOVE.

/SYMBOL_FILE=filespec

Specifies that DOCTOR should also read a file that contains symbol definitions that would be used on the DOCUMENT command line through the /SYMBOL qualifier.

/VERSION_1

Specify this qualifier only for sources that must be processed with VAX DOCUMENT V1.2B. It inhibits <SUBHEAD*> tags to be given a symbol.

Digital Internal Use Only

DOCTOR/PS—Modify PostScript files

This utility allows to add blank pages to PostScript files when required to have a balanced set of odd/even pages. It also allows to replace the prolog part of a PostScript file or to produce an alternative output file that consists of a subset of pages.

FORMAT

DOCTOR/PS *PostScript_file [/qualifiers]*

Command Qualifiers	Defaults
/BLANK[=filespec]	*/NOBLANK*
/CHANGE_PROLOG	*/NOCHANGE_PROLOG*
/EXTRACT=range	*None.*
/[NO]FIGURES=keyword	*/FIGURES*
/LEADING_BLANK	*NOLEADING_BLANK*
/LOG	*/NOLOG*
/OUTPUT=filespec	*next higher version of input file*
/PROLOG=filespec	*None.*
/NOPROLOG	*None.*
/SADDLE	*/NOSADDLE*
/2UP	*/NO2UP*

PARAMETERS

PostScript_file

Input file to be scanned through and to produce an output file from. The file must adhere to the Adobe *minimal conformant* conditions V2 or V3, as described in the *PostScript Reference Manual*, published by Addison & Wesley. See Section 6.7.

restrictions

- Input file must be minimal conformant PostScript coded.
- Input file may not be an encapsulated PostScript file: must have proper prolog and trailer.

QUALIFIERS

/BLANK[=filespec]

This qualifier makes sure the output file consists of a balanced set of odd and even pages to facilitate proper printing on a double sided printing device. It will also append a blank page if the final page is odd. It will also prefix a blank page if an extract range starts on an even page or append a blank page if a range ends on an odd page.

Digital Internal Use Only

You can specify a file specification that contains a complete description of a blank page. This file should not include the `%%Page:` comment, but should otherwise be complete in itself. If no file is specified, a simple blank page is produced through the `showpage` PostScript command.

/CHANGE_PROLOG

This qualifier allows to replace the standard prolog with one that defines an additional line at the top (header) and bottom (footer) of each page, as well as a faint grey diagonal text line across the page. In addition these changes can be saved in a new prolog file to be created, that can be used in subsequent cases through `/PROLOG` qualifier.

The `/CHANGE_PROLOG` qualifier takes twelve arguments, all optional, but at least one of them must be specified:

- TOP=text - specifies the text line to be printed as a header on each page. If the text contains blanks, the entire text string should be within quotation marks.
- BOTTOM=text - specifies the text line to be printed as a footer on each page. If the text contains blanks, the entire text string should be within quotation marks.
- COUNTER=integer - specifies the starting number to be printed on the bottom line, following the text specified with BOTTOM. The number increments with each page and can be used for numbering pages that would otherwise be unnumbered.
- DIAGONAL=text - specifies the text line to be printed in a faint grey font diagonally across each page and underlaying the text on the page. If the text contains blanks, the entire text string should be within quotation marks.
- OUTPUT=filespec - If specified, the new prolog is also output as a separate file. This file can be used in subsequent occassions by inputting it through the `/PROLOG`=filespec qualifier.
- HSIZE=pointsize - If specified, determines the horizontal page width used to accomodate the top line. If omitted, A4 sized paper is assumed.
- VSIZE=pointsize - If specified, determines the vertical page width used to accomodate the bottom line. If omitted, A4 sized paper is assumed.
- PAGESIZE=papertype - This allows specification of a particular standard sized papersheet. It is mutually exclusive to both HSIZE and VSIZE. The papersizes recognized are LETTER, LEDGER, LEGAL, EXECUTIVE, 7X9, 35MM, A5, A4, A3, B5, B4, C6, C5 and C4.
- BORDER - if specified the underlay page is surrounded with a thin rectangle: the text is printed within this rectangle. The default is NOBORDER.

Digital Internal Use Only

- GRAYSCALE=number - If used, you must specify a number between 00 and 99. This is translated into a percentage (00-99%) indicating the grey scale used for the diagonal text. A value of 00 is black, 99 is almost white. If omitted, it defaults to 95%: a light grey text. Note that you must enter a leading zero for percentages 01-09, as a simple "0." is prefixed to the number, making a "1" into "0.1" - i.e. 10%.
- FONT=fontname - If used, a valid PostScript font must be specified to be used for the top, bottom or diagonal line. For a full list of keywords see Table 6–1. When omitted, Helvetica-Bold is used.
- SIZE=number - allows specification of the top and bottom line font size. When omitted, it defaults to 15 points. Values above 20 points are pointless, but allowed.

The `/PROLOG` and `/CHANGE_PROLOG` qualifiers are mutually exclusive.

/EXTRACT=range

This qualifier allows to copy a subset of pages from the original file into the output file. The range can be specified by giving the start of the range (compulsory) and either the number of pages to copy from there onwards or the page number of the final page. For this purpose the qualifier takes three arguments:

- START=folio - specifies the starting page number. E.g. III or 4-6 or INDEX-6. The valid numbers are those that are available in the input PostScript file. If uncertain, do a

  ```
  $ SEARCH postscript_file  %%Page:
  ```

  command to see which page folio numbers are available. Do not specify the opening and closing parenthesis if the folio numbers are surrounded by these. (Adobe V3 comment standards only).
- END=folio - specifies the last page of a range of pages to be printed. A valid folio number must be specified, otherwise the DOCTOR will include all remaining pages upto the end of the file.
- NUMBER=integer - specifies the number of pages to extract from the specified starting folio onwards. If this number is larger than the remaining number of pages, the rest of the file is copied.

Either END or NUMBER must be present. If both are present, the shortest range span from START to either END or NUMBER is taken. If neither are present, the remainder of the file, from the START folio onwards, is copied.

The EXTRACT qualifier can be specified several times to allow for several ranges to be included in the same output file.

Digital Internal Use Only

/[NO]FIGURES

This qualifier allows manipulation of properly inserted encapsulated PostScript figures. The /NOFIGURES qualifier removes the figures from the input file. The /FIGURES (default) leaves them untouched. By adding the keyword EXTRACT to either /FIGURE or /NOFIGURE the figures are written into separate figure files as encapsulated PostScript. They receive their original names (displayed when /LOG is specified). If also the keyword PAGE_NUMBER is specified, the figures get the same file name as the input file, but with the page number appended to it. This allows for easy reference of the figure to the printed document page.

/[NO]LEADING_BLANK

This qualifier will produce a blank page before the actual document starts. This is required when the PostScript output file is to be used for reduced size printing (with 2 document pages on a single side of a sheet of paper). This qualifier is only required when the output file is to include the first page of the document also. Default is /NOLEADING_BLANK. The blank page produced does not use a user-specified blank page file that can be specified with /BLANK.

