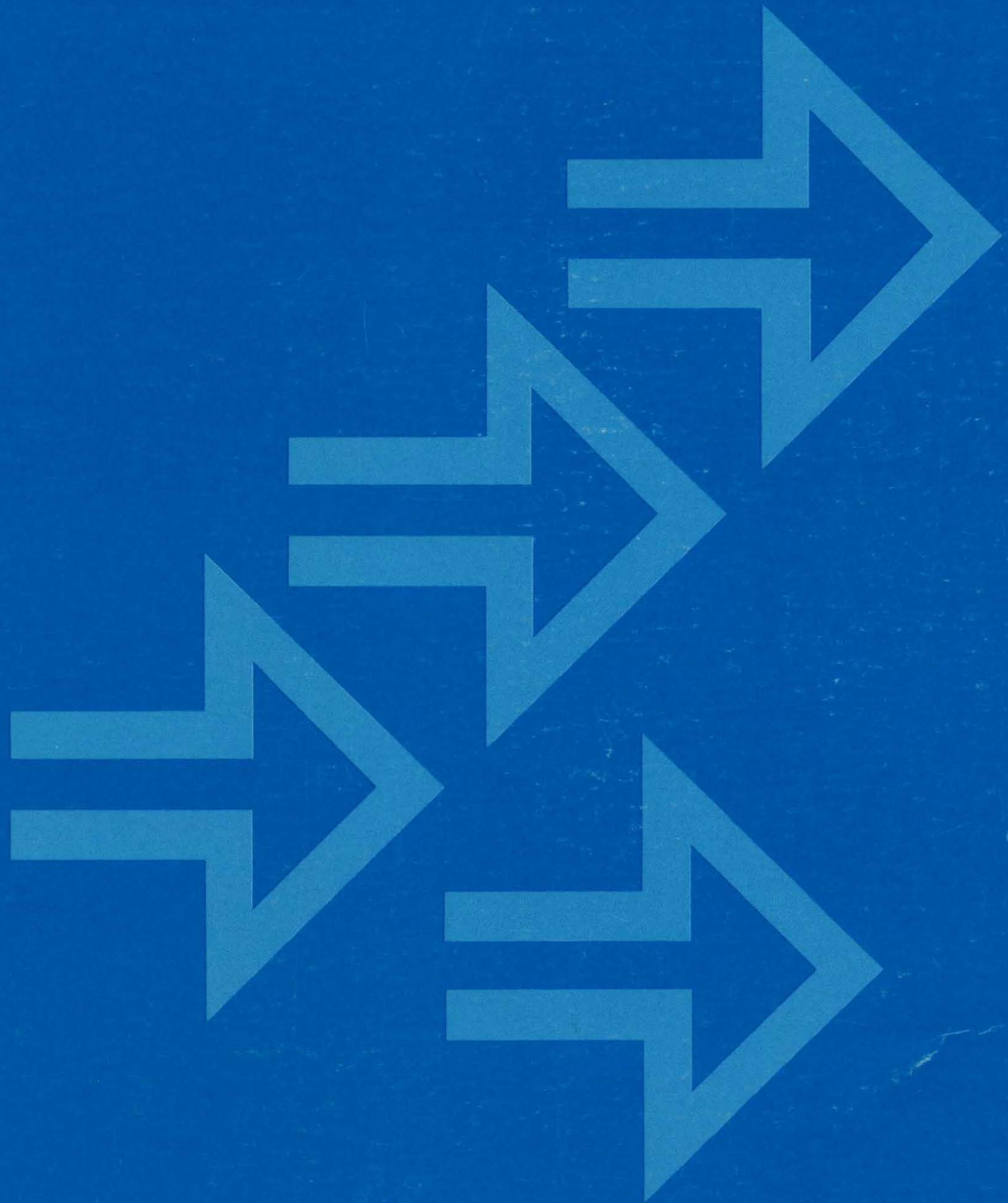




**Document Composition Facility  
Generalized Markup Language  
Starter Set User's Guide  
Release 3.2**

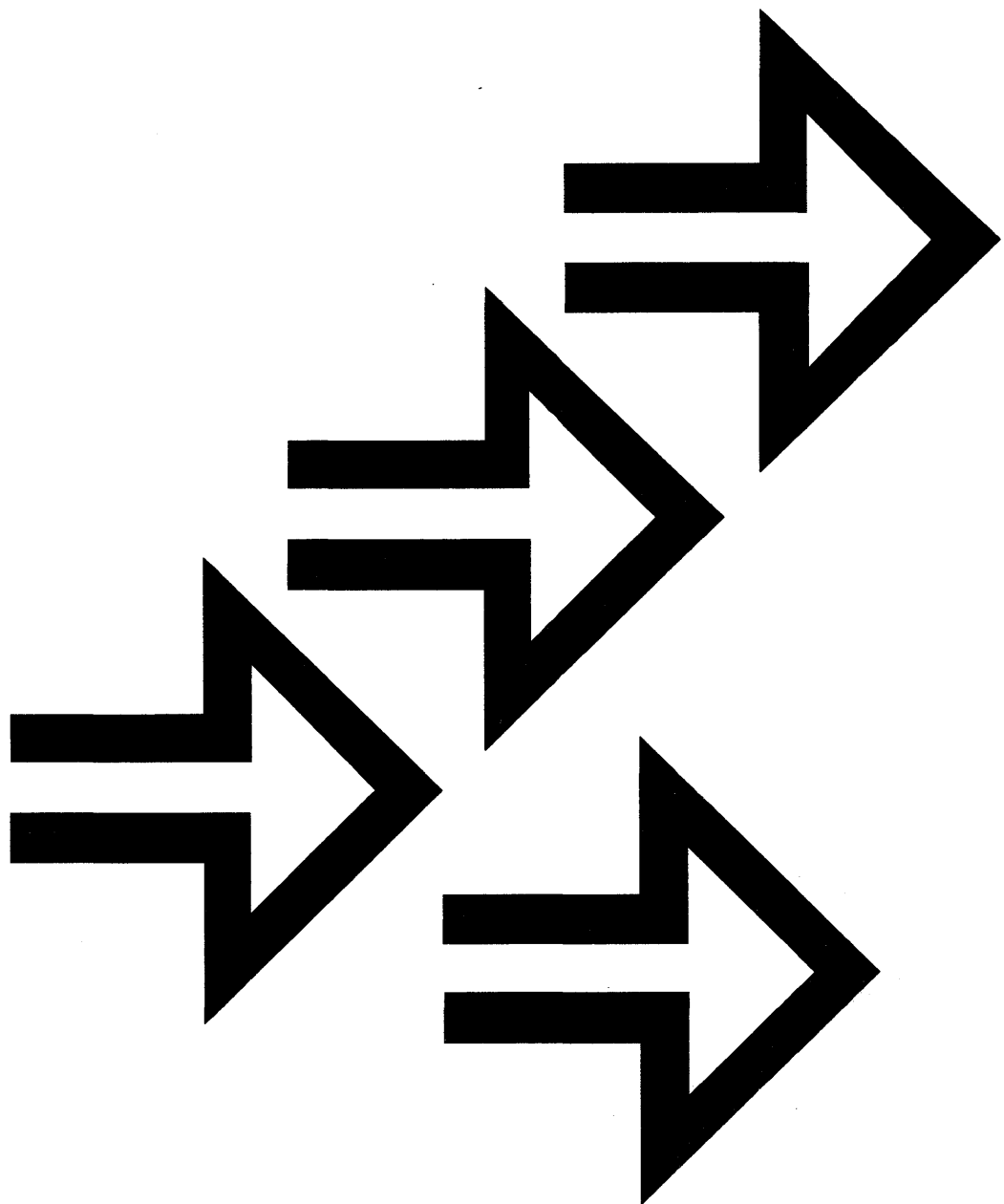
SH20-9186-06





**Document Composition Facility  
Generalized Markup Language  
Starter Set User's Guide  
Release 3.2**

SH20-9186-06



This book was formatted with IBM Publishing Systems BookMaster licensed program (Program Number 5688-015). When you use the GML starter set tags, your formatted output may differ from the examples shown in this book because of differences in column line length, fonts, line spacing, device types, indentation, and other formatting differences between GML and BookMaster.

The following terms, denoted by a double asterisk (\*\*), used in this publication, are trademarks of other companies as follows:

PostScript	Adobe Systems Incorporated
Monotype	Monotype Corporation, plc.
Monotype Times New Roman	Monotype Corporation, plc.

### **Seventh Edition (October 1989)**

| This is a repackaged edition of *Document Composition Facility: Generalized Markup Language Starter Set User's Guide*, SH20-9186-06.

This edition contains information from and makes obsolete the *Document Composition Facility: Generalized Markup Language Starter Set User's Guide*, SH20-9186-05. Changes are indicated by a vertical line to the left of the change.

This edition applies to Release 3.2 of the IBM Document Composition Facility Licensed Program, 5748-XX9, and to any subsequent releases of that program until otherwise indicated in new editions or technical newsletters.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not available in your country. Such references or information must not be construed to mean that IBM intends to announce such products in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

| Comments regarding this publication may be sent to this address: IBM Corporation, Box 1900,  
| Department 588, Boulder, Colorado, U.S.A. 80301-9191. IBM may use or distribute any of the  
information you supply in any way it believes appropriate without incurring any obligation whatsoever.  
You may, of course, continue to use the information you supply.

**DCF: GENERALIZED MARKUP  
LANGUAGE STARTER SET USER'S  
GUIDE (SAMPLE TITLE PAGE)**

*Document Number SH20-9186-06*

*October 3rd, 1989*

Author's Name

Author's Address



---

## Preface

The IBM Document Composition Facility (DCF) is a text-processing program; its main component is the text formatter, called SCRIPT/VS.

One of the major features of SCRIPT/VS is the ability to develop and use the Generalized Markup Language (GML). With DCF, IBM supplies, a *starter set* of GML, to give users a starting point in using GML and developing their own GML to meet their specific needs.

This book is intended for new users of DCF and particularly for new users of GML. This book teaches you how to use GML in general and how to use the starter set in particular. It also prepares you to use any specific GML that your organization might develop.

**Note: Central Programming Service support and maintenance are available for GML applications shipped with DCF. Support and maintenance, however, are *not* available if these applications have been modified in any way. If you do make modifications, it is recommended that you also maintain an *unmodified* set for diagnostic purposes.**

References to the IBM 3800 Printing Subsystem refer to the IBM 3800 Model 1 and the IBM 3800 Model 3 and Model 6 in all-points-addressability mode unless otherwise noted.

References to page printers apply to the 4250 Printer, the 3800 Printing Subsystem Models 3 and 6, the 3820 Page Printer, printers that are equivalent to the 3820 Page Printer at the data stream level, and PostScript\*\* devices configured to accept 8-bit ASCII, unless otherwise explicitly stated.

**Note:** When you use the GML starter set tags, your formatted output may differ from the examples shown in this book because of differences in column line length, fonts, device types, and other minor formatting differences between GML and BookMaster.

This book has the following major sections:

- “Part One: Getting Started” is a brief introduction to the GML starter set and describes what you need to know before you can start using GML.
- “Part Two: Tag Guide” describes the use of the GML starter set tags to mark up a document.
- “Part Three: Document Formatting” describes what you need to know in addition to the tags themselves to take full advantage of the starter set and SCRIPT/VS.
- “Part Four: Tag Reference” contains a reference summary for the starter set tags.
- “Part Five: Appendixes” contains a sample GML document showing the marked-up source file and the formatted output for that file. This part also contains the solutions to the exercises.

## Publication Library Guide for the Document Composition Facility

The following table lists the DCF publications by number as they relate to user tasks. "DCF Publications" on page v lists the titles and the order numbers that correspond to the numbers listed in the table.

Number	User Tasks	Typical Audience	Brief Description
(1) (2) (3) (20)	Planning and introducing DCF/DLF	Users, system planners	Provide a general overview of text processing, library facility, and available books.
(3) (4) (5) (12) (16) (20)	Formatting documents (using the GML starter set)	Novice to experienced end users	Provide an introduction to the Generalized Markup Language (GML) starter set and describes the GML starter set tags and SCRIPT/VS messages.
(6)	Creating bar codes with DCF/GML	Experienced end users	Provides information about using GML to create bar codes.
(9) (10) (11) (12) (17) (18) (19) (22)	Formatting documents (using SCRIPT/VS control words)	Knowledgeable to experienced end users	Describe the function and use of all SCRIPT/VS control words, macros, diagnostic aids, and the formatting features and messages.
(14) (15)	Converting RFTDCA for SCRIPT/VS formatting	Novice to experienced RFTDCA users	Describe the function and use of the optional Office Document Feature, including diagnostic aids and messages.
(4) (5) (7) (9) (10) (11) (19) (22)	Modifying GML starter set <sup>1</sup>	Document administrator and text programmer <sup>2</sup>	Contain material about GML starter set tags, SCRIPT/VS control words, and GML starter set modifications.
(4) (5) (7) (8) (9) (10) (11) (16) (19) (22)	Creating GML application processing functions	Document administrator and text programmer <sup>2</sup>	Provide information about designing your own GML and about GML concepts, GML starter set tags, SCRIPT/VS control words, and usage guidelines.
(10) (11) (12) (13) (19) (21) (22)	Installing, modifying, and maintaining DCF	System programmer	Give information on error isolation, program tailoring, and use of SCRIPT/VS.
(23)	Creating mathematical formulas with SMFF	Experienced end users	Describes the function and use of the SCRIPT Mathematical Formula Formatter (SMFF), including .EQ control word and messages.
(24)	Creating memos, transparencies, and schedules with GML applications	Novice to experienced end users	Describes the use of the memo, transparencies, and schedule applications, including messages.

<sup>1</sup> Central Programming Service support and maintenance are provided **only** on the unmodified GML applications shipped with DCF. If you modify any of these GML applications shipped with DCF, it is recommended that you also maintain an **unmodified** copy for diagnostic purposes.

<sup>2</sup> The document administrator is responsible for defining markup conventions and procedures for an organization. The text programmer implements application processing functions (APFs) that provide the processing specified by the document administrator.