/[NO]LOG

Outputs informational messages on blank pages that are inserted.

/OUTPUT=filespec

Specifies the output file that will be created. If omitted, the next higher version of the input file is created and a message is issued to warn the user not to purge (which deletes the original, complete file).

/PROLOG=filespec

Specifies the text of the prolog part of the output PostScript file. When specified, the original prolog part is removed and the contents of the specified file is included. This file must be complete and adhere to the PostScript conformant style (see Section 6.7. The final line should be %%EndProlog. A simple prolog file, using a header, footer and diagonal line, can be created through the /CHANGE_PROLOG qualifier.

If /NOPROLOG is specified, the entire prolog is omitted from the output file. This way one can include the output file into another PostScript file, e.g. through the <FIGURE_FILE>(ps\filespec\size) tag. In this case, the output file type is defaulted to .EPS instead of .PS.

The /PROLOG and /CHANGE_PROLOG qualifiers are mutually exclusive.

/SADDLE

The use of this qualifier is mutually exclusive with all others that are allowed with the /PS qualifier. It allows to re-order pages inside the PostScript file for differently ordered output. This output can be used to produce brochures by folding pages in the middle (to become a signature)

Digital Internal Use Only

and saddle stich them or by driving a staple through the spine. This requires that printing is done with two pages on a single sheet side. For more information see Section 6.6. Default is /NOSADDLE.

Allowed qualifiers with /SADDLE are:

/LOG	Produce additional output during processing. Default is /NOLOG.
/OUTPUT=filespec	Specify output file specification. Default is the next higher version of the input file
/GATHER=number	Number must be a multiple of 4. It indicates the range of pages that are printed as a single signature. When not specified it defaults to the number of pages in the PostScript file.

/2UP

For PostScript files produced by VAX DOCUMENT V2.1 it is possible to insert additional code that allows you to print the file on double sized paper (e.g. A3) with two pages on a side (eg. two A4 pages). This only works for VAX DOCUMENT V2.1 produced PostScript files and the qualifier must be specified following the /PS/SADDLE.

EXAMPLES

The next example shows how blank pages are added to an existing PostScript file and the result is copied into BALANCED.PS.

1

```
$DOCTOR/PS  myfile.PS /BLANK/OUTPUT=balanced.PS
```

The following example shows the production of a file that contains only the table of contents, 5 pages of Chapter 2 and the Index (which is the final part of the book) of an existing PostScript file. Blank files are added where needed.

2

```
$DOCTOR/PS book.ps /EXTRACT=(START=III,END=XIV) -
                      /EXTRACT=(START=2-1, NUMBER=5) -
                      /EXTRACT=(START=INDEX-1) -
                      /OUTPUT=SUMMARY.PS /BLANK
```

The following example shows the production of a file that has a bottom line added "Internal Use Only - Rev. B" and diagonal text "Review Only", which is printed with a 90% grey scale (light grey. 00 would be black, 99 almost invisibly white).

Digital Internal Use Only

3
```
$DOCTOR/PS book.ps -
                      /CHANGE=(BOTTOM="Internal Use Only", -
                               DIAGONAL="Review Only", -
                               GRAYSCALE=90) -
                      /OUTPUT=SUMMARY.PS /BLANK
```

The next example shows how a 29 page book is converted into a 32 page booklet at half size by printing it on an LPS20 with two pages on each side and double sided:

4
```
$DOCTOR/PS/SADDLE book.ps /GATHER=32/OUTPUT=booklet.PS
$PRINT/QUEUE=LPS20/PARAMETER=(DATA=POST,NUMBER=2,SIDE=TUMBLE)
```


Digital Internal Use Only

DOCTOR/SDML—Markup Files Included

The DOCTOR/SDML, scans through the VAX DOCUMENT source files for other sources included by them, upto a nesting level of 20. These included files can be VMS files or CMS elements, if DEC/CMS is installed on the system. These files can be any of the ones normally accepted through standard tags.

It reports the document structure as a comment block in the source file and can produce an MMS build file as well as a list of all the occurrences of index tags.

FORMAT

DOCTOR/SDML *filespec [/qualifiers]*

Command Qualifiers	Defaults
/CMS[=generation]	*/NOCMS*
/DOCTYPE=doctype	*/DOCTYPE=(PAPER=REPORT, ONLINE=SOFT.ONLI*
/DESTINATION=destination	*defaults: LN03, PS, BOOKREADER, ROFF, MAIL, LI*
/IGNORE=(options)	*/NOIGNORE*
/INCLUDE=filespec	*/NOINCLUDE*
/INDEX=filespec	*/NOINDEX*
/LOG	*/NOLOG*
/MMS[=filespec]	*/NOMMS*
/OUTPUT=filespec	*/OUTPUT=input_filespec*
/RANDOM_TEXT	*/NORANDOM_TEXT*
/RULES	*/RULES*

PARAMETERS

filespec

A valid VMS file specification. Wildcards are allowed. In that case all files matching the specification are processed individually.

restrictions

- The specified file specification on the command line must be a normal VMS filespec. If it resides in CMS, you need to fetch it from there first. Any file referenced by the specified file will be fetched by DOCTOR/SDML itself.
- If DEC/CMS is not installed on your system DOCTOR/SDML can only inspect VMS files, not CMS elements. See also Preface.
- DOCTOR/SDML cannot inspect any .STT (precompiled tag definition) file due to the compressed tag definitions in those

Digital Internal Use Only

- Filespecs must be explicit. DOCTOR/SDML will not recognize filespecs that are defined within a symbol and refered to by <REFERENCE>(symbol) tags.
- DOCTOR/SDML will notice <INCLUDE_TEX_FILE> tags but will not process any of the specified TEX files.

prompts

File: filespec

DESCRIPTION

DOCTOR/SDML reads the specified VAX DOCUMENT source file and scans for those tags that include other files. These tags are <ELEMENT>, <INCLUDE>, <FIGURE_FILE>,<INCLUDE_TEX_FILE>, <ICON_FILE>, <EXAMPLE_FILE>, <TABLE_FILE> and <INCLUDES_FILE>. The tags introduced in VAX DOCUMENT V1.1 are also supported: <FILE_SPEC> and <MEMO_FILE>. When the qualifier /INDEX is specified, it also looks for <X> and <Y>.

It will then open those files and repeat the process, upto 20 levels deep. When finished, it will write a special comment header at the start of the source file, indicating the structure of the document and the files it refers to. When DOCTOR/SDML is run several times over the same file, this comment block is refreshed, if left unmodified by the user (it recognizes the

If DEC/CMS is used on your system, DOCTOR/SDML can also attempt to look up the included files in your current CMS libraries if you specify the /CMS qualifier. If a generation or CMS class is specified, only elements that belong to that generation are looked up.

Apart from information inside the source file, it can also generate a description file for MMS to process the file scanned through or any of its parts.

QUALIFIERS

/[NO]CMS[=generation]

Using this qualifier will urge DOCTOR/SDML also to look into your currently set CMS library if any of the files specified in the SDML source files cannot be found in the specified directories. After inspections, these fetched files are deleted again. The ones found in CMS are marked with "(CMS Library)" in the comment header.

It only looks into CMS for included files: the "main" source file you specified on the command line must exist as file in your directory and will not be fetched from CMS. Default value is /NOCMS.

If the error message %MARFIN-F-NOCMSSUP occurs when /CMS is specified, it means that DEC/CMS is not installed or available for use on the system. See Preface.

Digital Internal Use Only

If with /CMS you also specify a CMS class name, then DOCTOR/SDML will only look into the CMS library for elements that belong to this class. In all other cases, the required file is reported as not found.

/DESTINATION

This qualifier accepts four keywords, each allowing to modify the default destination used for DOCUMENT processing:

- BOOKREADER= . Defaults to BOOKREADER for online destinations
- POSTSCRIPT= . Defaults to POSTSCRIPT for PostScript destinations
- LN03= . Defaults to LN03 for LN03 destinations
- MAIL= . Defaults to MAIL for MAIL destinations
- LINE_PRINTER= . Defaults to LINE_PRINTER for line printer destinations
- ROFF= . Defaults to ROFF for Roff destinations

If more than one destination is modified, the individual keywords are separated by commas.

/DOCTYPE

You can specify the type of document to build for paper or online destinations. It takes three keywords:

- PAPER= . Accepts any valid paper doctype name. Defaults to REPORT.
- ONLINE= . Accepts any valid online doctype name. Defaults to SOFTWARE.ONLINE
- MANPAGE= . Accepts any valid manpage doctype name. Defaults to MANPAGE

/IGNORE=(keyword-list)

When specified, DOCTOR will also scan certain blocks of text that are normally ignored. By default any <COMMENT> or <LITERAL> block is not scanned for file inclusion tags. Specifying /IGNORE will allow you to force DOCTOR to scan one or both of these block types. The allowed option keywords are LITERAL and COMMENT. When both are specified, they must be separated by a comma: /IGNORE=(COMMENT,LITERAL).

/INCLUDE=filespec

Specifies an MMS include file that allows the user to specify additional rules, macros or symbols. Results in a .INCLUDE line in the MMS description file.

Digital Internal Use Only

/INDEX=filespec

Instructs DOCTOR/SDML also to produce a file containing all the index tags, together with their occurrence (file spec and line within file).

/[NO]LOG

This outputs additional informational messages on each included file during the processing of DOCTOR/SDML. Default is /NOLOG.

/[NO]MMS[=filespec]

This generates an MMS description file to rebuild the document. This file has the same filename as the source file processed, but with file type .MMS. A filespec specified to this qualifier will take precedence. You should not specify filenames and/or types if the input file specification contains wildcards.

/OUTPUT=filespec

When specified it will produce the new version of the SDML file (that will contain the DOCTOR/SDML header) as the specified filespec. When omitted, a new version of the input file parameter is made. Note that when wildcard input file specifications are used one should not specify filenames and/or types.

/RANDOM_TEXT

Adds some additional random text to the /INDEX produced output to allow processing of the file by VAX DOCUMENT without running into TEX memory exceeded errors. Only valid in combination with /INDEX.

/RULES

Specifying /NORULES suppresses the generation of MMS suffixes and default action rules to retrieve elements from CMS libraries. By default, /RULES is used.

Digital Internal Use Only

DOCTOR/TAG_COUNT—Count all tags in SDML file

The DOCTOR/TAG_COUNT utility counts all tag constructs inside an SDML document. This document may be a complete book (profile plus elements). It reports the counted tags in the produced output file.

FORMAT

DOCTOR/TAG_COUNT *filespec [/qualifiers]*

Command Qualifiers	Defaults
/OUTPUT	*/OUTPUT=.TAG_COUNT*
/LOG	*/NOLOG*

PARAMETERS

filespec

A valid VMS file specification of an SDML file. No wildcards are allowed. File may be the profile of the book and may refer to other SDML files.

restrictions

- Any tag-like construct is seen and counted, including those inside <COMMENT> or <LITERAL> blocks.

prompts

File: filespec

DESCRIPTION

DOCTOR/TAG_COUNT scans through the specified input SDML file and all its referenced SDML files to count all tags. Tags are any consecutive number of alphabetic or numeric characters or underscores, surrounded by angle brackets. It reports its findings in the output file. For details see Chapter 8.

QUALIFIERS

/LOG

This qualifier result in additional messages on files being processed. By default no such messages are issued (/NOLOG).

/OUTPUT=filespec

This qualifier specifies the output file to generate from the input file. When omitted, the output file take the same values as the input file, except for the file type, which becomes .TAG_COUNT. It is allowed to specify only a part of a filespec. Missing fields default from the input fields (except the type part).

Digital Internal Use Only

DOCTOR/XREF—List cross reference symbols

The DOCTOR/XREF utility enables you to produce a file with <DEFINE_SYMBOL> tags of all symbols defined within a book, derived from its .XREF file.

It also enables you to produce a listing of all symbols, sorted alphabetically or by symbol type (chapter, table, example etc)

FORMAT

DOCTOR/XREF *xref_filespec [/qualifiers]*

Command Qualifiers	Defaults
/ALPHABETIC	*/ALPHABETIC*
/BOOKTITLE	*none*
/FULL	*/NOFULL*
/LIST	*/NOLIST*
/LOG	*/NOLOG*
/NUMERIC	*/NUMERIC*
/SYMBOL_FILE	*/NOSYMBOL_FILE*
/PREFIX	*none*
/SORT	*/NOSORT*
/SYSTEM_SYMBOLS	*/SYSTEM_SYMBOLS*
/TEXT	*/NOTEXT*
/VALUE	*/VALUE*

PARAMETERS

xref_filespec

A valid VMS file specification of a .XREF cross reference file, produced by VAX DOCUMENT during a book built. Default file type is .XREF. No wildcards are allowed.

restrictions

- It is assumed the input file is a proper .XREF file produced by VAX DOCUMENT
- At least one of the qualifiers /SYMBOL_FILE, /LIST or /SORT must be specified in order to have DOCTOR/XREF perform any action.

prompts

File: filespec

DESCRIPTION

DOCTOR/XREF inspects a valid DOCUMENT .XREF file produced during a bookbuild operation of a multiple file document. It allows the production of either a file where all symbols are defined once more explicitly to allow

Digital Internal Use Only

you to incorporate those symbols into another set of documents and enable cross referencing between books. The qualifier /SYMBOL_FILE is then required.

In addition it allows you to produce tabular listing of all symbols defined in a document, for reference by the author or writers. These tables are generated as .SDML files with <TABLE> constructs that can subsequently be processed individually by DOCUMENT using the REPORT doctype style.

QUALIFIERS

/ALPHABETIC

Only valid when also /SORT is specified. It indicates whether the sorted listing should be ordered by symbol name or by the object or section number. When none is specified, /SORT/ALPHABETIC is assumed. To get both features in one file, both also /SORT must be specified.