## DCF Publications

Number	Titles and Order Numbers
(1)	<i>Document Composition Facility and Document Library Facility Executive Overview and Product Summary</i> , GX20-2332.
(2)	<i>Document Composition Facility and Document Library Facility General Information</i> , GH20-9158.
(3)	<i>Document Composition Facility: Introduction to Generalized Markup Language</i> , G544-3192.
(4)	<i>Document Composition Facility: Generalized Markup Language Starter Set User's Guide</i> , SH20-9186.
(5)	<i>Document Composition Facility: Generalized Markup Language Starter Set Reference</i> , SH20-9187.
(6)	<i>Document Composition Facility: Bar Code User's Guide</i> , S544-3115.
(7)	<i>Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide</i> , SH35-0050.
(8)	<i>Document Composition Facility: Generalized Markup Language Starter Set Concepts and Design Guide</i> , SH20-9188.
(9)	<i>Document Composition Facility: SCRIPT/VS User's Guide</i> , S544-3191.
(10)	<i>Document Composition Facility: SCRIPT/VS Text Programmer's Guide</i> , SH35-0069.
(11)	<i>Document Composition Facility: SCRIPT/VS Language Reference</i> , SH35-0070.
(12)	<i>Document Composition Facility SCRIPT/VS Messages</i> , SH35-0048.
(13)	<i>Document Composition Facility: Diagnosis Guide and Reference</i> , LV32-0523.
(14)	<i>Document Composition Facility: Office Document Feature User's Guide</i> , G544-3129.
(15)	<i>Document Composition Facility: Office Document Feature Reference</i> , S544-3130.
(16)	<i>Using the Document Composition Facility</i> , SR21-0515 (Training Course 32291).
(17)	<i>Using DCF with the 4250 Printer</i> , SR20-8486 (Training Course 32908).
(18)	<i>Using DCF with Page Printers</i> , SR21-1211 (Training Course 32243).
(19)	<i>Document Composition Facility—Release 3 (SCRIPT/VS) for Document Administrators and Text Programmers</i> , SR20-7525 (Training Course).
(20)	<i>Document Composition Facility (SCRIPT/VS) Student Text</i> , SC20-1894 (Training Course).
(21)	DCF Program Directory.
(22)	<i>Document Composition Facility: TSO Enhancements Update Guide</i> , G544-3345.
(23)	<i>Document Composition Facility: SCRIPT Mathematical Formula Formatter User's Guide</i> , S544-3306.
(24)	<i>Document Composition Facility: Generalized Markup Language (GML) Applications User's Guide</i> , G544-3305.

**Note:** The following reference booklets and a three-ring binder are also available:

*Document Composition Facility: GML Starter Set Quick Reference*, SX26-3723.

*Document Composition Facility: SCRIPT/VS Text Programmer's Quick Reference*, SX26-3719.

*Document Composition Facility Post-Processor Examples*, S544-3484.

*Document Composition Facility SCRIPT/VS*, SH35-0086 (binder).



---

## Related Publications

You should use the following publications to evaluate the use of DCF in different operating environments:

- *IBM Virtual Machine/System Product: Introduction*, GC19-6200  
This publication contains an introduction to Conversational Monitor System (CMS), which is one of the interactive systems that SCRIPT/VS operates with.
- *IBM Virtual Machine/System Product: Terminal User's Guide*, GC19-6206  
This publication describes the various types of terminals supported by VM/SP for those who plan to use VM/SP in their operations.
- *OS/VS2 TSO Terminal User's Guide*, GC28-0645  
This publication gives detailed user information about OS/VS2 TSO (Time-Sharing Option), which is one of the interactive systems that SCRIPT/VS operates with. It describes the TSO EDIT facility and related facilities for text entry, text editing, and data set management.
- *Advanced Text Management System-III (ATMS-III) General Information Manual*, GH20-2404  
This publication provides an overview of the ATMS-III licensed program and the functions it can perform.
- *A Guide to IBM's Advanced Function Printing*, S544-3095  
This publication describes the use of a licensed program (PSF, DCF, GML, OGL, GDDM, and PMF) and the use of a subset of a licensed program in conjunction with the IBM Advanced Function Printing (AFP) printers, including the IBM 3800 Printing Subsystem Models 3 and 6 and the IBM 3820 Page Printer.
- *Advanced Function Printing Software: General Information*, G544-3415  
This publication defines Advanced Function Printing (AFP), describes the features and functions of the AFP licensed programs, and shows how the programs work together. It is intended for the people in your organization who will plan for, install, use, and maintain IBM's AFP software products. It also contains a list of the AFP publications.
- *Composed Document Printing Facility: General Information*, GC33-6133.  
This publication provides general information needed for product evaluation and preliminary planning for the programming support for the IBM 4250 Printer.

---

## Related PostScript Publications

You can use the following publications to learn more about the PostScript language:

- *PostScript Language Tutorial and Cookbook*, Addison-Wesley Publishing Company, Copyright 1986 by Adobe Systems Incorporated
- *PostScript Language Reference Manual*, Addison-Wesley Publishing Company, Copyright 1986 by Adobe Systems Incorporated.

---

## Related Products

The following products are related to Generalized Markup Language:

- **MARKUP:** A PC-based editor that helps you create GML documents. Refer to *MARKUP User's Guide and Tutorial*, which you can order with the MARKUP product, 6476161. For information about ordering MARKUP, contact your local IBM Branch Office.
- **Publishing Systems BookMaster:** A host-based document-creation application that runs under control of DCF and is designed for high-volume in-house publishing applications. This application contains a superset of the GML Starter Set. For information about ordering BookMaster (Program Number 5688-015), contact your local IBM Branch Office.



---

# Contents

---

<b>Part One: Getting Started</b> .....	1
<b>Chapter 1. Introduction</b> .....	3
Phrases Used in This Book .....	3
<b>Chapter 2. What Is GML?</b> .....	5
GML Document Types .....	5
The Starter Set Profile (DSMPROF3) .....	6
Creating and Formatting Documents .....	6
Your Computer System .....	7
<hr/>	
<b>Part Two: Tag Guide</b> .....	9
<b>Chapter 3. Paragraphs and Headings</b> .....	11
Paragraphs .....	11
Headings .....	11
More About Heads .....	14
Exercise: Paragraphs and Headings .....	15
<b>Chapter 4. Lists</b> .....	17
Simple, Unordered, and Ordered Lists .....	17
Definition Lists .....	21
Glossary Lists .....	24
Nested Lists .....	25
List Parts .....	27
Implied Paragraphs .....	28
Exercise: Lists .....	29
<b>Chapter 5. Examples and Figures</b> .....	31
Examples .....	31
Paragraph Continuation .....	32
Figures .....	32
Page Segments .....	38
Reserving Space for a Page Segment .....	38
PostScript Images .....	39
Space Units .....	39
Exercise: Examples and Figures .....	43
<b>Chapter 6. Highlighting, Citing, Noting, and Quoting</b> .....	45
Highlighted Phrases .....	45
Highlighting for Various Devices .....	46
Title Citations .....	46
Notes .....	47
Quotations—the Long and Short of It .....	47
Inline Quotations .....	48
Excerpts (Long Quotations) .....	48
Exercise: Highlighting, Citing, Noting, and Quoting .....	50
<b>Chapter 7. Tables</b> .....	51

Defining the Table	51
Building the Table	55
Starting a Row	56
Starting a Cell	57
Using a Table Header	57
Using a Table Footer	58
Using a Table Caption and a Table Description	58
Using a Table Note	59
Sample Tables	60
Creating a Three-Row Table with Text in the Cells	60
Creating a Table with Two Different Row Definitions	61
Changing the Highlight Level of the Cells	62
Using the ALIGN, VALIGN, and CONCAT Attributes	63
Using the ROTATE and MINDEPTH Attributes	64
Creating a Table Note, Caption, and Description	65
Using the ARRANGE and CWIDTHS Attributes	66
Step 1. Sketch the Row	66
Step 2. Form a Grid	66
Step 3. Determine the Widths of the Cells	67
Step 4. Number Each Box	67
Step 5. List the Numbers	67
Step 6. Enter the Row Definition	67
Using the ARRANGE Attribute	68
Creating a Table Header	68
Exercises: Tables	70
First Exercise	70
Second Exercise	71
<b>Chapter 8. Cross-References and Footnotes</b>	73
To Headings	73
To List Items	74
To Figures	75
To Tables	76
To Footnotes	77
Unmatched References	78
Listing of Cross-References	79
Exercise: Cross-References and Footnotes	80
<b>Chapter 9. Overall Document Structure</b>	83
The Front Matter	84
Title Page	84
Title	85
Document Number	85
Date	85
Author	85
Address	86
Abstract	86
Preface	86
Table of Contents, List of Figures, and List of Tables	87
The Body	87
The Appendixes	88
The Back Matter	88
Document Structure Summary	89
Combining Input Files	89
Exercise: Overall Document Structure	91

<b>Chapter 10. Indexing</b> .....	93
Creating Index Entries .....	93
Page Ranges — and Other Things You Can Do with Page References .....	97
See and See Also .....	98
Sorting Index Entries .....	99
Notes on the Normal Sort .....	101
Where to Put Index Entries .....	101
Getting the Index .....	102
Summarizing the Indexing Tags .....	102
Readable Source Files .....	104
Exercise: Indexing .....	105
<b>Chapter 11. Process-Specific Controls</b> .....	107
Physical Devices .....	108
Logical Devices .....	108
Conditions .....	108
<hr/>	
<b>Part Three: Document Formatting</b> .....	111
<b>Chapter 12. Symbols</b> .....	113
Symbols Provided by the Starter Set .....	113
Your Own Symbols .....	116
<b>Chapter 13. Formatting Instructions</b> .....	119
Instructions to the Starter Set (SYSVAR) .....	120
Instructions to SCRIPT/VS .....	123
Identifying the Document to Be Processed .....	124
SCRIPT Command Options .....	124
Processing Options: .....	125
Logical Device and Output Destination .....	130
Error Handling .....	132
Font Requirements for Page Printers .....	133
Font Requirements for PostScript Devices .....	133
Overriding the Default Font Using the CHARS Option .....	134
<b>Chapter 14. Error Handling</b> .....	135
Starter Set and SCRIPT/VS Messages .....	135
Common Problems — Symptoms and Solutions .....	136
Symptom: Unexpected Formatting Results .....	136
Symptom: Text Disappears .....	136
Symptom: Text Doesn't Format .....	137
Symptom: Column Is Narrow .....	137
Symptom: The Colon after a Level 5 or 6 Heading Is Missing .....	137
Symptom: Page Number References Are Wrong .....	138
Symptom: Figures Are Out of Order .....	138
Symptom: Blanks Disappear .....	139
Symptom: Semicolon Won't Print .....	139
Symptom: Front Matter Lists Are Empty .....	139
Symptom: Index Is Empty .....	139
Symptom: Blank Inserted at GML Tag .....	139
<b>Chapter 15. Cross-Reference Listing</b> .....	141
The Imbed Trace .....	141

---

<b>Part Four: Tag Reference</b> .....	143
<b>Chapter 16. General Guidelines for Markup Entry</b> .....	145
<b>Chapter 17. Alphabetic Summary of Tags</b> .....	147
<b>Chapter 18. Summary by Document Elements</b> .....	151

---

<b>Part Five: Appendixes</b> .....	155
<b>Appendix A. A Sample GML Document</b> .....	157
<b>Appendix B. Solutions for Exercises</b> .....	169
Markup for Paragraphs and Headings .....	169
Markup for Lists .....	170
Markup for Examples and Figures .....	171
Markup for Highlighting, Citing, Noting, and Quoting .....	172
Markup for Tables .....	173
First Table Exercise .....	173
Second Table Exercise .....	173
Markup for Cross-References and Footnotes .....	174
<b>Glossary</b> .....	175
<b>Index</b> .....	185

---

# Figures

1. Terminology Used in a Discussion of Indexing	94
2. Example of SYSVAR Parameters	123
3. Example of a Title Page	161
4. Example of a Table of Contents	162
5. Example of a Figures List	163
6. Example of a Tables List	164
7. Example of Text with Headings	165
8. Example of Text with a Figure	166
9. Example of a Table	167
10. Example of an Index	168





# **Tables**

1.	Example of a Table Note, Caption, and Description . . . . .	65
2.	Example Table Showing the ARRANGE Attribute. . . . .	68
3.	Example Table Showing a Table Header . . . . .	69
4.	Example Table Showing U. S. Automobile Production . . . . .	70
5.	A Captioned Table, to Demonstrate Cross-Referencing . . . . .	76
6.	Table Cross Reference Exercise . . . . .	80
7.	SCRIPT/VS Logical Line Devices . . . . .	130
8.	SCRIPT/VS Logical Page Devices . . . . .	131
9.	SCRIPT/VS Logical PostScript Devices . . . . .	132
10.	Alphabetic Summary of Starter Set Tags . . . . .	148
11.	Major Document Elements . . . . .	151
12.	Indexing . . . . .	151
13.	Headings . . . . .	152
14.	Basic Text . . . . .	152
15.	Examples and Figures . . . . .	152
16.	Tables . . . . .	153
17.	Footnotes . . . . .	153
18.	Process Specific Controls . . . . .	154
19.	Lists . . . . .	154



---

# Summary of Amendments

DCF, Release 3.2<sup>3</sup>, includes the following new features:

- **PostScript Support:** Adds support for the PostScript page description language so that you can use SCRIPT/VS to format DCF output for a PostScript device configured to accept 8-bit ASCII and include PostScript images in DCF documents.
- **Generic Page Printer Support:** Allows generic specification of page printer devices.
- **Office Document Feature:** Can convert an RFTDCA<sup>4</sup> file into an RFT/GML file for formatting by SCRIPT/VS when this optional feature is installed with Release 3.2. This feature can also convert an RFT/GML file back into an RFTDCA file. The RFTDCA files can be created on an office system that produces the RFTDCA data stream, such as the DisplayWrite 4 or DisplayWrite/370.

Release 3.2 includes the following functions:

- **VM/XA SP Support:** Adds VM/XA support in 370 compatibility mode or 370-XA toleration mode that was made available as a program temporary fix (PTF) to Release 3.1.
- **Mapped Page Segments:** Allows page segments to be downloaded to printer storage prior to the start of the page. When the page segment is used, it is retrieved from printer storage rather than downloaded at that time. This is a performance savings when a page segment is used more than once in a document, because the page segment is downloaded only once.