/BOOKTITLE="string"

A quoted string specifying the name of the book from which the cross reference file originates. This text is used during referencing of the symbols, to output e.g. "See *The Life of Brian* Chapter 4, Mothers..." The text specified is defined as a symbol *prefix*_BOOK to allow references to it. The prefix is set by /PREFIX. Default is an empty string.

/FULL

Defines the symbol value to consist of name, number and full text (e.g. *Chapter 4, Where it all happens*). Default is /NOFULL. See also /VALUE.

/LIST[=filespec]

Specifies the filespec of a normal VAX DOCUMENT file in which all symbols defined in the .XREF input file are listed in alphabetical order. This output can be processed through VAX DOCUMENT and used by authors and writers for their own reference. When no filespec is given, it defaults to the same name as the input file, but with filetype .SDML_SYMBOL_LIST. Default is /NOLIST.

/LOG

Outputs additional informational messages. Default is /NOLOG.

/NUMERIC

Only valid when also /SORT is specified. It indicates whether the sorted listing should be ordered by symbol name or by the object or section number. When omitted, /SORT/ALPHABETIC is assumed. To get both features in one file, both qualifiers must be specified.

/PREFIX=filespec

A short string to prefix to all symbols defined in the input file to make them unique. If omitted no prefix is used, but symbols from several

Digital Internal Use Only

books may have been defined identically, causing referencing conflicts. A warning is then issued.

/SORT[=filespec]

Similar to /LIST. All symbols are again listed, but now grouped by their class (figure, table, chapter etc). When no filespec is given, it defaults to the same name as the input file but with filetype .SDML_SYMBOL_TYPE.

/SYMBOL_FILE[=filespec]

Specifies the output filespec of the file that will contain all the <DEFINE_SYMBOL> definitions. If the filespec is omitted, it defaults to your current directory, with filename identical to the input file, but file type .SDML_XREF.

/SYSTEM_SYMBOLS

For its own use, DOCUMENT also generates a number of symbols (like highest section number in document, page prefix, online chunk). These are normally not of interest for the user and therefore omitted in the DOCTOR /XREF output. Specifying /SYSTEM_SYMBOLS also includes those. Default is /NOSYSTEM_SYMBOLS.

/TEXT

Defines the symbol value to consist of the text part of the symbol only. (e.g. "Where it all happens"). Default is /NOTEXT. See /VALUE.

/VALUE

Defines the symbol value to consist of the name and value number only (e.g. Chapter 4). This is the default if no /TEXT or /FULL is specified.

Digital Internal Use Only

A Error messages

A.1 DOCMSG Messages

CREATED, Created 'string'

Severity: INFORMATIONAL

Explanation: Indicates the successful creation of the output file specified.

User Action: None.

PASS1, Start Pass 1: Read all messages and sort them alphabetically

Severity: INFORMATIONAL

Explanation: Scans through the source .MSG file and registers all messages defined.

User Action: None.

PASS2, Start Pass 2: Generate message file

Severity: INFORMATIONAL

Explanation: Using the information gathered in Pass 1, the output SDML file with messages is constructed for each facility and each messages output in alphabetical order.

User Action: None.

A.2 DOCTOR Messages

IDENT, This is 'string' version 'string'

Severity: INFORMATIONAL

Explanation: Indicates the current version of the utility that is invoked. This is important when bugs and wishes are expressed to the development team.

User Action: Specify number when submitting SPR's or QAR's.

NOTIMPL, Functionality 'string' not yet implemented

Severity: INFORMATIONAL

Explanation: The specified functionality is not yet implemented, even though the command interface is already present.

User Action: Wait till the next release.

Digital Internal Use Only

SENDQAR, Internal error. Please send QAR. Error ID: 'string'

Severity: INFORMATIONAL

Explanation: You managed to give a set of input data that caused the utility program to go to "catch all" statements that should never be reached.

User Action: Please notify the developers by SPR or QAR, describing the error and specifying the version of the software used, the error messages displayed and possibly a sample of the input files that caused the error to occur, together with the specified command statement.

A.3 DTAGCOUNT Messages

CREATED, Created 'string'

Severity: INFORMATIONAL

Explanation: Indicates the successful creation of the output file specified.

User Action: None.

FNF, File 'string' is referenced in document but does not exist - will ignore

Severity: WARNING

Explanation: Indicates that the document contains a <INCLUDE> or <INCLUDES_FILE> or <ELEMENT> tag refering to a file that currently does not exist at that location.

User Action: Make file available or correct the SDML file by removing the reference to it. The DOCTOR/TAG utility will ignore this file in its counting procedure.

SCAN, Scanning 'string'

Severity: INFORMATIONAL

Explanation: Indicates the current SDML file being processed in the search for tags used in it.

User Action: None.

A.4 GLOSSARY Messages

CREATED, Created 'string'

Severity: SUCCESS

Explanation: Indicates the successful completion of the utility and the creation of the specified output file.

User Action: None.

DUPLICATE, Duplicate entry for <GTERM> ''string'' in file

Severity: FATAL

Explanation: The SDML file that contains the glossary contains multiple identical entries defined by <GTERM>. This makes sorting impossible.

User Action: Remove duplicates or combine them to a single <GTERM> entry with a <LIST>(numbered) to indicate the different meanings of the word.

GTERMOUTBOUND, Found <GTERM> tag outside <GLOSSARY> - <ENDGLOSSARY> range

Severity: FATAL

Explanation: DOCTOR found <GTERM> tags ouside the valid range of <GLOSSARY> and <ENDGLOSSARY>. This is invalid for DOCUMENT and also unacceptable for DOCTOR.

User Action: Move the <GTERM> tags inside the proper glossary block.

NEXTGLOSSARY, Multiple <GLOSSARY> tags encountered. Can only sort one block

Severity: FATAL

Explanation: Several <GLOSSARY> tags are encountered in the specified input file. Because DOCTOR can only sort a single block between the first <GLOSSARY> and <ENDGLOSSARY> tag, the current file is not acceptable.

User Action: Combine both glossary blocks into a single one or split them into two or more files, which are included by <INCLUDE> into the final document. DOCTOR can sort each one of the glossary files individually.

NOALPHA, Found term "'string'" without alphabetic characters. Do not use /SORT=NONALPHA=IGNORE

Severity: FATAL

Explanation: The sorting process is conducted either explicitly or by default to ignore nonalphabetic characters. The current term consists solely of nonalphabetic characters and can therefore not be sorted as a null string.

User Action: Specify /SORT=NONALPHA=BEFORE or AFTER to allow the term to occur at the start or end of the glossary. Alternatively, remove the term.

NOENDGLOSSARY, No closing <ENDGLOSSARY> tag found

Severity: WARNING

Explanation: During the scanning of the input file, no <ENDGLOSSARY> tag was found.

User Action: Add <NOGLOSSARY>. Or ignore, if the glossary items are included via an <INCLUDE> tag that is positioned between the <GLOSSARY> and <ENDGLOSSARY>.

Digital Internal Use Only

NOGLOSSARY, Opening tag <GLOSSARY> not found before first <GTERM> tag

Severity: WARNING

Explanation: DOCTOR discovered a <GTERM> without a preceding <GLOSSARY>.

User Action: None. User may wish to check whether elsewhere in the document the tag <GLOSSARY> is specified. There must be one if this document is to be processed correctly by VAX DOCUMENT.

NOGTERMS, Input file does not contain <GTERM> tags

Severity: FATAL

Explanation: The SDML input file does not contain any <GTERM> tag to sort on.

User Action: Check on use of proper file or add glossary terms

SAMEFILESPEC, Output file is next higher version of input file.
Purging will lose original file.

Severity: WARNING

Explanation: You did not specify an explicit output file, hence the utility defaults to the input file specification. However, this may cause the lost of the original input file is an accidental purge command is issued.

User Action: Rename the latest version into another file name or copy the file to another location to avoid losing the original copy.

A.5 MARFIN Messages

ATLAST, Created 'string'

Severity: INFORMATIONAL

Explanation: Indicates the creation of a new version of the main VAX Document file Marfin has finished reading.

User Action: None

CREATED, Created 'string'

Severity: INFORMATIONAL

Explanation: Indicates the successful completion of the utility and the creation of the specified output file.

User Action: None.

Digital Internal Use Only

FNF, File 'string' referenced, but not found

Severity: WARNING

Explanation: In trying to locate all files that were referenced by the document, the indicated number could not be found. In the comment header of the SDML file specified on the command line, you can see which ones they are.

User Action: Either remove the indicated references or retrieve/create the files required.

HUMANS, Humans - always giving the wrong commands.
Marfin do this, Marfin do that

Severity: FATAL

Explanation: Invalid DCL parameters or combination of qualifiers were given.

User Action: Specify a correct command line. Read the manual!

IGNORED, 'string'

Severity: WARNING

Explanation: At the command line two definitions for the same entity are made (e.g. destination specified with /DESTINATION as well as for the document type /POSTSCRIPT). In this case the message indicates what is ignored.

User Action: Remove one of the conflicting definitions or accept the set default.

MMSTOO, Made MMS description file 'string'

Severity: INFORMATIONAL

Explanation: Indicates Marfin also created an MMS description file to automate the building of the VAX Document source file

User Action: None. Use this file with MMS/DESCRIPTION=file.MMS

MUMBLE, 'string'

Severity: INFORMATIONAL

Explanation: Marfin is having one of his moods.

User Action: Bear with dignity after all the work he did for you

NOCMSSUP, DEC/CMS not available. Reissue command without /CMS

Severity: FATAL

Explanation: DOCTOR/SDML could not find the DEC/CMS product library and therefore assumes this product is not installed on your system.

User Action: Reissue the command without /CMS. In addition, you may want to install (after purchase) DEC/CMS as version management system for your document files. After that, DOCTOR/SDML/CMS will operate on these libraries too.

Digital Internal Use Only

NOTFOUND, There were 'unsigned decimal number' files referenced that could not be found

Severity: WARNING

Explanation: In trying to locate all files that were referenced by the document, the indicated number could not be found. In the comment header of the SDML file specified on the command line, you can see which ones they are.

User Action: Either remove the indicated references or retrieve/create the files required.

OHMYDIODES, 'string'

Severity: FATAL

Explanation: Internal programming errors.

User Action: Please submit bug report to `IJSAPL::KLERK` or `Theo de Klerk @UTO`.

OHNO, I won't enjoy processing 'string'

Severity: INFORMATIONAL

Explanation: Indicates the main VAX Document file Marfin tries to read

User Action: None

A.6 ONLINE Messages

CREATED, Created modified file 'string'

Severity: INFORMATIONAL

Explanation: The modified input file(s) is written out as new file.

User Action: None.

CRELIST, Created list file 'string'

Severity: INFORMATIONAL

Explanation: The list input file with actions to perform is written.

User Action: Read the file and comply with the actions suggested.

FNF, File to include 'string' not found - will be ignored.

Severity: WARNING

Explanation: A <ELEMENT> or <INCLUDE> specifies a file that must be included in the document, and as such would also be inspected by DOCTOR/ONLINE. However, the

given file spec does not refer to an existing file. DOCTOR/ONLINE will not attempt to find and inspect the specified file.

User Action: the specified file spec may be incomplete (e.g. only name and type mentioned) and you may currently execute DOCTOR from another directory. In that case, you may want to change your default directory to the correct one for DOCTOR to find the files specified.
Otherwise either locate the specified file and start again or ignore it.

INPUTFNF, Input file 'string' not found

Severity: FATAL

Explanation: The specified input file cannot be found.

User Action: Specify existing input file. Correct spelling or device/directory specification.

INSPECT, Inspecting 'string'

Severity: INFORMATIONAL

MODIFY, Modifying 'string' to be compliant for BOOKREADER destinations

Severity: INFORMATIONAL

Explanation: The original input file(s) are modified to be properly formatted for DOCUMENT online processing

User Action: None.

PASS1, Collecting label and reference information

Severity: INFORMATIONAL

Explanation: The input file and the files it references are inspected for any labels defined and referenced, to build a data structure to be used in Pass 2.

User Action: None.

PASS2, Generating labels, references and figure_file tags

Severity: INFORMATIONAL

Explanation: The input file and the files it references are inspected and missing labels are inserted. Unreferenced labels are referenced.

User Action: None.

Digital Internal Use Only

REMOVE, Removed label from 'string' tag

Severity: WARNING

Explanation: Some section tags have no label associated with them in VAX DOCUMENT V1.* . They are allowed in V2, but the /VERSION_1 qualifier that was used caused it to be removed.

User Action: None, or do not specify /VERSION_1 qualifier. In this case, the output can only be processed by VAX DOCUMENT V2.*

A.7 PSSCAN Messages

ABSENTPAGESCHEME, Page number indicated as 'string'
Blank page insertion mechanism may not work properly here

Severity: WARNING

Explanation: It is allowed by PostScript comment standards to define a page comment as '%%Page: ? n' or '%%Page: "" n' if or any other text string without an explict page number specified and printed on the physical page. The '?' or "" is replaced by DOCTOR/PS to be "?-0" to allow proper processing of the page as if it were a known page number system. However, since it is unknown whether "?" pages are odd or even, there is no way to know whether additional blank pages should be inserted.

User Action: None. The use of /BLANK on the input file is probably not useful in this context.

CONFORM, File assumed to conform to 'string'

Severity: INFORMATIONAL

Explanation: Input PostScript file seems to conform to the Adobe Inc. (minimal) conformant coding standards.

User Action: None.

CREATED, Created 'string'

Severity: SUCCESS

Explanation: Indicates the successful completion of the utility and the creation of the specified output file.

User Action: None.