Release 3.2 includes the following changes to this book:

- Some chapter titles and other headings were changed slightly.
- The chapter "Highlighting, Citing, Noting, and Quoting" was moved to follow the chapter "Examples and Figures."
- The former chapters 13 and 14 were combined into "Chapter 14. Error Handling."
- "Appendix A. Creating and Editing a File" was removed. A briefer version of this information can be found in the *Document Composition Facility: Introduction to Generalized Markup Language*.
- "Appendix D. Differences from Earlier Releases" was replaced by this Summary of Amendments.

**Note:** This version of the book was produced, primarily, in response to an overwhelming number of comments from our readers who stated that they preferred the 8.5-inch by 11-inch, 5-hole punch format for DCF publications. Some minor technical and editorial changes were also made in this book.

---

<sup>3</sup> Release 3.2 of DCF may be used with Release 3 of the Document Library Facility.

<sup>4</sup> Revisable-Form-Text Document Content Architecture.

**Central Programming Service support and maintenance are available for the GML applications shipped with DCF. Support and maintenance, however, are not available if these applications have been modified in any way. If you do make modifications, it is recommended that you also maintain an *unmodified* set for diagnostic purposes.**

---

## Part One: Getting Started

Before you learn how to use GML, you need some basic information to get started. This information is found in the following chapters:

- Chapter 1, "Introduction" on page 3
- Chapter 2, "What Is GML?" on page 5.

Also refer to the *Document Composition Facility: Introduction to Generalized Markup Language*.



---

## Chapter 1. Introduction

This book assumes that you already know how to use a computer and how to create and edit files in your operating environment. If you need help, refer to the *Document Composition Facility: Introduction to Generalized Markup Language* and the following books:

For the CMS environment:

- *IBM Virtual Machine/System Product: CMS User's Guide*, SC19-6210
- *IBM Virtual Machine/System Product: System Product Editor User's Guide*, SC24-5220.

For the TSO environment:

- *OS/VS2 TSO Terminal User's Guide*, GC28-0645
- *Interactive System Productivity Facility/Program Development Facility for MVS: Program Reference*, SC34-2089.

For the ATMS-III environment:

- *Advanced Text Management System-III Terminal Operator's Guide*, SH20-2425.

---

## Phrases Used in This Book

We use a number of descriptive phrases throughout this book and introduce some of the most commonly used ones here to help you understand them when you come to them later. You can refer to them as needed. A glossary with an extensive listing of words and phrases is located at the back of this book.

### **Document element**

Any part of a document: a single character, a word, or a sentence. Also refers to any part of a document you can identify with a GML tag, such as a paragraph, figure, or heading.

### **GML tag**

In GML markup, an abbreviated name used to identify a document element. In the GML starter set, for example, the :P (paragraph) tag identifies the beginning of a paragraph.

### **Paragraph unit**

Paragraphs and other document elements that have the same structure as a paragraph, such as notes and paragraph continuations.

### **Implied paragraph**

Paragraphs for which no :P tag has been entered, such as definition descriptions, list parts, figure descriptions, and footnotes.



### **Attributes**

Additional information used with tags. Attributes are placed immediately following the tag. GML tags use two kinds of attributes:

- Value attributes – these are one-word descriptions.
- Labeled attributes – these attributes always take an equal sign (=). The attribute is given, followed by =, followed by a value, like this:

PLACE=top

where PLACE is the attribute and top is its value.

### **Default**

The value that is assumed when none is explicitly specified.

### **GML delimiter**

The first character of a tag. The colon (:) is the starter-set delimiter. It calls attention to the fact that “This is a tag!” Some tags need to show where they end. End tags consist of the tag name prefixed with the end tag delimiter. The starter set end tag delimiter is :e.

### **Markup content separator**

This signals the end of the tag and any attributes specified with the tag. A period (.) is the starter set markup content separator.

Before we show you how to use GML, you need to know a few things about GML in general; they're covered in Chapter 2, “What Is GML?” on page 5.

---

## Chapter 2. What Is GML?

To use any text-processing program, you must create a document that includes information, in addition to the text itself, that tells the text processor what to do. This information is called *markup*.<sup>5</sup>

The two basic ways of doing markup using DCF are as follows:

- You can tell the text-processing program (which in our case is SCRIPT/VS) exactly how you want something, such as a numbered list, to look on the page. You do this with *control words*. When you use control words, you have to tell the text-processing program exactly how many lines to skip, how many characters to indent, how far to offset text, and so on.
- You can describe what something is (rather than what it looks like on the page) and let the text-processing program take care of what it looks like on the page. This approach is called *Generalized Markup Language* or *GML*.

GML is a general SCRIPT/VS capability; the starter set of GML tags provided with DCF is a specific set of GML markup designed for general documents.

Using GML, you describe what something is by including tags in the text.

When you request formatting for your document, you give SCRIPT/VS the name of a profile to be used. The *profile* is a file that is processed before your document and that sets things up so SCRIPT/VS can figure out what to do about each of the tags. The starter set profile is called DSMPROF3.

In this book, we tell you how to use the starter set GML tags to mark up your documents.

We also tell you how to use DSMPROF3 to format your documents.

---

### GML Document Types

As we mentioned above, the starter set of GML tags is designed for general documents.

Depending on the kind of organization you work in and the kind of work you do, you might be working on different specific types of documents. For example, the documents you work on could include insurance policies, parts catalogs, maintenance manuals, or even functional specifications.

Each of these types of documents has its own characteristics and, within your organization, new GML might be developed (by your document administrators and your text programmers)<sup>6</sup> to handle the special requirements for the types of documents you produce.

---

<sup>5</sup> This term owes its origins to earlier days in the printing trade, when drafts of manuscripts were literally "marked up" with instructions to typesetting employees.

<sup>6</sup> In this book, the term *document administrator* refers to the person in your organization who is responsible for defining your markup requirements and procedures. The term *text programmer* refers to the person who writes the SCRIPT/VS instructions for processing the markup.

The starter set supplied by IBM is intended to give your organization a starting point in developing its own GML. Because we don't know what your specific requirements are, the starter set is designed for *general* rather than *specific* document types.

If you were, for example, working on parts catalogs, your organization might have GML tags for marking up part numbers, which are specific items you need in a parts catalog. You won't find these tags in the starter set, but you will find tags for headings, paragraphs, lists, examples, figures, tables, and so on—the *general* kinds of things that are found in most documents.

The GML tags in the starter set generally are used as a core for designing specific markup. Therefore, the general rules on markup and most of the tags you will learn about in this book are valid regardless of the specific markup your organization might develop.

Don't be too concerned about this right now. The purpose of this book is to teach you how to use GML in general and the starter set in particular, and to get you ready to use any specific GML that your organization develops.

**Note:** When you use the GML starter set tags, your formatted output may differ from the examples shown in this book because of differences in column line length, fonts, line spacing, device types, indention, and other minor formatting differences.

---

## The Starter Set Profile (DSMPROF3)

When we talk about the *starter set*, we are referring to the markup (GML tags) that you put in your document, the profile (DSMPROF3), and the macro library (DSMGML3). When we say *DSMPROF3* does something, we are referring to the IBM-supplied processing instructions (written in the SCRIPT/VS language and contained in the DSMGML3 library) that process that markup. This distinction is important, because your organization might produce its own processing instructions to process the starter set markup, and the processing results might be different even though the markup is the same.

For example, DSMPROF3 puts five levels of body heading into the table of contents (head levels 0 through 4). If you want only four levels put into the table of contents, your organization might develop a new profile (or modify DSMPROF3) to do that.

In this book, we teach you how to do markup by illustrating how text looks when the IBM-supplied processing instructions process it. But remember, when you use GML you describe what something is, not how it looks on the page.

---

## Creating and Formatting Documents

Using GML and SCRIPT/VS is a two-step process:

1. You create your document (called a *source document*) using a text-editor program or word-processing equipment. In this book, we concentrate on *what* you put in the source document (that is, the markup), not how you put it there.

2. You request formatting for your document and tell SCRIPT/VS what you want done with the formatted output, which could be one of the following:
  - Display it at your terminal (if you are using one). You could do this to check your formatting at the terminal or, if you are using a typewriter-like terminal, to print a copy.
  - Have it printed on a high-speed printer.
  - Have a camera-ready copy prepared for final printing.
  - Save it as a computer file, which can then be displayed at the terminal or printed whenever you need a copy.

When you need to update your document, you modify the source document you created in the first step (again using a text-editor program, distributed system, or word processor), and then you request formatting again.

## Your Computer System

You can use SCRIPT/VS and GML in many different computer systems. The computer system you use has no effect on the GML markup you put into your source document or on the output results; therefore, most of this book applies regardless of the system you use. However, the system you use does affect how you request formatting.

If you use a system called TSO, CMS, or ATMS-III, we show you in this book how to request formatting. (Note, however, that someone in your organization may have set up some procedures or special commands on your computing system to make it easier for you to request formatting by SCRIPT/VS. If that is so, you should use your organization's procedures rather than those we show you here.)

If you are using anything other than TSO, CMS, or ATMS-III, you should find out from your document administrator or supervisor what procedures you should follow to request formatting.

To process SCRIPT/VS documents in batch environments (that is, to have them processed after you have signed off your computer system), you need the IBM Document Library Facility program product. You can direct processed output to either a printer or a document data set. Consult the document administrator at your installation for information about the batch processing procedures in use.



---

## Part Two: Tag Guide

Now you are going to learn how to use all the GML tags. Information about the tags is found in the following chapters:

- Chapter 3, "Paragraphs and Headings" on page 11
- Chapter 4, "Lists" on page 17
- Chapter 5, "Examples and Figures" on page 31
- Chapter 6, "Highlighting, Citing, Noting, and Quoting" on page 45
- Chapter 7, "Tables" on page 51
- Chapter 8, "Cross-References and Footnotes" on page 73
- Chapter 9, "Overall Document Structure" on page 83
- Chapter 10, "Indexing" on page 93
- Chapter 11, "Process-Specific Controls" on page 107.



---

## Chapter 3. Paragraphs and Headings

You will use a few of the many GML tags in almost every document. Learning these common tags can help you to prepare most documents without having to refer frequently to the GML publications. In this chapter, we talk about the tags that are used most often—the ones for paragraphs and headings.

Throughout this document, we show tags in capital letters in the text and in small letters in the examples. You can enter the tags in either uppercase or lowercase in your source document, whichever is easier for you.

---

### Paragraphs

The tag you will use most often is P, for *paragraph*.

Like all tags in the starter set, the paragraph tag begins with a colon (:) and ends with a period (.).<sup>7</sup> So, this paragraph was entered like this:

```
:p.Like all tags in the starter set, the paragraph...
```

You don't have to skip a line before a paragraph, or indent it, or anything like that. It's a good idea to start each sentence within a paragraph on a new line; the lengths of the input lines themselves are not important. Don't end any of the input lines with a hyphen.

**Remember: you don't have to worry about what the paragraph will look like on the page. The starter set takes care of that during formatting.**

We could also have entered the previous paragraph like this

```
:p.  
Like all tags in the starter set, the paragraph...
```

with the tag on a line by itself and the text starting on the very next line.

We will be showing you a lot more paragraphs as we go on; they're all tagged the same way.

Because paragraphs are the *document elements* (parts of a document) you use most often, you now know how to mark up half of what you have to enter.

---

### Headings

The starter set provides seven levels of headings: H0 through H6. (Headings look different, depending on which printer you use to print your document.)

H0 (head level 0) is not used often. You use it when you want to divide a large document into major parts, each part having several chapters. The starter set always starts a new page when it finds an H0 tag.

---

<sup>7</sup> Ending the tag with a period is not always necessary. However, it is never wrong to use the period, so using one to end your tag is a good habit to develop. Later, we discuss in detail when you can omit the period.



The heading “Part Two: Tag Guide” on page 9 was entered like this:

```
:h0 id=part2.Part Two: Tag Guide
```

The “id=part2” is called an *attribute* and, in this case, is a *labeled* attribute (it contains an attribute name (label), an equal sign, and the attribute’s value). Attributes allow us to pass additional information to the processing system. This attribute allows us to make cross-references to the heading from elsewhere in the document. We explain cross-referencing in Chapter 8, “Cross-References and Footnotes” on page 73.

Notice that we start the markup with a colon (:) and end it (after the attribute) with a period (.). If we didn’t want to make a cross-reference to that heading, we would have entered it like this:

```
:h0.Part Two: Tag Guide
```

One important rule to remember about headings is that all the text of the heading must be either on the same line as the end of the markup or on the line following the markup.<sup>8</sup>

That is, our heading could have been entered like this:

```
:h0.  
Part Two: Tag Guide
```

but *not* like this (with the text of the heading split between lines).

```
:h0.Part  
Two: Tag Guide
```

The other important rule about headings is that no tags can be used within the text of the heading. That is, you couldn’t put another tag within the line “Part Two: Tag Guide.”

The H1 (head level 1) tag is used for chapters. You will use it a lot, as you will the rest of the head-level tags.

The starter set always starts a new page when it finds an H1 tag.

The heading for this chapter, Chapter 3, “Paragraphs and Headings” on page 11, was entered like this:

```
:h1 id=gs.Paragraphs and Headings
```

Again, ignore the ID attribute. If we didn’t want to make a cross-reference to that heading, we would have entered it like this:

```
:h1.Paragraphs and Headings
```

The starter set uses the text of headings 0 and 1 to print a running footing. If the heading is too long to make a neat running footing, you can specify a shorter version. The H0 and H1 tags have an attribute, called STITLE (for *short title*), that allows you to specify the shorter version to be used for the running footing.

---

<sup>8</sup> Actually, all the text must be in the same input record, which can be longer than the length of the line on your terminal. Find out how to create longer input records using your editor in order to get long heads into a single record. The maximum length that SCRIPT/VS allows in an input record is 256 characters.

You would specify it like this:

```
:h1 stitle='Short Footing Title'.Very Very Long Chapter Title
```

Notice that the text you want used for the running footing is the value that you give to the STITLE attribute, and that it goes in single quotation marks. Many attributes are specified this way; that is, with the attribute name, followed by an equal sign, followed by the value in single quotation marks.<sup>9</sup> You'll see as we go along that there are other kinds of attributes, and we'll show you how to specify them.