Digital Internal Use Only

CREATOR, Input file was created by 'string'

Severity: INFORMATIONAL

Explanation: This message indicates the origin of the PostScript file that is currently being processed. Depending on this origin, some features of DOCTOR/PS may not be used. All VAX DOCUMENT produced files will be handled properly.

User action: None

INSERTED, Page 'string' inserted

Severity: INFORMATIONAL

Explanation: Indicates that a blank page was inserted into the output document to balance the number of odd/even pages.

User Action: None.

INVALCOUNT, Invalid page counter 'signed decimal number' . Must be positive integer

Severity: FATAL

Explanation: The page counter to be added to the footer must be a positive integer. Negative values and zero are not allowed.

User Action: Specify positive integer from 1 upwards.

INVORDINAL, Input file violates Adobe standard: ordinal starts at 'signed decimal number'

Severity: WARNING

Explanation: The comments of the first page inside the PostScript file should indicate the folio number as well as the sequence number of the page within the file. The first page is assumed to have sequence number 1. For some reason, the program that output the PostScript file did not start counting from 1 upwards, but from the indicated number.

User Action: None - The DOCTOR will rectify this in the output file it produces. You may wish to file a Problem Report to the producers of the software package that made your PostScript file, stating they violate the Adobe programming standards.

INVPAGCOM, Page comment "'string'" at line 'unsigned decimal number' not conforming to Adobe PostScript standards. Ignored as page marker.

Severity: WARNING

Explanation: The input file contains invalid conformant coding comments, whereby the utility cannot interpret the page data correctly. Although DOCTOR will continue, it ignores this line as page and assumes there is no new page that starts.

User Action: Inspect the final output. It may not be exactly what you wanted, but it may. This can be checked through the

Digital Internal Use Only

```
$ SEARCH filespec %%Page:
```

DCL command and see if all pages are correctly odd/even positioned.

NEGPAGNO, Invalid page number to convert: 'signed decimal number'

Severity: FATAL

Explanation: A page number was passed to a routine to be converted to an ASCII Roman of Arabic numeral. This error occurs when a negative number is passed from the PostScript file or through an internal error.

User Action: See DOCTOR SENDQAR message if the PostScript file does not contain negative page numbers.

NOCHANGE, Input file needed no modification. No new file generated

Severity: INFORMATIONAL

Explanation: The operation of the utility on the input file resulted in an identical copy for the output. Therefore no output is generated. The message is also issued when one or more page ranges should be extracted but not a single range could be located.

User Action: Be happy with the current PostScript file or specify correct page ranges.

NOPAGE, File contains no valid %%Page: markers and is not Adobe Coding Standard conformant

Severity: WARNING

Explanation: Despite the fact that the file starts with a `%!PS-Adobe` header to suggest it conforms to Adobe's Structured Coding PostScript standards,

it does not, because individual pages are not marked by a `%%Page` **:** comment line. Therefore DOCTOR is not able to distinguish between pages for extraction or re-ordering.

User Action: Try to create the PostScript file again from the text processor tool, enforcing Adobe standards. If MS-Windows applications are used, the Print Setup... menu will allow you (after various layers of Options... or Setup... sub menus) to check the box for "Adobe Structured Coding Standards". Once checked, output the PostScript file again.

NOPROLOG, File does not contain valid %%EndProlog comment. Not Adobe conformant

Severity: WARNING

Explanation: the `%%EndProlog` comment line fails in the PostScript file. As such, it is not conforming to the Adobe Structured Coding standards.

User Action: Try to create the PostScript file again from the text processor tool, enforcing Adobe standards. If MS-Windows applications are used, the Print Setup... menu will allow you (after various layers of Options... or Setup... sub menus) to

check the box for "Adobe Structured Coding Standards". Once checked, output the PostScript file again.

NOTCONFORM, File is not (minimally) conforming to Adobe PostScript standards. No output created.

Severity: FATAL

Explanation: The specified input PostScript file does not seem to adhere to the Adobe Inc. minimal conformant coding standard. Hence the utility cannot correctly interpret the input file (even if it is correct PostScript code).

User Action: Whatever you wanted to do must be done manually.

NOTVAXDOC, /CHANGE used on PS file not produced by DOCUMENT or DECwrite. Success is not guaranteed

Severity: SUCCESS

Explanation: The /CHANGE is only guaranteed to work for VAX DOCUMENT and DECwrite produced PostScript files. Depending on the adherence to standards by other PostScript creators, /CHANGE may also work successfully here, but no success is guaranteed.

User Action: None. Use at your own risk.

PASS, Starting input file scan Pass 'unsigned decimal number'

Severity: INFORMATIONAL

Explanation: Because so many PostScript files have figures and plot files included that all have their own %%Trailer commment, it is necessary to find the very last %%Trailer position in the file before starting the real processing.

User Action: None.

RANGESKIP, Page range starting at 'string' not found

Severity: WARNING

Explanation: Issued at the end of the processing of a PostScript file from which a range of pages had to be extracted. The starting page number could not be located in the input file, hence the entire range is not included.

User Action: Specify correct starting page.

Digital Internal Use Only

SAMEFILESPEC, Output file is next higher version of input file.
Purging will lose original file.

Severity: WARNING

Explanation: You did not specify an explicit output file, hence the utility defaults to the input file specification. However, this may cause the lost of the original input file is an accidental purge command is issued.

User Action: Rename the latest version into another file name or copy the file to another location to avoid losing the original copy.

SHORTRANGE, Two end range values specified.
Using shortest of 'unsigned decimal number' and 'string'

Severity: INFORMATIONAL

Explanation: You specified the /EXTRACT qualifier with both an END and NUMBER keyword. Whichever of the two indicates the smallest number of pages from the specified START page, will be used.

User Action: Accept or respecify the /EXTRACT range, using only one of the two ending keywords END or NUMBER.

SKIPFIGURE, Removing figure from output document: 'string'

Severity: INFORMATIONAL

Explanation: The specified figure file is removed from the PostScript output to be created. This allows conversion tools to reconstruct the original text file without the graphics information inserted from other tools.

User Action: None

STARTRANGE, Including page range starting at 'string'

Severity: INFORMATIONAL

Explanation: Informs you that the specified /EXTRACT range was found and is activated. The specified range of pages will be copied to the output file.

User Action: None.

TRUNCATED, Resulting figure file name too long. Truncated to 'string'

Severity: WARNING

Explanation: By adding the page number to the original input file name to compose the figure EPS output file name, the name exceeded the maximum allowed RMS filename length. DOCTOR therefore truncated the name by removing the last few characters of the input file name to make it fit.

User Action: None. You may wish to rename the file name manually after DOCTOR has completed.

Digital Internal Use Only

UNKWNPAG, Unknown page type: 'string'

Severity: FATAL

Explanation: The utility can only handle arabic or roman numbered pages such as 1, 2, 3 or 1-1 or Glossary-6 or PART1-IX or III. The input file seems to contain page numbering schemes other than these two. Thereby this utility cannot interpret those pages correctly.

User Action: Whatever you wanted to do must be done manually.