If your short title and your full title are too long to fit on one line, you can enter the text of the heading on the line following the tag and attribute, like this:

```
:h1 stitle='Short Footing Title'.  
Very Very Long Chapter Title
```

If you have a lot of attributes to specify on a tag, you could spread the markup itself over several lines, as long as you don't split an attribute in the middle. (You couldn't put "stitle=" on one line and "Short Footing Title" on the next.) And remember that all the text of the heading must be either on the same line as the end of the markup or on the next line. You could, for example, enter a level 1 heading with two attributes like this:

```
:h1 stitle='Short Footing Title'  
id=part1.  
Very Very Long Chapter Title
```

You'll notice that a period (.) was entered after the last attribute, before the text associated with the heading. The period indicates that the markup is ended—what follows is the text. (The official name of that period is the *markup content separator*.)

**Note:** The ampersand (&) character should not be used in the STITLE text of the H0 or the H1 tag. Also, when the STITLE attribute is not specified, the ampersand should not be used in the text of the section heading. The ampersand is the page number symbol character, and its use in a H0 or H1 tag causes the current page number to be printed in the text of the running footing. If an ampersand must be used in the text of a head level control, use the &amp; symbol.

## *Example of a Head 2*

This heading was entered like this:

```
:h2.Example of a Head 2  
:p.This heading was entered...
```

The starter set puts headings in the correct heading format for each heading level, no matter how you enter them at your terminal. The correct heading format may be all capitals or it may be both uppercase *and* lowercase characters, depending on the output device you use to print your document. You should, however, enter your headings as you want them to appear in the table of

---

<sup>9</sup> You need to include the quotation marks only if the value of the attribute contains blanks or special characters, such as punctuation characters. Here we need them because we have blanks in our attribute value. However, the single quotation marks around an attribute value, like the period at the end of the markup, are *never* wrong, so when in doubt, put them in.

contents, because they are put into the table of contents in the same form as you enter them. (Take a look at the table of contents for this book.)

### Example of a Head 3

This one was entered like this:

```
:h3.Example of a Head 3  
:p.This one was entered...
```

### *And an Example of a Head 4*

Like so:

```
:h4.And an Example of a Head 4  
:p.Like so:
```

Head level 4 is the last level that the starter set puts into the table of contents. Heads 5 and 6 are not included in the table of contents.

**Example of a Head 5:** This was entered like this:

```
:h5.Example of a Head 5  
:p.This was entered...
```

**And a Head 6:** This one was entered like this:

```
:h6.And a Head 6  
:p.This one was...
```

## More About Heads

Did you notice that we didn't enter a colon following the level 5 or the level 6 headings above? The starter set puts it there automatically, but *only* when a P tag immediately follows the H5 or H6 tag line. If you don't have a P tag immediately after the H5 or H6 tag line, you will not get the colon. Furthermore, if your heading text is too long or if other tags follow the heading tag, the colon may end up in an unpredictable location.

You may want the headings in your document numbered. ***You do not have to number the headings yourself.*** The starter set does it for you if you ask it to when you request formatting. We'll discuss how to get the formatting you want and how to get headings numbered when we get to Chapter 13, "Formatting Instructions" on page 119.

---

## Exercise: Paragraphs and Headings

Now that you know how to enter paragraphs and headings, you can enter sixty percent of what you have to do.

See if you can create a document (call it EXER1) that looks like the one below. (See "Markup for Paragraphs and Headings" on page 169 for the correct markup.)

---

Exercise

---

### This Is a Head Three

It has two short paragraphs following it. This is the first paragraph. Each sentence was started on a new line.

This is the second paragraph. Look to see how DSMPROF3 has formatted it (a blank line before and no indentation).

**This Is a Head Five:** The paragraph following it begins on the same line as the head, even though it was entered on a separate line.

The next paragraph under the head five formats like this.

---

End of Exercise

---

To see the results at your terminal (in single-column format rather than the offset style we show you here), do one of the following:

**In TSO enter:**<sup>10</sup>

```
script exer1 prof(dsmprof3) cont
```

EXER1 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**In CMS enter:**<sup>10</sup>

```
script exer1 (prof(dsmprof3) cont
```

EXER1 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**In ATMS-III enter:**<sup>10</sup>

```
script * prof(dsmprof3) cont
```

The document to be formatted (EXER1 – referred to by the \* in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

---

<sup>10</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.

**Note:** When the starter set formats for a display terminal, it lays out the page as if the terminal were a typewriter-like terminal, so you may see only part of a full page at a time.

The CONT (continue) option at the end of each command allows processing to continue after SCRIPT/VS detects an error condition and issues an error message. If you do not include this option, formatting stops at the first error. When SCRIPT/VS encounters an error that is too severe for processing to continue, it stops processing even when CONT is specified.

---

## Chapter 4. Lists

We'll discuss here the various kinds of lists you can create with the starter set. The starter set takes care of all the formatting for lists.

---

### Simple, Unordered, and Ordered Lists

Simple lists are just lists of items, usually brief. For example, here's a simple list:

Formatting Example

```
bread
butter
cheese
bananas
```

End of Formatting Example

You can also have a *compact* simple list. A compact list leaves out the blank spaces between the items of the list:

Formatting Example

```
pears
milk
lettuce
cheese
oregano
paper towels
```

End of Formatting Example

Unordered lists are similar to simple lists, except that each item in an unordered list is preceded by a special symbol. (The special symbol used depends on the device on which you are having your output printed.) You would use an unordered list when the items in the list are fairly long, maybe even many paragraphs, but don't need to be in a specific sequence.

Here's an example of an unordered list:

Formatting Example

- This is an item in an unordered list. To distinguish it from other items in the list, the starter set puts a bullet beside it.  
  
This item consists of two paragraphs. This paragraph does not get a bullet because it is not a separate list item.
- This is a separate list item in our unordered list.

End of Formatting Example

And then there's the ordered list. Use an ordered list when the items you're listing need to be in a specific sequence. An example of an ordered list follows.

Formatting Example

1. Cream butter and sugar together until fluffy.
2. Beat in egg yolks one at a time.
3. Add nutmeg, cinnamon, and vanilla; mix thoroughly.
4. Add flour and beat for five minutes. The batter should be smooth and glossy and stream off the spoon in ribbons.
5. Fold in beaten egg whites.  
  
Do not overmix; the batter should be light and fluffy.

End of Formatting Example

When you create an ordered list, you don't have to number the items yourself. The starter set does it for you. This saves you a lot of work when you decide to insert, delete, or rearrange items.

Unordered and ordered lists can also be compact.

Each of these list types starts with a tag that tells what kind of list it is and ends with a matching end tag.

This is our first use of end tags. End tags are used when it is necessary to indicate that something has ended. (You don't need to end a paragraph, because the start of a new paragraph shows that you're finished with the old one. And you don't need to end a heading because it all goes on one line and nothing else is allowed on that line.)

End tags consist of the tag name, prefixed with the end tag delimiter. The end tag delimiter in the starter set is the characters “:E” — so, if the tag for simple list is “:SL” (which it happens to be), then you know the tag for ending the list is “:ESL.”<sup>11</sup>

The tags for starting and ending the lists we’ve been talking about are as follows:

Tags	List Type
:sl, :esl	Simple list
:ul, :eul	Unordered list
:ol, :eol	Ordered list

**Note:** *Regardless* of the list type, all items within these lists are tagged with the same tag — LI (for *list item*). This saves having to remember a lot of different tag names.

Like headings, list items also can have ID attributes. Any list item that you give a *name* by using the ID attribute can be referred to from some other place within your document.

Even though any list item can be named, you probably will find naming useful within ordered lists only where you can refer back to a particular number. (We’ll talk about this in more detail in Chapter 8, “Cross-References and Footnotes” on page 73.)

List items do not need an end tag, because the list item is ended by another list item or by the end of the list itself.

So, here’s how you would enter the simple list shown previously:

```
:sl.  
:li.bread  
:li.butter  
:li.cheese  
:li.bananas  
:esl.
```

We began the list with SL, marked every item in it with LI, and ended it with the SL end tag.

The period following the SL tag is optional. Because the SL tag is always followed by another tag (LI) and has no text of its own, it is not necessary to end the SL markup with a period. However, it is not wrong to put the period in. If you get into the habit of *always* using the period, then you don’t have to worry about those cases where leaving it out causes an error.

<sup>11</sup> Understand that the end tag is “SL” prefixed with the end tag delimiter (“:E”) rather than “ESL.” This is because your organization could change the end tag delimiter (as they could also change the tag delimiter). You might, for example, have an exclamation point as the tag delimiter and two exclamation points as the end tag delimiter, so your simple list tags would actually be entered as “!SL” and “!!SL.”

In this book we say *matching end tag* or *SL end tag* rather than “ESL” — to remind you that the “E” is part of the *delimiter* and not the tag itself.



If you want to make a list compact, add the `COMPACT` attribute to the `SL` tag. The markup for the simple, compact list we showed you previously would look like this:

```
:sl compact.  
:li.pears  
:li.milk  
:li.lettuce  
:li.cheese  
:li.oregano  
:li.paper towels  
:esl.
```

Notice that the `COMPACT` attribute is different from the attributes we've already looked at; it's just the single word "`COMPACT`." This form of attribute is called a *value attribute*, because you enter only the value and not an attribute label.

Value attributes (single-word attributes) are *not* enclosed in single quotation marks.

Below is the markup of the unordered list:

```
:ul.  
:li.This is an item in an unordered list.  
To distinguish it from other items in the list, the starter set  
puts a bullet beside it.  
:p.This item consists of two paragraphs.  
This paragraph does not get a bullet  
because it is not a separate list item.  
:li.This is a separate list item  
in our unordered list.  
:eul.
```

Notice that the second paragraph in the first item is marked with the `P` tag. The starter set knows what to do with that second paragraph when it's within a list item.

If we had wanted the unordered list to be compact, we would have put the `COMPACT` attribute on the `UL` tag, just as we did with `SL`. The `COMPACT` attribute applies only to the space between list items, and not to any space between paragraphs within a list item, so you would probably want to use it only on lists with short list items.

And here is the markup of the ordered list:

```
:ol.  
:li.Cream butter and sugar together until fluffy.  
:li.Beat in egg yolks one at a time.  
:li.Add nutmeg, cinnamon, and vanilla;  
mix thoroughly.  
:li.Add flour and beat for five minutes.  
The batter should be smooth and glossy  
and stream off the spoon in ribbons.  
:li.Fold in beaten egg whites.  
:p.Do not overmix; the batter  
should be light and fluffy.  
:eol.
```

Again, we could have used the COMPACT attribute, just as we did with our simple list.

---

## Definition Lists

Definition lists are used when you want to pair a term or phrase with its description.

Here's an example of a definition list with headings:

Formatting Example	
<b>TERM</b>	<b>DESCRIPTION</b>
<b>gopher</b>	A burrowing rodent that feeds on roots of plants.
<b>lawn</b>	Gopher highway.  Can be identified by dinner-plate-sized mounds of dirt where grass used to be.
<b>agapanthus</b>	Lovely flowering plant, the roots of which are the preferred food of gophers.  If your flourishing agapanthus suddenly keels over, it means a gopher has had a feast.
End of Formatting Example	

This list differs from the other lists in several ways. This list can have headings; it also has three labeled attributes and two value attributes.

To mark up a list like this, you must first decide how much space you want reserved for the term and the term heading (the left-hand side). If it's 10 character spaces, you don't have to specify anything, because 10 character spaces is the *default*. (A default is the value of an attribute that is assumed if you don't explicitly specify one.)

For the previous list we specified 25 millimeters, to give us enough space for the term "agapanthus" as well as additional space for the *gutter* (the space between the terms and their descriptions).

**Note:** The default unit of measure is actually *character spaces*. But to provide a more universal example, we used millimeters. For more information about space units in general, and character spaces in particular, see "Space Units" on page 39.

You must also decide whether you want the headings and terms to be highlighted. Highlighting the terms and headings is the default, which we chose to do. The appearance of highlighting depends on the device you are printing on. Highlighting is explained in Chapter 6, "Highlighting, Citing, Noting, and Quoting" on page 45.

The start tag for a definition list is `DL` and, like the other lists, it has a matching end tag. `DL` has five attributes:

```
:DL HEADHI=highlight level  
      TERMHI=highlight level  
      TSIZE=horizontal space  
      BREAK  
      COMPACT.
```

where:

**HEADHI**=highlight level

allows you to specify the highlighting for the headings. The default for **HEADHI** is highlight level 3.

**TERMHI**=highlight level

allows you to specify the highlighting for the terms. The default for **TERMHI** is highlight level 2.

**TSIZE**=horizontal space

allows you to specify the combined space for the definition terms and the *gutter* (space) between the terms and headings. The default for **TSIZE** is 10 character spaces. Character spaces is discussed in more detail in "Space Units" on page 39.

**BREAK**

allows you to specify that the description should start on the next line after the term if the length of the term exceeds **TSIZE**. In such cases, the default is to leave one blank after the term and then start the definition.

**COMPACT**

allows you to specify when you don't want blank lines between the individual terms and descriptions. The default is to put a blank line between the individual terms and descriptions.

The default for each of the attributes is used unless you specify something else when you enter your markup.

Within the definition list, you tag the heading for the terms with the `DTHD` (definition term heading) tag and the heading for the descriptions with the `DDHD` (definition description heading) tag. Similarly, you tag the terms with the `DT` (definition term) tag and their accompanying descriptions with the `DD` (definition description) tag.

So a definition list is made up of headings for the terms and descriptions (`DTHD` and `DDHD`) and pairs of `DT` and `DD` tags, within `DL` and its matching end tag.

Here's the markup for the definition list shown page 21:

```

:dl tsize=25mm.
:dthd.TERM
:ddhd.DESRIPTION
:dt.gopher
:dd.A burrowing rodent that feeds on roots
of plants.
:dt.lawn
:dd.Gopher highway.
:p.Can be identified by dinner-plate-sized
mounds of dirt where grass used to be.
:dt.agapanthus
:dd.Lovely flowering plant,
the roots of which are the
preferred food of gophers.
:p.If your flourishing
agapanthus suddenly keels over,
it means a gopher has had a
feast.
:edl.

```

Did you notice the P tag? The starter set knows exactly what to do with a P tag within a definition description.

Notice also that we did not have to put the value of the TSIZE attribute in single quotation marks. Single quotation marks are required only when special characters or blanks are in the attribute value. (For example, if you have a period in an attribute value and do not put the value in quotation marks, SCRIPT/VS thinks it is the period that indicates the end of the markup. Similarly, blanks and other special characters, such as punctuation characters, can be misinterpreted by SCRIPT/VS unless you use quotation marks.) The use of quotation marks is never wrong on labeled attributes (attributes with an equal sign), so when in doubt, put them in.

Two rules you need to remember about definition lists are that all the text for the definition term, definition term heading, and definition description heading must be either on the same line as the tag or on the next line, and that no tags are permitted within the text. (These are the same rules you've already learned for headings.)

If we use the same markup shown previously and use several different attributes on the DL tag, we can change the appearance of the listing.

For example, by changing only the first line of our markup for the definition list as follows:

```
:d| tsize='8' headhi='1' termhi='0' break.
```

the output would look like this:

Formatting Example	
TERM	DESCRIPTION
gopher	A burrowing rodent that feeds on roots of plants.
lawn	Gopher highway.  Can be identified by dinner-plate-sized mounds of dirt where grass used to be.
agapanthus	Lovely flowering plant, the roots of which are the preferred food of gophers.  If your flourishing agapanthus suddenly keels over, it means a gopher has had a feast.

End of Formatting Example

As you can see in the previous example, specifying values for the HEADHI and TERMHI attributes changes the highlighting used for the definition list headings and terms. See Chapter 6, “Highlighting, Citing, Noting, and Quoting” on page 45 for information on highlighting.

## Glossary Lists

A glossary list is an additional specialized type of list for which the starter set has tags.

Following is an example of items that might appear in a glossary. (In fact, they appear in the glossary at the back of this book.)

Formatting Example	
<b><i>formatter:</i></b>	A computer program that prepares a source document to be printed.
<b><i>Generalized Markup Language (GML):</i></b>	A language that can be used to identify the parts of a source document without respect to particular processing.
<b><i>GML delimiter:</i></b>	A special character that denotes the start of GML markup. In the starter set, it is initially a colon (:).

End of Formatting Example

The glossary usually appears in the back matter of a book. Look at the “Glossary” on page 175 to see what this book’s glossary looks like.

A glossary term is printed in highlight level 2 followed by a colon.

**Note:** The appearance of highlighting depends on the device you are printing on. For more information on highlighting see Chapter 6, “Highlighting, Citing, Noting, and Quoting” on page 45.

The start tag for a glossary list is `GL`. You end the glossary list with its matching end tag.

You can use the `COMPACT` attribute with glossary lists, but it probably wouldn't be a good idea, because the list would look too crowded.

The other attribute that you can specify on the `GL` tag is `TERMHI`. It works the same on glossary lists as it does on definition lists – the default is to highlight the term.

Following is the markup for the glossary list example:

```
:gl.
:gt.formatter
:gd.A computer program that prepares
a source document to be printed.
:gt.Generalized Markup Language (GML)
:gd.A language that can be used to identify the
parts of a source document without respect to particular processing.
:gt.GML delimiter
:gd.A special character that denotes the start of
GML markup. In the starter set, it is initially a colon (:).
:egl.
```

All the text for the glossary term must be either on the same line as the tag or on the next line. No tags are permitted within the text.

---

## Nested Lists

A nested list is a list contained within another list. The starter set allows all of the lists we've just described to be nested within another list of the same type or of a different type, many levels deep. Because each nested list is indented from the left edge of the previously nested list, the only restriction is the width of your column.

Perhaps the most common use of nested lists is ordered lists, such as the example that follows.

---

### Formatting Example

1. First item in a first-level list.
2. Second item in a first-level list. It has a nested list within it.
  - a. First item in a second-level list.
  - b. Second item in a second-level list.
3. Third item in a first-level list.

---

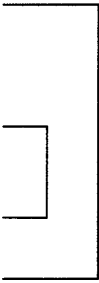
End of Formatting Example

If you remember that each list must start and end with a tag that says what type of list it is, and that all list items within a list are marked as list items, and that you don't have to worry about numbering the items, you will have no trouble nesting lists. This is how the nested list was marked up:

```

:ol.
:li.First item in a first-level list.
:li.Second item in a first-level list.
It has a nested list within it.
:ol.
:li.First item in a second-level list.
:li.Second item in a second-level list.
:eol.
:li.Third item in a first-level list.
:eol.

```



The diagram consists of a large right-facing square bracket on the right side of the code block. The top horizontal line of the bracket aligns with the start of the first-level list (:ol.). The middle horizontal line aligns with the start of the second-level list (:ol.). The bottom horizontal line aligns with the end of the first-level list (:eol.).

**Note:** The lines to the right of the markup are to help you see where the lists begin and end; they're not part of the markup.

As you can see, the second list is entirely contained within a list item of the first list. Similarly, when you are nesting lists within a definition list, the nested list must be entirely contained within a definition description of the definition list.

The same general approach is used for nesting lists of different types. Following is an unordered list with an ordered list within it, and a definition list within that:

Formatting Example

- When creating a CMS file, you must specify the following:
  1. filename
  2. filetype
 

In System Information we generally use these file types:

*SCRIPT* For files containing text.

SCRIPT/VS assumes a filetype of SCRIPT; thus, you do not have to specify it when you request formatting.

*EXEC* For files containing CMS commands.

*OPTIONS* For files containing the options to be used when SCRIPT/VS formatting is requested.
  3. filemode
 

Filemode defaults to A1 if not specified.
- When creating a TSO file...

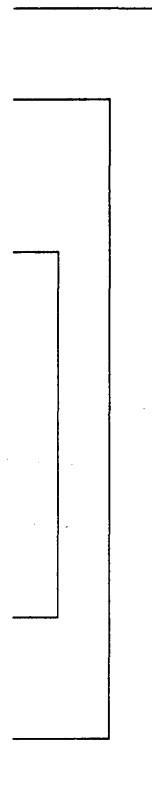
End of Formatting Example

Here's the markup for the previous example:

```

:ul.
:li.When creating a CMS file,
you must specify the following:
:ol.
:li.filename
:li.filetype
:p.In System Information we
generally use these file types:
:d1 termhi=1.
:dt.SCRIPT
:dd.For files containing text.
:p.SCRIPT/VS assumes a filetype of
SCRIPT; thus, you do not have to
specify it when you request formatting.
:dt.EXEC
:dd.For files containing CMS commands.
:dt.OPTIONS
:dd.For files containing the options
to be used when SCRIPT/VS
formatting is requested.
:ed1.
:li.filemode
:p.Filemode defaults to A1
if not specified.
:eol.
:li.When creating a TSO file...
:eul.

```



## List Parts

Sometimes in an ordered list you want to break the list for some explanatory material and then resume the numbering where you left off. The starter set lets you do this with the LP (list part) tag.

For example, suppose you wanted to do the following:

### Formatting Example

1. Saute the shallots and chopped mushrooms until the shallots are tender and the liquid from the mushrooms has cooked away.
2. Brown the sausage and add to the mushroom mixture.

The above may be prepared several hours in advance and refrigerated. Then, 30 minutes before serving time, finish the dish.

3. Mix one can of tomato sauce with the mushroom and sausage mixture and bring to a slow simmer.
4. Add the heavy cream and immediately pour into a casserole.
5. Bake at 350 degrees for 15 minutes.

End of Formatting Example



We used the LP tag to break the list. It was then continued by the next LI (list item) tag.

If you have more than one list part between items in a list, begin each list part with the LP tag.

Here's the markup for the previous example:

```
:ol.  
:li.Saute the shallots and chopped  
mushrooms until the shallots are tender and the liquid from the  
mushrooms has cooked away.  
:li.Brown the sausage and add to the mushroom mixture.  
:lp.The above may be prepared several  
hours in advance and refrigerated.  
Then, 30 minutes before serving time, finish the dish.  
:li.Mix one can of tomato sauce  
with the mushroom and sausage mixture  
and bring to a slow simmer.  
:li.Add the heavy cream and immediately  
pour into a casserole.  
:li.Bake at 350 degrees for 15 minutes.  
:eol.
```

List parts can be used in any of the list types.

---

## Implied Paragraphs

We mentioned previously that all paragraphs were tagged with a P. That's not entirely true. You may have noticed when we talked about list items and definition descriptions that we said that any paragraphs after the first one were tagged with a P. Of course, the first one is also a paragraph, but because we've already described it as a list item (with the LI tag) or a definition description (with the DD tag), we don't have to say that it is a paragraph, because that is "implied" by the LI or DD.

Implied paragraphs show up in a number of places; you'll be able to see from the examples where you need an explicit P tag and where you don't.

---

## Exercise: Lists

See if you can create a document that looks like this (call it EXER2). The markup is shown in "Markup for Lists" on page 170:

---

### Exercise

---

Please send the following items:

<b>Number</b>	<b>Description</b>
<b>A107C2</b>	Class A Widget, 100 x 200 mm in assorted colors (no white).
<b>B321DJ</b>	Brown and yellow Whuzzit, with linkage for optional nutcracker attachment.
<b>BY7532</b>	Assorted gimcracks, with the following characteristics: <ul style="list-style-type: none"> <li>• Soft</li> <li>• Cuddly.</li> </ul>

When I get around to it, I will

1. Balance my checkbook.
2. Prune the shrubbery.
3. Write to the following:

My mother in New Rochelle  
 My great aunt in Detroit  
 My niece in Ossining  
 My friend in South Bend.

4. Clean out the garage.

---

End of Exercise

---

To see the results back at your terminal (in single-column format), do one of the following:

**TSO users enter:**<sup>12</sup>

```
script exer2 prof(dsmprof3) cont
```

EXER2 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**CMS users enter:**<sup>12</sup>

```
script exer2 (prof(dsmprof3) cont
```

EXER2 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

---

<sup>12</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.

**ATMS-III users enter:**<sup>12</sup>

```
script * prof(dsmprof3) cont
```

The document to be formatted (EXER2 – referred to by the \* in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

## Chapter 5. Examples and Figures

General documents often have many examples and figures. We'll show you how to do those next.

### Examples

Examples are easy; you just begin them with an XMP tag and end them with an XMP end tag. Examples can occur anywhere in text except within other examples, figures, or footnotes.

Within the range of the XMP tag, the starter set turns off formatting and treats each input line as an output line; it does not link them into paragraphs. The starter set keeps all of the lines of an example together, so an example should not be longer than the column. If an example cannot fit within the remaining space of a column, the example starts at the beginning of the next column.

Here's a typical example:

```

_____ Formatting Example _____
10 LET A = B
20 IF A GT C THEN GO TO 40
30 LET A = C
40 PRINT A, C
_____ End of Formatting Example _____

```

The markup for the previous example looks like this:

```

:xmp.
10 LET A = B
20 IF A GT C THEN GO TO 40
30 LET A = C
40 PRINT A, C
:exmp.

```

XMP has a DEPTH attribute, which allows you to specify an amount of vertical space to leave for you to paste in the example. The space is specified in *vertical space units*, which can be any of the ones described in "Space Units" on page 39. If you specify the DEPTH attribute and put some text in your example, the starter set gives you the vertical space first, and then puts in your text. The DEPTH attribute works the same way for examples as it does for figures. We describe DEPTH in more detail in "Figures" on page 32.

The normal formatting of text (justification and concatenation) is suspended within an example; each input line becomes a separate output line. Other GML tags should be used with caution within examples, because concatenation of input lines is suppressed. Unexpected line breaks or concatenation of source lines may occur when GML tags are used within examples.

---

### Paragraph Continuation

Sometimes, following an example or a list, you may want to continue a paragraph that was begun before the example or list. You do this by using the paragraph continuation (PC) tag instead of the P tag following the example or list. For example:

---

Formatting Example

When you enter

```
loc /A + B
```

the editor locates the next line containing A + B.

---

End of Formatting Example

The markup for the previous example would look like this:

```
:p.When you enter  
:xmp.  
loc /A + B  
:exmp.  
:pc.the editor locates the next line  
containing A + B.
```

Because no apparent difference exists between how the starter set formats a paragraph and how it formats a paragraph continuation, you may wonder why you have to bother to specify them differently.

Here's why:

The formatting style the starter set uses is called *block paragraph*, where the first line of the paragraph is not indented. However, if your organization were to change the formatting style to indent paragraphs, you would not want your paragraph continuations to be indented also.

By correctly specifying the markup at the outset, you never have to worry about a change in formatting style producing unexpected output.

---

### Figures

Figures are a little more complicated than examples, but only because figures have more attributes, which let you do more things.

For instance, you can choose to have a figure formatted within a column (if you're formatting for two columns or the offset style we are using in this publication) or formatted the full width of the page. And you can either have it placed exactly where you entered it (inline), or you can allow the starter set to move it so as to fill out a page. (The figure *floats* (moves) to the top or bottom—you can select which—of the next column or page.)

You can also choose to have a figure set off with rules across the page, put in a box, or formatted with no frame at all.

You can tell the starter set you want space left in the figure so you can paste in a picture.

You can give the figure a caption, which also causes it to be listed in the list of illustrations. You can extend the figure caption with a figure description. (Only the caption goes into the list of illustrations.)

If you give your figure a name (ID) and a caption, you can make cross-references to that figure. (We cover this in Chapter 8, "Cross-References and Footnotes" on page 73.)

Because you can do all these things, the markup for a figure is more extensive than anything you've seen before. But, all you have to do is figure out what you want and then ask for it.

Here's the markup for an inline figure that is the width of the column. We've let the frame default to rules.

```
:fig place='inline' width='column'.  
Just as with XMP,  
the input lines entered in the  
figure body are treated  
as output lines;  
they won't be formatted.  
:efig.
```

The figure begins with a FIG tag with PLACE and WIDTH attributes and ends with a FIG end tag. We've made this figure inline and haven't put a caption on it. Notice that, like examples, the text is printed as it is entered:

Formatting Example

```
Just as with XMP,  
the input lines entered in the  
figure body are treated  
as output lines;  
they won't be formatted.
```

End of Formatting Example

The lines (called *rules*) are produced by the starter set when the FIG tag uses the default for the FRAME attribute.

The normal formatting of text (justification and concatenation) is suspended within a figure; each input line becomes a separate output line. Other GML tags should be used with caution within figures, because concatenation of input lines is suppressed. Unexpected line breaks or concatenation of source text lines may occur when GML tags are used within figures. No hyphenation or spelling check is performed.