A.8 SADDLE Messages

2UP, Preparing file for 2UP printing in the next file version...

Severity: INFORMATIONAL

Explanation: The file that was sorted for saddle-stitching is once more modified to add PostScript code to allow it to be printed as 2-up without any reduction of the page size.

User Action: None

2UPNOTSUP, The /2UP feature is only supported for VAX DOCUMENT V2.1

Severity: FATAL

Explanation: The PostScript code added to allow 2-up printing is specific for the code generated by VAX DOCUMENT V2.1. Until a more generic code is developed, no other PostScript files can be processed with /2UP. The file created can be used for reduced saddle stitch printing, using the /PARAMETER=(SIDE=TUMBLE,NUMBER=2) feature of the printservers.

User Action: Use the /SADDLE by itself with /PARAMETER=(SIDE=TUMBLE,NUMBER=2)

CONFORM, File assumed to conform to 'string'

Severity: INFORMATIONAL

Explanation: Input PostScript file seems to conform to the Adobe Inc. (minimal) conformant coding standards.

User Action: None.

CREATED, Created 'string'

Severity: SUCCESS

Explanation: Indicates the successful completion of the utility and the creation of the specified output file.

User Action: None.

Digital Internal Use Only

CREATOR, Input file was created by 'string'

Severity: INFORMATIONAL

Explanation: This message indicates the origin of the PostScript file that is currently being processed. Depending on this origin, some features of DOCTOR/PS may not be used. All VAX DOCUMENT produced files will be handled properly.

User action: None

DEFGATHER, No /GATHER qualifier specified.
Using /GATHER='unsigned decimal number' as default

Severity: INFORMATIONAL

Explanation: When no /GATHER is specified, the entire book is re-ordered into a single gathering. This allows you to either fold the paper stack into a brochure or to cut the paper stack in the middle for a paperback glueing.

User Action: None or specify /GATHER with a different value

INVORDINAL, Input file violates Adobe standard: ordinal starts at 'signed decimal number'

Severity: FATAL

Explanation: The comments of the first page inside the PostScript file should indicate the folio number as well as the sequence number of the page within the file. The first page is assumed to have sequence number 1. For some reason, the program that output the PostScript file did not start counting from 1 upwards, but from the indicated number.

User Action: None - The DOCTOR will rectify this in the output file it produces. You may wish to file a Problem Report to the producers of the software package that made your PostScript file, stating they violate the Adobe programming standards.

NOFOUR, The /GATHER='unsigned decimal number' is not divisible by 4

Severity: FATAL

Explanation: Saddle stapling can only occur when multiples of 4 pages are specified (that can be printed on a single sheet - front and back side). The specified number is not divisible by 4.

User Action: Specify a correct number that can be divided by 4.

Digital Internal Use Only

NOPAGE, No valid %%Page: comment lines. File is not Adobe Structured Coding conformant

Severity: FATAL

Explanation: The specified input PostScript file does not seem to adhere

to the Adobe Inc. minimal conformant coding standard as it lacks the %%Page: page separation comment lines. Hence the utility cannot correctly interpret the input file (even if it is correct PostScript code).

User Action: Attempt to re-create the PostScript file enforcing the text processor to output conformant Adobe Structured Coding Standard code. For MS-Windows applications, this may imply to enter the PRINT SETUP... menu and find (under various layers of additional Setup... or Options... and Advanced Options... menus) the check box for Adobe Structured Coding Standards. Check this and export the PostScript file again.

NOTCONFORM, File is not (minimally) conforming to Adobe PostScript standards. No output created.

Severity: FATAL

Explanation: The specified input PostScript file does not seem to adhere to the Adobe Inc. minimal conformant coding standard. Hence the utility cannot correctly interpret the input file (even if it is correct PostScript code).

User Action: Whatever you wanted to do must be done manually.

ORDERED, PostScript file already is in special order. Output may be useless

Severity: WARNING

Explanation: DOCTOR discovered the PostScript comment `%%PageOrder: Special` inside the PostScript file. This indicates that the pages in the file have already been ordered in some special way. Although DOCTOR will proceed, the final output may have re-arranged the pages in such a way that the order is not useful.

User Action: None. Either do not try to saddle stitch the file or look at the printer output if the result is acceptable.

PAGES, Document contains 'unsigned decimal number' pages

Severity: INFORMATIONAL

Explanation: DOCTOR found the indicated number of `%%Page:` entries that were valid page entries.

User Action: None. It may help in determining the size of the gathering you want to specify.

Digital Internal Use Only

PASS, Starting input file scan Pass 'unsigned decimal number'

Severity: INFORMATIONAL

Explanation: Starts reading the input PostScript file in Pass 1 to collect all page related data before the pages are randomly accessed in Pass 2 to allow extraction or re-ordering them in the final output.

User Action: None.

SAMEFILESPEC, Output file is next higher version of input file.
Purging will lose original file.

Severity: WARNING

Explanation: You did not specify an explicit output file, hence the utility defaults to the input file specification. However, this may cause the lost of the original input file is an accidental purge command is issued.

User Action: Rename the latest version into another file name or copy the file to another location to avoid losing the original copy.

UNSUPVER, File produced by 'string' is not supported.
See Message Appendix in DOCTOR manual for manual correction

Severity: FATAL

Explanation: The PostScript file is produced by a product that outputs PostScript code that cannot be handled by DOCTOR/PS/SADDLE.

User Action: For some products this is because there is a bug in their PS code.

The following products allow manual correction and then you can resubmit the corrected file for correct processing:

DECwrite V1.1
Edit the PS file and at the final occurrence of the `%%Trailer` line interchange the following two lines:

```
$D restore
%%Trailer
```

into

```
%%Trailer
$D restore
```

Now the restore operation remains with the trailer at the end of the saddle stitched file rather than moving with the original last page, causing "offending command is a" errors.

Digital Internal Use Only

A.9 XREF Messages

CREATED, Created 'string'

Severity: INFORMATIONAL

Explanation: XREF has succesfully created the indicated output file.

User Action: None

NOBOOKTITLE, No /BOOKTITLE specified with /SYMBOL_FILE

Severity: WARNING

Explanation: When a file is created with <DEFINE_SYMBOL> tags, to allow cross referencing between books, it is almost imperative that you also specify the title of the book, so that the cross reference can contain both title and section of the book involved.

User Action: Add /BOOKTITLE with proper title of the book

NOPREFIX, No prefix specified. This may lead to conflicting references

Severity: WARNING

Explanation: The user did not specify the /PREFIX qualifier. Hence all symbols from the XREF file to be extracted as symbol definitions may turn out to be identical to symbols defined elsewhere.

User Action: It is good practise to make all symbols of a particular book unique by prefixing them with a short identification. This can be an acronym of the book, an author number, etc. Keep in mind that the total length of prefix + symbol name must not exceed the maximum length allowed to symbols, being 31 characters (A-Z, 0-9 and underscore). The prefix should not start with an underscore but may end with one.