Let's take a look at the FIG tag itself. FIG has five attributes, so a fully specified FIG tag would look like this:

```
:FIG ID=name  
    PLACE=placement  
    WIDTH=horizontal space  
    FRAME=frame  
    DEPTH=vertical space.
```

where:

**ID**= name

allows you to make cross-references to a figure (when you also use the FIGCAP tag). We describe cross-referencing in Chapter 8, "Cross-References and Footnotes" on page 73.

**PLACE**= placement

allows you to specify where you want the figure to go. You can specify any one of three placements:

**TOP** Causes your figure to float to the top of the next available column or page. If necessary, text is moved in front of the figure to fill out the current column or page. TOP is the default.

**INLINE** Causes your figure to be set exactly where you entered it; no text is moved around it.

**BOTTOM** Causes your figure to float to the bottom of the next available column or page. If necessary, text is moved in front of the figure to fill out the column or page.

**Note:** Usually, you will want a floating figure to have a caption and a cross-reference to it.

**WIDTH**= horizontal space

allows you to specify the width of your figure in one of three ways:

**space units** Can be specified as any of the horizontal space units described in "Space Units" on page 39.

If the width you specify for your figure is narrower than the width of the current column, the starter set places the figure within the width of the column. If the width is wider than the width of the current column, the figure is set the full page width.

Specifying space units rather than PAGE or COLUMN allows you to create figures that are somewhat independent of the general layout of the document and gives you a flexibility that COLUMN or PAGE does not allow. Then, if you print the document with different column or page widths than originally specified, or if you decide to change a single-column document to double column, you do not have to change the markup for your figure.

For these reasons, it is a good idea to use space units in specifying your figure width rather than COLUMN.

- COLUMN** Sets the figure to the width of the current column.
- Space units should be explicitly used where there is any possibility that the column width might change at some future time.
- PAGE** Sets the figure width to the full page width. PAGE is the default.

**FRAME=frame**

allows you to specify one of four types of frames:

- RULE** Gives you the rules we showed you in the previous example. RULE is the default.
- BOX** Gives you a box around the figure.
- NONE** Puts nothing around the figure.
- 'string'** Allows you to specify a string of characters to be used in place of the rules. For instance, if you said

```
frame='*'
```

you'd get a line of asterisks instead of a rule.

**Note:** When a figure frame defaults to rules or when a specific *string* is requested, the rule or string appears across the top and bottom of the figure when the figure is placed *inline*. When a figure is floated to the top of a page, only the rule or string at the bottom of the figure appears. Similarly, when a figure is floated to the bottom of a page, only the rule or string at the top of the figure appears.

**DEPTH=vertical space**

allows you to specify an amount of vertical space to be left for pasting in graphics and so on. The vertical space is specified in *vertical space units*, which can be any of the ones described in "Space Units" on page 39.

If you specify the DEPTH attribute and also put some text in your figure, the starter set gives you the vertical space first and then puts in your text.

The depth attribute can be used on the XMP tag, with the same result.

Enter the FIG tag and its attributes before you enter the contents of your figure. Once the figure content has been completed, you can decide if you want a figure caption.

If you want a figure caption (and you must have one to get the figure numbered and placed in the list of illustrations), use the FIGCAP tag at the end of the body of your figure, before the FIG end tag. All the text of the caption must go either on the same line as the FIGCAP tag or on the next line, and it must not contain any tags.

You don't have to enter the word "Figure" or the figure number. The starter set takes care of all that for you.

If you have a lot that you want to say about the figure, you can also have a figure description, which can go on for as many lines as you need. It won't be put into the list of illustrations, but it will be added to the figure caption in the text.



You do this with the FIGDESC (figure description) tag, which is placed immediately after the FIGCAP tag, before the FIG end tag.

To summarize, the information used to identify a starter set figure can include both a figure caption and a figure description.

The FIGCAP tag:

- Numbers the figure automatically
- Formats the text that was included with the FIGCAP tag
- Places the figure number and text from the FIGCAP tag into the list of illustrations.

The FIGDESC tag:

- Enters a colon (:) after the text of the FIGCAP
- Follows the colon with the text of the figure description (FIGDESC) tag.

Now we'll show you a couple of examples of what we've been talking about. Be aware that our formatted figure examples do not look the same as when used with an unmodified GML starter set.

### Formatting Example

This is an inline figure with a box around it.  
Because we said it was 16 picas wide, the starter set can decide whether it can fit within the column or must be full-page width.

**Figure 1. An Inline Figure.** This example also has a figure description associated with it, which can be several lines.

### End of Formatting Example

Here is the markup for the previous figure:

```
:fig width=16p place=inline frame=box.  
This is an inline figure with a box around it.  
Because we said it was 16 picas wide, the starter  
set can decide whether it can fit within the  
column or must be full-page width.  
:figcap.An Inline Figure  
:figdesc.This example also has a figure  
description associated with it, which can be several lines.  
:efig.
```

Notice that you do not have to worry about the punctuation between the figure caption and the figure description. The starter set supplies the punctuation when it sees the FIGDESC tag.

Here's an example of a full-page figure with vertical space left in it. We'll give this one a frame composed of asterisks (\*\*\*\*) by specifying '\*' as the value of the FRAME attribute on the FIG tag.

Formatting Example

\*\*\*\*\*

## Figure 2. A Page-Wide Figure with Space

\*\*\*\*\*

End of Formatting Example

Here is the markup for the previous figure:

```
:fig place='inline' depth='10p' width='page' frame='*'.
:figcap.A Page-Wide Figure with Space
:efig.
```

The four attributes we've specified with this figure tag aren't in any particular order; the order of the attributes is not important on any of the tags.

In this chapter we've made all of our figures inline to show examples with their explanations and markup. However, this is not what you usually want to do with figures, because if you make a figure inline and there isn't enough space for it in the column, it leaves the rest of that column empty and goes to the next column or page to fit the figure in. This can leave big areas of white space in your document where you don't really want them. As a rule, all but the very smallest of figures should be allowed to float (this is why floating to the top is the default on the PLACE attribute).

## Page Segments

You can include page segments that may be a picture, logo, or some other kind of image when printing on a page printer that is not a PostScript device (for PostScript devices, see "PostScript Images" on page 39). To integrate a page segment into your text, you need to specify the SCRIPT/VS .SI [Segment Include] control word.

The .SI [Segment Include] control word identifies the place in your document where you want that image placed. If you want the included segment to have a caption, you can specify that the page segment be included within the FIG start and end tags, like this:

```
:fig place=inline.
.si segname
:figcap.The Golden Fleece
:efig.
```

where:

*segname* is the name of the file that contains the image to be imbedded.

If the document is printed on an MVS, an ATMS-III, or a TSO system, *segname* must be the name of a member in a segment library. However, on a CMS system, *segname* must be a CMS file name.

If the document is printed on a VSE system, segment libraries are not searched by DCF, but can be used at print time.

The image contained in the file named *segname* is included in the document when it is printed on the appropriate page printer, provided the page segment is available.

If the page segment you request is in the default library, you don't need to specify the library that the page segment is in. If, however, the page segment is in a segment library you have created, you must specify the SEGLIB option on the SCRIPT command. SCRIPT/VS searches only one segment library, either the default library or the one that you created.

## Reserving Space for a Page Segment

Sometimes you may want to reserve space for a page segment that is incomplete or that does not yet exist. You can reserve this space by specifying a width, a depth, or both. For example, if your document were one-column format and you expected the page segment to take up a large part of the output page, you might specify:

```
.si clash width 6i depth 7.5i
```

which reserves 6 inches of horizontal space and 7.5 inches of vertical space for the proposed page segment. ("Clash" is the segment name.) Later, when the actual page segment is included in your document, any width and depth values you have specified are replaced by the actual size of the page segment as it has been specified in the segment library.

See “Instructions to SCRIPT/VS” on page 123 for descriptions of the SEGLIB and NOSEGLIB options and refer to the *Document Composition Facility: SCRIPT/VS User’s Guide* for more information about using page segments.

---

## PostScript Images

If you plan to print your document on a PostScript device, you can include PostScript image files in the document. PostScript image files contain graphics or text that are created using the PostScript page-description language. PostScript image files are included in a DCF document using the .PO [PostScript] control word rather than the .SI [Segment Include] control word.

The .PO control word identifies the place in the document where you want the PostScript image to be included and allows you to specify:

- Whether the image is placed within a line of text
- The width and depth of the image
- The dimensions by which you want the image to be scaled (made larger or smaller).

To learn how to use the .PO control word, refer to the *Document Composition Facility: SCRIPT/VS User’s Guide* for more information about using page segments.

If you want an included PostScript image to have a caption, you can specify that the image be included within FIG start and end tags like this:

```
:fig place=inline.  
.po image1  
:figcap.Wapiti  
:efig.
```

**Note:** You must specify a PostScript device type on the SCRIPT command to format your document so that PostScript images are included, and you must print the document on a PostScript printer (the IBM 4216 Personal Page Printer, for example).

---

## Space Units

Space units supported by SCRIPT/VS are:

```
inch  
millimeter  
centimeter  
pica  
cicero  
line (vertical space unit)  
character (horizontal space unit)  
em-space  
device units.
```

You may enter space units in either uppercase or lowercase letters. Enter fractional numbers with either a period or a comma (1.25 and 1,25 are both correct).

**Note:** If you enter an attribute value using a period as a decimal point, you must enclose that value in single quotation marks; otherwise SCRIPT/VS interprets that period as the end of your markup. Here's how such a markup should be coded:

`depth='3.2i'`

Explanations of the various space units and how to specify them are as follows:

**inch** There are 39.37 inches in a meter.  
Specify inches as a decimal number (which can have a decimal point or comma and up to two decimal places), followed by an "i," like this:

<code>3i</code>	(3 inches)
<code>2.5i</code>	(2.5 inches)
<code>4,25i</code>	(4.25 inches)

**millimeter** One thousandth of a meter. There are 25.4 millimeters in an inch.

Specify millimeters, like inches, as a decimal number (which can have a decimal point or comma and up to two decimal places) followed by "mm," like this:

<code>85mm</code>	(85 millimeters)
<code>40.25mm</code>	(40.25 millimeters)
<code>15,5mm</code>	(15.5 millimeters)

**centimeter** One hundredth of a meter. There are 2.54 centimeters in an inch.

Specify centimeters just like millimeters, except use "cm" instead of "mm," like this:

<code>1cm</code>	(1 centimeter)
<code>2.5cm</code>	(2.5 centimeters)
<code>18,25cm</code>	(18.25 centimeters)

**pica** A standard printer's measurement in the United States and Great Britain. A pica is .1663 inches, and there are twelve points to a pica (roughly, 6 picas to an inch and 72 points to an inch).

Specify the number of picas, followed by a "p," followed by the number of points, like this:

<code>3p2</code>	(3 picas and 2 points)
<code>12p</code>	(12 picas, no points)
<code>p8</code>	(no picas, 8 points)

Picas can be specified in tenths of units (for example, `1.5p` = 1.5 picas).

**cicero** A standard printer's measurement in most countries except for the United States and Great Britain. Ciceros (and didot points) are comparable to picas and points. The cicero is .1776 inches, and there are twelve didot points in a cicero.

Specify the number of ciceros, followed by a "c," followed by the number of didot points, like this:

4c3	(4 ciceros and 3 didot points)
5c	(5 ciceros, no points)
c8	(no ciceros, 8 didot points)

Ciceros can be specified in tenths of units (for example, 1.5c = 1.5 ciceros).

**lines** A number of output lines, specified as just a number, like this:

4 (lines)

which would leave 4 blank lines in the output.

For page printers, line spaces are equal in size to the line spacing of the current font.

As a general rule, if you are leaving space for a figure to be pasted in or included at the time of printing, you would not want to use lines as your measure. The space left then depends on the size of the font you are using for page printers, or the number of lines per inch for line printers. Because this could change when you print on different devices or even on the same device with a different lines-per-inch specified, the space left for your figure could change, and the figure wouldn't fit. Therefore, you are better off if you use one of the other measures, such as inches or picas, on your DEPTH attribute.

**characters** A number of output characters, also specified as just a number, like this:

12 (characters)

which specifies a width equal to 12 character spaces in the default font.

For page printers, character spaces are equal in size to the figure space (approximately one-half an em-space) of the default font. The default font is the initial font at the start of the formatting run. For all other devices, a character width is the width of a blank.

You would use this specification when the width you specify is keyed to the character count. (If you are formatting for a page printer, use ems instead of characters.) Like lines, the actual space can vary depending on your output device, so you would use it only when the character count is the critical element in your measurement.

**ems** A number of ems, where em is the width or the height of the character "m" in the current font.

For the 3800 Model 1, this width may be:

1/10 inch	(2.54 millimeters)
1/12 inch	(2.117 millimeters)
1/15 inch	(1.693 millimeters)

depending on the "pitch" (the number of characters per inch) of the font being used.

The height may be:

1/6 lpi (lines per inch)

1/8 lpi

1/12 lpi

You can specify ems as a decimal number followed by an "m" or "mh" for horizontal em space, or an "mv" for vertical em space, like this:

12m (12 horizontal ems)

20.5mh (20. horizontal ems)

14.25mv (14.25 vertical ems)

Fractional ems are rounded to the nearest device unit. For line devices, ems are translated to the nearest whole number.

### device units

A number of device units, the size of which depends on the device for which SCRIPT/VS is formatting. Device units are specified as a whole number (no decimal points allowed), followed by a "dh" for horizontal device units, or a "dv" for vertical device units, like this:

45dh (45 horizontal device units)

600dv (600 vertical device units)

For line devices, only vertical line spacing is possible. SCRIPT/VS converts other measures to the closest it can come in line spaces. Similarly, horizontal spacing depends on the device being used, and SCRIPT/VS comes as close as it can come in line spaces.

For the IBM 4250 Printer, SCRIPT/VS can space as small as 1/600 inch, both vertically and horizontally. For the IBM 3820 Page Printer and the IBM 3800 Printing Subsystem Model 3 and Model 6, the smallest space possible is 1/240 inch. For the IBM 1403 Printer, the closest SCRIPT/VS can space is one character space horizontally and one line space vertically.

For PostScript devices, SCRIPT/VS can space as small as 1/72000 inch; however, this device unit does not necessarily match the resolution of your PostScript device.

For example, suppose you specify 60dv on the DEPTH attribute of the FIG tag.

- On the 4250, a space of 1/10 inch would be placed
- On the 1403, a space of 10 inches would be placed.

You can see by the wide range of device units available on the various printers that specifying spacing in device units will probably make your document *device dependent*. That is, the places in your document where you specify spacing in device units will probably format correctly only when the document is printed on the device for which you have specified the device units.

## Exercise: Examples and Figures

See if you can create a document that looks something like this (call it EXER3); of course, your figures will come out numbered 1 and 2 instead of what we show here. The markup is shown in "Markup for Examples and Figures" on page 171.

### Exercise

Here's an example of some BASIC statements:

```
10 PRINT USING 55 A, B, C
20 LET J = K + 2
30 IF J = X GO TO 80
```

#### AN INLINE, PAGE-WIDE FIGURE

Because the contents of a figure format EXACTLY as entered, you can enter blanks on the line (before text) and the lines will print exactly the same as they were entered!

**Figure 3. An Inline, Page-Wide Figure.** This is the first figure I have entered myself.

This paragraph follows the FIG end tag. Here we have another figure (inline and column wide):

Let's create another figure that is column wide, which will create a second item for a list of illustrations in a future exercise.

#### Figure 4. A Column-Wide Figure

### End of Exercise

To see the results at your terminal (in single-column format rather than the *offset style* we show you here):

#### TSO users enter:<sup>13</sup>

```
script exer3 prof(dsmprof3) cont
```

EXER3 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

#### CMS users enter:<sup>13</sup>

```
script exer3 (prof(dsmprof3) cont
```

EXER3 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

#### ATMS-III users enter:<sup>13</sup>

<sup>13</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.



```
script * prof(dsmprof3) cont
```

The document to be formatted (EXER3 – referred to by the &asterik. in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

## Chapter 6. Highlighting, Citing, Noting, and Quoting

This chapter shows several ways in which you can emphasize text.

### Highlighted Phrases

You've already seen many examples of *highlighted* text in this book. In documents produced on a page printer, highlighting is text that is printed in an *italic font*, a **bold font**, or a combination of *italic and bold* versions of the current font. If you request printing on a 3800 Model 1, the highlighting is done using underscoring, a bold font, or a combination of underscoring and bold font. (The markup, however, is the same no matter which device prints your document.)

We number the highlighting types: 0; 1; 2; and 3. Highlight 0 is just what you're looking at now — no emphasis.

We have four tags for highlighting, but they are easy to remember. They all consist of HP (to start the highlighted phrase) plus the number for the type of highlighting you want; that is, to start highlight 2, you use HP2, and to end it, you use EHP2.

If you want to do something like this:

Formatting Example

**Attention!** This is *very* important! ***Don't go out in the rain without your galoshes!***

End of Formatting Example

you would enter it like this:

```
:p.:hp2.Attention!:ehp2.
This is :hp1.very:ehp1. important!
:hp3.Don't go out in the
rain without your galoshes!:ehp3.
```

Do you remember the HEADHI and TERMHI attributes on definition and glossary lists that we told you about in "Definition Lists" on page 21? The number you specify after the equal sign on the HEADHI and TERMHI attributes (0, 1, 2, or 3) actually refers to the same levels of highlighting that you get with the HP tag. (Of course, if you don't specify these attributes, you get the default values for the highlighting.)

You can also *nest* highlighted phrases — that is, put one inside another. The starter set keeps track of where you are. When you end the inside phrase, the starter set returns to the previous phrase.

Nesting makes your markup easier when you do complicated things with highlighted phrases: programming syntax examples, for instance.

For example, to get this result:

```
_____ Formatting Example _____  
COMMAND KEYWORD = 'DEFAULT|value' KEYWORD  
_____ End of Formatting Example _____
```

you would enter:

```
:p.:hp2.COMMAND KEYWORD=':hp3.DEFAULT:ehp3. |:hp1.value:ehp1.  
,  
KEYWORD:ehp2.
```

As you can see from the example, nesting highlighted phrases is less work than turning on and off the various highlight types.

Nesting is the reason we have highlight 0 (HP0). Sometimes, in a complicated structure, you might want a phrase with no emphasis. Highlight 0 produces a phrase with no emphasis without having to end all the highlights you have going and then restart them.

### Highlighting for Various Devices

The following paragraphs describe the highlighting available on the various devices for which SCRIPT/VS produces output.

Because the 1403 printer (and other printers that work like a 1403) and typewriter terminals do not have a bold font available, highlight 2 and 3 make the text stand out by overstriking the characters several times (highlight 3 also underscores the characters).

On display terminals, where neither overstriking nor bold is available, the starter set shows highlight 2 and 3 by forcing the text to uppercase (highlight 3 is also underscored).

For page printers and PostScript devices, where many fonts are available, highlighting is done with a change of fonts: italic, bold, and bold italic are used for highlight levels 1, 2, and 3, respectively.

---

### Title Citations

The tags for title citations (CIT and its end tag) are similar to the highlighting tags.

Use title citations when you refer to the title of a book; for example:

```
_____ Formatting Example _____  
Margaret Mitchell's book, Gone With the Wind, is my favorite book.  
_____ End of Formatting Example _____
```

The previous example would be marked up as follows:

```
:p.Margaret Mitchell's book,  
:cit.Gone With the Wind,:ecit.  
is my favorite book.
```

Because CIT and HP1 appear to produce the same result, you may wonder why we bother to have these extra tags. Perhaps this is a good place to consider again what GML is all about.

By creating precise markup, you are investing in the future of your text data base. Someday you might want to search your text data base for all title citations to ensure that you aren't citing obsolete titles. Using descriptive tags allows you to do this; the more accurately you describe the parts of a document, the more useful your data base becomes.

For instance, you should not mark text as an example unless it really is an example, even though today it happens to come out looking like an example. Much of the text being worked on today was created before DSMPROF3, and much more will be created that will still be around long after DSMPROF3 has been retired and newer profiles have taken its place. If you cheat in your markup now, you may have to rework the data base later, and none of us wants to do that.

## Notes

The starter set has a tag called NOTE.

A note would be marked up as follows:

```
:note.The text of a note is an implied paragraph.  
Notes can be used anywhere that paragraphs can be used.
```

The note would format like this:

Formatting Example

**Note:** The text of a note is an implied paragraph. Notes can be used anywhere that paragraphs can be used.

End of Formatting Example

A note is implicitly ended by a paragraph or a higher-level element.

## Quotations – the Long and Short of It

The starter set provides for two different kinds of quotations: inline quotations and excerpts (which we call *long quotations*).

### Inline Quotations

An inline quotation looks like this:

```
_____ Formatting Example _____  
  
FDR said, "The only thing we have to fear is fear itself."  
  
_____ End of Formatting Example _____
```

The markup for that (using the Q (quoted phrase) tag and its end tag) looks like this:

```
:p.FDR said, :q.The only thing we have  
to fear is fear itself.:eq.
```

A first-level quote takes double quotation marks; a nested quote within it takes single quotation marks; a nested quote within that takes doubles again, and so forth.

You don't have to decide whether a quote should take single or double quotation marks – DSMPROF3 knows!

You may well ask, "Why use quote tags at all?"

We now have a variety of output devices on which to print our documents. Some of these devices print using fonts that distinguish among open quotation marks, close quotation marks, and apostrophes. By using the correct markup, we can print our files on any of the available devices with no change to the markup.

### Excerpts (Long Quotations)

An excerpt (long quotation) looks like this:

```
_____ Formatting Example _____  
  
Please distribute the following shipping guidelines to those people in your area  
who are responsible for handling outgoing packages.  
  
    Packages weighing more than 20 pounds must arrive in the mailroom  
    before 2:30 P.M. if they are to be shipped the same day. If they are to be  
    shipped air freight, they require an authorization signed by a manager.  
  
    Packages that are to be shipped overseas must be submitted to the  
    mailroom with the correct customs-clearance forms filled out.  
  
Failure to comply with these guidelines could result in unacceptable delays in  
mailing packages.  
  
_____ End of Formatting Example _____
```

Notice that the starter set has indented the excerpt on both the right and the left, according to traditional publishing style for handling information quoted from another source.

Here's what the markup (using LQ and its end tag) looks like:

**:p.**Please distribute the following shipping guidelines to those people in your area who are responsible for handling outgoing packages:

**:lq.**

**:p.** Packages weighing more than 20 pounds must arrive in the mailroom before 2:30 P.M. if they are to be shipped the same day. If they are to be shipped air freight, they require an authorization signed by a manager.

**:p.** Packages that are to be shipped overseas must be submitted to the mailroom with the correct customs-clearance forms filled out.

**:elq.**

**:p.** Failure to comply with these guidelines could result in unacceptable delays in mailing packages.

Notice that the LQ tag does *not* imply a paragraph. You must use a P tag if the first sentence in your excerpt begins a paragraph. Of course, the sentence could be something else, such as a list.

---

### Exercise: Highlighting, Citing, Noting, and Quoting

Now try creating a document that looks like the following. Call this one EXER4. You'll find the markup in "Markup for Highlighting, Citing, Noting, and Quoting" on page 172.

---

#### Exercise

---

Print **boldly** on the 3800, **strikingly** on the 1403.

Title references are *also* used for movie titles, such as *Casablanca*. Humphrey Bogart is often misquoted as having said "Play it again, Sam" in that movie. I wish people would learn to quote **accurately**.

If I said, "he said 'yes' to me," that would be an example of a nested quote.

Look at this excerpt from the *GML User's Guide*:

You may well ask, "Why use quote tags at all?"

We now have a variety of output devices on which to print our documents. Some of these devices print using fonts that distinguish among open quotation marks, close quotation marks, and apostrophes. By using the correct markup, our files can be printed on any of the available devices with no change to the markup.

**Note:** Notes lose their impact if used excessively.

---

#### End of Exercise

---

To see the results at your terminal (in single-column format rather than the *offset style* we show you here):

**TSO users enter:**<sup>14</sup>

```
script exer4 prof(dsmprof3) cont
```

EXER4 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**CMS users enter:**<sup>14</sup>

```
script exer4 (prof(dsmprof3) cont
```

EXER4 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**ATMS-III users enter:**<sup>14</sup>

```
script * prof(dsmprof3) cont
```

The document to be formatted (EXER4—referred to by the \* in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

---

<sup>14</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.

## Chapter 7. Tables

Like examples and figures, tables allow you to organize information and present it in a format that can improve both the understanding and the usability of the information.

When you build a table, you use three basic elements:

- Cell** A cell is rectangular and is usually separated from other cells by horizontal and vertical rules.
- Row** A row is a horizontal, rectangular collection of one or more cells. The cells that make up a row may have different widths and depths.
- Table** A table is a collection of one or more rows. The rows that make up a table may contain different cell arrangements.

The following example shows a table with two rows, each using the same cell arrangement of three cells per row:

This is the first cell in the first row of this table.	This is the second cell in the first row of this table.	This is the third cell.
This is the first cell in the second row of this table.	This is the second cell in the second row of this table.	This is the third cell in the second row.

The attributes on the table tags provide flexibility in specifying different characteristics of the table. For example, you can specify the highlight level to be used in a cell and the vertical alignment of the contents of the cell.

We will show you how to add captions and descriptions to your tables.

You can specify a particular row, called a *header*, to appear at the top of every column on every page on which the table appears. Likewise, you can specify a particular row, called a *footer*, to appear at the bottom of every column on every page on which the table appears.

---

### Defining the Table

Before you can create a table, you need to define its *attributes* using the RDEF tag. These attributes do the following:

- Provide a name for the row definition: ID (this is the only required attribute)
- Specify the number, arrangement, and size of the cells in the row: CWIDTHS and ARRANGE
- Specify the characteristics of the cells in the row: HP, ALIGN, CONCAT, VALIGN, ROTATE, and MINDEPTH.

For this last set of attributes, more than one value can be specified. The first value specified applies to the first cell, the second value to the second cell, and so on. If fewer values are specified than the number of cells, the remaining cells use the last value specified. If a greater number of values



are specified than the number of cells, the extra values are ignored. If multiple values are given, the entire string of values must be enclosed in single quotation marks. This technique is shown in examples later in this chapter.

The values for all of these attributes (except ROTATE and MINDEPTH)<sup>15</sup> can be abbreviated. For example, on the ALIGN attribute, the values "LEFT," "LEF," "LE," and "L" are all valid ways to specify left alignment for the contents of the cell.

The following shows the attributes used on the RDEF tag:

```
:RDEF ID=name
      HP=highlight level
      ALIGN=horizontal alignment
      CONCAT=YES|NO
      VALIGN=vertical alignment
      ROTATE=rotation
      MINDEPTH=vertical space
      ARRANGE=arrangement of cells
      CWIDTHS=horizontal space.
```

where:

**ID**=name

is a required attribute that allows you to refer to this row definition when starting a table, row, header, or footer. This identifier is not case sensitive and can be a mixture of letters and numbers, up to seven characters in length.

**HP**=highlight level

specifies the highlight level to be used for the text in the cells of the table. The valid values are 0, 1, 2, and 3.

The four values correspond directly to the four levels of highlighting available with the HP tag.

This attribute is optional, and the default is 0. For more information about highlighting, see Chapter 6, "Highlighting, Citing, Noting, and Quoting" on page 45.

**ALIGN**=horizontal alignment

determines the horizontal alignment of the contents of your cells. This attribute is optional, and the default is LEFT. You can specify any of six choices for horizontal alignment:

**LEFT** Aligns the cell contents with the left-hand side of the cell.

**CENTER** Aligns the cell contents in the middle of the cell.

**RIGHT** Aligns the cell contents with the right-hand side of the cell.

---

<sup>15</sup> The ROTATE and MINDEPTH attributes use numbers and vertical space unit notations as values, so these values cannot be abbreviated.

**INSIDE** On odd-numbered pages, aligns the cell contents with the left-hand side of the cell. On even-numbered pages, aligns the cell contents with the right-hand side of the cell.

**OUTSIDE** On odd-numbered pages, aligns the cell contents with the right-hand side of the cell. On even-numbered pages, aligns the cell contents with the left-hand side of the cell.

**JUSTIFY** Inserts additional white space between words in the cell, producing even left and right edges.

**CONCAT**=YES|NO

determines whether the input lines will be concatenated when producing output lines. This attribute is optional, and the default is YES.