NOSORTSPEC, Qualifier 'string' is invalid without /SORT. Will be ignored

Severity: WARNING

Explanation: It is only possible to produce sorted alphabetic or numerically ordered lists.

User Action: Either add /SORT to the command line (to activate the currently specified qualifier, or remove the specified qualifier.

SYMTRUNC, Symbol 'string' exceeds maximum allowed symbol length. Truncated

Severity: WARNING

Explanation: Due to the prefixing of the specified string with /PREFIX the length of the entire symbol exceeded the maximum length allowed by VAX DOCUMENT. It is truncated to maximum allowable length by removing the final character(s).

User Action: None. Alternatively, a shorter prefix can be choosen or in the original document from which the XREF file originates, one should shorten the symbolic names

Digital Internal Use Only

to avoid the problem. This requires processing of that file by VAX DOCUMENT again after editing.

WHATTODO, You must specify at least one of /SYMBOL_FILE, /LIST or /SORT

Severity: FATAL

Explanation: DOCTOR/XREF can only perform one or several of its three possible operations if one of the three qualifiers indicated is specified. Without either of them, nothing is done.

User Action: Specify one, two or all three specified qualifiers.

Digital Internal Use Only

Index

A

ABSENTPAGESCHEME • A–8
ATLAST • A–4

B

%%BeginDocument: comment • 6–8, 6–14
blank page • 6–2
  user specified • 6–4
Bookreader
  Adding references • 5–7
  Adding symbols • 5–7
  correcting files for - • 5–1
  figure file for - • 5–7
  with DOCUMENT V1.2B • 5–7
  with DOCUMENT V2 and higher • 5–7
Border • 6–6
Brochure, producing reduced sized • 6–10
Bug Reports • vii

C

CONFORM • A–8, A–13
CREATED • A–1, A–2, A–4, A–6, A–8, A–13, A–17
CREATOR • A–9, A–14
CRELIST • A–6
cross reference • 9–3

D

DOCUMENT/GRAPHICS - See RAGS
DEC/CMS • 7–3
  class and DEC/MMS rebuild • 7–5
DEC/MMS • 7–1
  Use of MMS and DOCUMENT • 7–9
DECwindows/Motif • 7–11
DEFGATHER • A–14
<DEFINE_SYMBOL> tag • 9–2
DOCTOR
  components • 2–1
  installation • vii
  invocation • vii
  using DEC/CMS • vii
DOCTOR/ONLINE • 5–1
document structure • 7–1
  build MMS description file • 7–3
DUPLICATE • A–3

E

%%EndComments comment • 6–14
%%EndDocument: comment • 6–8
%%EndDocument comment • 6–14
%%Endprolog comment • 6–14

F

.FACILITY • 4–5
figure
  extracting - from PostScript file • 6–8, 6–9
  removing - from PostScript file • 6–9
FNF • A–2, A–5, A–6
FSE format • 5–7

G

Glossary
  International character set sort • 3–3
  sorting • 3–1

Digital Internal Use Only

Glossary (cont'd)
sort order • 3–2
GTERMOUTBOUND • A–3
<GTERM> tag • 3–1

H

HELP/MESSAGE utility • 4–1, 9–12
HUMANS • A–5

I

IDENT • A–1
IGNORED • A–5
Include files • 7–1
Index entries • 7–11
correction of • 7–12
INPUTFNF • A–7
INSERTED • A–9
INSPECT • A–7
INVALCOUNT • A–9
INVORDINAL • A–9, A–14
INVPAGCOM • A–9

L

LaTeX • 8–2
LPS20 printer
Use for double sided printing • 6–12
LPS40 printer
Use for printing • 6–12

M

MARFIN • 7–3
Message file, VMS • 4–1
elements of a - • 4–2
sorting messages • 4–2
<MESSAGE_SECTION> tag • 4–2
MMSTOO • A–5
MODIFY • A–7
<MSG> tag • 4–2
<MSG_TEXT> tag • 4–2
MUMBLE • A–5

N

NEGPAGNO • A–10
NEXTGLOSSARY • A–3
NOALPHA • A–3
NOBOOKTITLE • A–17
NOCHANGE • A–10
NOCMSSUP • A–5
NOENDGLOSSARY • A–3
NOFOUR • A–14
NOGLOSSARY • A–4
NOGTERMS • A–4
NOPAGE • A–10, A–15
NOPREFIX • A–17
NOPROLOG • A–10
NOSORTSPEC • A–17
NOTCONFORM • A–11, A–15
NOTFOUND • A–6
NOTIMPL • A–1
NOTVAXDOC • A–11

O

OHMYDIODES • A–6
OHNO • A–6
ORDERED • A–15

P

%%Page: comment • 6–4, 6–14
PAGES • A–15

Digital Internal Use Only

PASS • A–11, A–16
PASS1 • A–1, A–7
PASS2 • A–1, A–7
Perfect bind • 6–11
PostScript
  Add blank page • 6–2
  %%BeginDocument: comment • 6–8
  conformant PostScript skeleton • 6–14
  drawing page border • 6–6
  encapsulated figures • 6–8
  %%EndDocument: comment • 6–8
  minimal conformant • 2–1, 6–13
  %%Page: comment • 6–4, 6–7
  page range description • 6–7
  prolog replacement • 6–4
  structured comments • 6–4
  structured comments V2 • 6–3, 6–14
  structured comments V3 • 6–3, 6–14
  user specified blank page • 6–4
PostScript file
  prolog replacement • 6–4
PostScript files • 6–1
PS2TEXT utility • 6–9
%!PS-Adobe comment • 6–14

Q

QAR reports • vii

R

RAGS • 5–7, 6–14
  use with MMS • 7–10
RANGESKIP • A–11
reference • 9–1
reference tag • 5–1
<REFERENCE> tag • 5–7
REMOVE • A–8
Rules for MMS and DOCUMENT • 7–9
Runoff • 8–2

S

Saddle stich • 6–10, 6–11, 9–19
SAMEFILESPEC • A–4, A–12, A–16
SCAN • A–2
SDML file
  adding symbols for Bookreader • 5–1
SENDQAR • A–2
SET DISPLAY command • 7–11
.SEVERITY • 4–5
SGML • 8–2
SHORTRANGE • A–12
SKIPFIGURE • A–12
STARTRANGE • A–12
.SUFFIXES for MMS and DOCUMENT • 7–9
symbol • 5–1, 5–7, 9–1
  definition of all symbols • 9–2
  list of all symbols • 9–1
SYMTRUNC • A–17

T

Tags
  counting • 8–1
TRUNCATED • A–12

U

UNKWNPAG • A–13
UNSUPVER • A–16
2UP • A–13
2UPNOTSUP • A–13
UTOX • 5–7

V

VMS
  Alpha • 4–1, 9–12

Digital Internal Use Only

VMS (cont'd)
Open • 4–1, 9–12

W

WHATTODO • A–18

X

XREF file • 9–1
<X> tag • 7–1, 7–11

Y

<Y> tag • 7–1, 7–11

Z

ZzTeX • 8–2