**YES** Specifies that the input lines will be concatenated to produce output lines that are as full as possible.

**NO** Specifies that the printed output lines will match the way they were entered.

**VALIGN**=vertical alignment

determines the vertical alignment of the contents of the cells. This attribute is optional, and the default is TOP. You have three choices for vertical alignment:

**TOP** Indicates the contents of the cell are to be placed at the top of the cell

**CENTER** Indicates the contents of the cell are to be centered vertically in the cell

**BOTTOM** Indicates the contents of the cell are to be placed at the bottom of the cell.

**ROTATE**=rotation

specifies the rotation of the contents of the cell. The valid values are 0, 90, -270, 180, -180, 270, and -90. This attribute is optional, and the default is 0.

**Note:** This attribute is valid only for devices that allow rotation. A composite rotation of 180° or -180° is not valid for the IBM 3800 Printing Subsystem Model 3 or Model 6.

If you specify a rotation of 90, -270, 270, or -90, the MINDEPTH attribute is required for that cell. See the description of MINDEPTH for more information.

This same attribute is used on the TABLE tag to specify rotation of the entire table.

**MINDEPTH**=vertical space

specifies the minimum depth of a cell. If the content of a cell does not fill the specified minimum depth, white space is added automatically at the top or bottom of the cell (depending on the VALIGN value for that cell) to make the cell the specified minimum depth. This attribute is optional for cells rotated 0°, 180°, or -180°. If this attribute is not specified for a particular cell, or if an

asterisk (\*) is specified, the contents of the cell and the depth of other cells in the row determine the depth of the cell.

If you specify a rotation value of 90, -270, 270, or -90 for a cell, then the MINDEPTH attribute is required for that cell. That's because the MINDEPTH value is used as the column line length for the contents of the cell. The actual depth of such a cell may be greater than the MINDEPTH value specified for that cell, depending on the depth of other cells in the row. However, the content of the cell is still formatted using the MINDEPTH value. If this produces undesirable formatting results, increase the MINDEPTH value for the cell.

**ARRANGE** = arrangement of cells

specifies the arrangement of the cells in a row. You must enter positive integers as values on this attribute. The values you enter correspond to the cell numbers used with the C tag. This attribute is optional, and the default is *n* cells, where *n* is the number of values given on the CWIDTHS attribute. If no CWIDTHS attribute is specified, the default is one cell.

See "Using the ARRANGE and CWIDTHS Attributes" on page 66 for a method of determining the ARRANGE and CWIDTHS values for a cell arrangement.

**Note:** Although the ARRANGE definition may have been defined using a separate input line for each line of the cell arrangement, the entire ARRANGE definition for a given row is processed as one input line. If the entire input line exceeds the SCRIPT/VS input line limit of 256 characters, it will result in SCRIPT error messages.

**CWIDTHS** = horizontal space

is used to determine the widths of the individual cells. Use any of the standard horizontal space units. (See "Space Units" on page 39 for the allowable space unit notations.)

If the ARRANGE attribute is not used, the CWIDTHS attribute determines the number of cells in the row and the width of each cell. There will be one cell for each value specified on the CWIDTHS attribute. The values specified in the CWIDTHS attribute are used as the widths of the cells.

**Note:** If you use the ARRANGE attribute, the number of values specified on the CWIDTHS attribute must be one of the following:

- The number of values given between slashes
- The number of values on one ARRANGE attribute (if slashes are not used).

The CWIDTHS values indicate the widths of the individual rectangles in the grid as defined with the ARRANGE attribute. The width of a particular cell is then the sum of the CWIDTHS values for all the grid rectangles that make up the particular cell. See "Using the ARRANGE and CWIDTHS Attributes" on page 66 for a method of determining the ARRANGE and CWIDTHS values for a cell arrangement.

The asterisk (\*) can also be used as an attribute value. This notation indicates that any remaining horizontal space be divided equally among the number of asterisks you enter on that particular attribute. A number can

precede the asterisk to indicate a factor to apply. For example, a cell with a width of "2\*" is twice as wide as a cell with a width of "\*."

If the CWIDTHS attribute is not used, the default is *n* cells of equal width, where *n* is the number of values given between one set of slashes on the ARRANGE attribute or on a single ARRANGE attribute if slashes are not used.

---

## Building the Table

Once you define the rows in your table, use the TABLE tag to start the table. Use the ROW tag and the C tag to begin the rows and cells of your table. Use the TABLE end tag to end your table.

The TABLE tag has seven attributes:

```
:TABLE REFID=name
      ID=name
      WIDTH=horizontal space
      COLUMN
      PAGE
      SPLIT=NO|YES
      ROTATE=rotation.
```

where:

**REFID** = name

is a required attribute that refers to a unique row definition previously defined with the RDEF tag. It is not case sensitive.

**ID** = name

is the attribute that allows you to make cross-references to this table with the TREF tag. (Cross-referencing is described in Chapter 8, "Cross-References and Footnotes" on page 73.) This identifier can be a mixture of letters and numbers up to seven characters in length. The identifier is case sensitive.

**WIDTH** = horizontal space

specifies the width of the table. The value specified can be any of the horizontal space units described in "Space Units" on page 39. If this attribute is not specified, the width of the table is determined by the COLUMN or PAGE attribute.

**COLUMN**

sets the width of the table to the current column line length. The COLUMN and PAGE attributes are mutually exclusive. The last one specified is used.

**PAGE**

sets the width of the table to the current line length. The COLUMN and PAGE attributes are mutually exclusive. The last one specified is used. PAGE is the default.

**SPLIT** = NO|YES

specifies whether a table should be split between rows when the table is too large to fit into a single column. You can also specify the SPLIT attribute on the ROW tag. See "Starting a Row" on page 56 for more information.

**NO** Specifies that if the table does not fit into the current column, the table cannot be split, except prior to rows that have SPLIT=YES specified on the ROW tag. The default is NO.

**YES** Specifies that if the table does not fit into the current column, the table can be split prior to any row, except rows that have specified SPLIT=NO on the ROW tag.

If a table does not fit on the current page and if no rows can be split, the entire table is moved to the next page. If the table does not fit on the next page, a warning message is issued and the table is split, fitting as many rows on the current page as possible.

### **ROTATE**=rotation

specifies a rotation for the entire table. The valid values are 0, 90, -270, 180, -180, 270, and -90. This attribute is optional and is valid only for devices that allow rotation. The default is 0.

A table does not rotate if the COLUMN attribute has been specified. A composite rotation of 180 or of -180 is not valid for the IBM 3800 Printing Subsystem Model 3 or Model 6.

## Starting a Row

Start each row of a table with a ROW tag. The cells in each row are formatted using the row definition values you entered previously on the RDEF tag.

A ROW end tag, which is optional, can be used to end a row. Another row, a table note, a table caption, a table description, or a TABLE end tag also cause the current row to end.

The ROW tag has two attributes:

```
:ROW REFID=name  
      SPLIT=NO|YES.
```

where:

### **REFID**= name

refers to a unique row definition previously defined with the RDEF tag. This attribute is optional and is not case sensitive. If not specified, the row definition from the previous row (or the TABLE tag if this is the first row in the table) is used.

### **SPLIT**=NO|YES

specifies whether the table should be split just ahead of this row.

**NO** Specifies that if the table does not fit into the current column, the table should not be split prior to this row. The default is NO.

**YES** Specifies that if the table does not fit into the current column, the table should be split prior to this row.

If the **SPLIT** attribute is not specified, the **SPLIT** value used on the **TABLE** tag determines whether the table can be split prior to this row.

If a table does not fit on the current page and if no rows can be split, the entire table is moved to the next page. If the table does not fit on the next page, a warning message is issued and the table is split, fitting as many rows on the current page as possible.

## Starting a Cell

Start each cell in a row of a table with a **C** tag, followed by the text you want placed in that cell. The cell contents can continue for more than one input line. The **C** end tag, which is optional, can be used to end a cell. A cell can also be ended by another cell tag or another row, a table note, table caption, table description, row end tag, table end tag, table header end tag, or table footer end tag.

The **C** tag has one attribute:

**:C** *n*

where:

*n*

is the *number* of the cell to be started.

This attribute is optional. If a cell number is not specified, the cell with the next highest cell number in the row is started. If no such cell exists, the current row is ended, a new row is started using the same row definition, and the cell with the lowest cell number in that row is started.

If you attempt to place document elements (text or page segments) outside a cell, the starter set ends the table and issues an error message.

## Using a Table Header

The **THD** tag defines a header for a table. A table header is a special row placed at the top of the table in all columns or pages on which the table appears. You must use the **THD** end tag to end the table header definition, and it must begin in column one of an input line.

The **THD** tag has one optional attribute:

**:THD** **REFID**=name.

where:

**REFID**=name

refers to a unique row definition previously defined with the **RDEF** tag. This attribute is optional. If not specified, the row definition from the previous row (or the **TABLE** tag if this is the first row in the table) is used. This attribute is not case sensitive.

The **THD** tag is valid only within a table. You must define the table header before you start the first row in the table. Use the **C** tag to fill the cells in your table header, just as you did to fill the cells in rows.

### Using a Table Footer

The TFT tag defines a footer for a table. A table footer is a special row placed at the bottom of the table in all columns or pages on which the table appears. You must use the TFT end tag to end the table footer definition, and it must begin in column one of an input line.

The TFT tag has one optional attribute:

**:TFT REFID=name.**

where:

**REFID = name**

refers to a unique row definition previously defined with the RDEF tag. This attribute is optional. If not specified, the row definition from the previous row (or the TABLE tag if this is the first row in the table) is used. This attribute is not case sensitive.

The TFT tag is valid only within a table. You must define the table footer before you start the first row in the table. Use the C tag to fill the cells in your table footer, just as you do to fill the cells in rows.

### Using a Table Caption and a Table Description

If you want a caption for your table, use the TCAP tag at the end of your table, prior to the TABLE end tag. Your table must have a caption for the caption to appear in the list of tables created with the TLIST tag. All the text of the caption must appear either on the same line as the TCAP tag or on the next line. There must be no GML tags in the caption.

You do not have to enter the word "Table" or the table number. The starter set takes care of that for you.

If you want to describe the table, you can also have a table description, which can go on for as many lines as you need. The table description text is not placed in the list of tables but is added to the table caption at the bottom of the table.

Place the table description (TDESC) tag immediately after the TCAP tag and before the TABLE end tag.

The TCAP tag:

- Numbers the table automatically
- Formats the text that was included with the TCAP tag
- Places the table number and text from the TCAP tag into the list of tables.

The TDESC tag:

- Places a colon (:) after the text of the table caption
- Follows the colon with the text of the TDESC tag.

**Note:** Because the table caption and description are not actually part of the table, they might be moved to a different column or page if the table itself is

large enough to fill the current column. Also, if the table is rotated, the table caption and description are not rotated.

### Using a Table Note

Use the TNOTE tag to insert a note in a table. The TNOTE tag starts a new row, consisting of only one cell that is as wide as the table. The word: "**Note:**" in a level-two highlighted phrase (HP2) precedes the note text you enter.

Use the TNOTE end tag to end a table note. The TNOTE end tag also causes the row to be ended, so you must specify another ROW tag to start a new row.



## Sample Tables

On the following pages we show a number of sample tables illustrating various attributes of the table tags. Examples of tables with different row definitions, text highlighting, text alignments, and cell rotations are shown. We also include tables that illustrate table notes, table captions, and table descriptions.

### Creating a Three-Row Table with Text in the Cells

Here is an example of a simple table consisting of three rows, all using the same row definition. The row definition does not need an `ARRANGE` attribute, because all of the cells in the row extend from the top to the bottom of the row. The `COLUMN` attribute is used on the `TABLE` tag to cause the table to be as wide as the current column, not the current page.

Here's cell 1.	This text is in cell 2.	Cell 3.
Here's the first cell in the second row.	The arrangement of this row is the same as the arrangement of the previous row and the next row.	The depth of the row is determined by the depth of the cell in that row with the most text, so this row ends up being deeper than the previous and the next rows.
This is the third row.		Cell 2 in this row is empty.

Here are the tags used to define and create the previous table:

```

:rdef id=exmpone cwidths='1.8i 1.8i *'.
:table column refid=exmpone.
:row.
:c.Here's the cell 1.
:c.This text is in cell 2.
:c.Cell 3
:row.
:c.Here's the first cell in the second row.
:c.The arrangement of this row is the
same as the arrangement of the previous row and
the next row.
:c.The depth of the row is determined by the depth of the
cell in that row with the most text, so this row ends up being
deeper than the previous and the next rows.
:row.
:c.This is the third row.
:c 3.Cell 2 in this row is empty.
:etable.

```

## Creating a Table with Two Different Row Definitions

The following table consists of four rows. The first three rows use one row definition, and the fourth row uses a different row definition. We did not specify either the COLUMN or WIDTH attributes, so the current line length determines the width of the table.

Here's cell 1.	This text is in cell 2.	Cell 3.
Here's the first cell in the second row.	The arrangement of this row is the same as the arrangement of the previous row and the next row.	The depth of the row is determined by the depths of the cells, so this row ends up being deeper than the previous and the next rows.
This is the third row.		Cell 2 in this row is empty.
This row has a different arrangement from the arrangement of the previous row.	This is the second cell.	This is the third cell. The fourth cell is empty.

Here are the tags used to define and create the previous table:

```

:rdef id=exmpone cwidths='2i 2i *'.
:rdef id=exmptwo cwidths='1i * * *'.
:table refid=exmpone.
:row.
:c.Here's cell 1.
:c.This text is in cell 2.
:c.Cell 3.
:row.
:c.Here's the first cell in the second row.
:c.The arrangement of this row is the same
as the arrangement of the previous row and the next row.
:c.The depth of the row is determined by the depths
of the cells, so this row ends up being deeper
than the previous and the next rows.
:row.
:c.This is the third row.
:c 3.Cell 2 in this row is empty.
:row refid=exmptwo.
:c.
This row has a different arrangement from the arrangement
of the previous row.
:c.This is the second cell.
:c.This is the third cell.
The fourth cell is empty.
:etable.

```

## Changing the Highlight Level of the Cells

The highlight level used for the text in a cell can be specified by using the HP attribute on the RDEF tag.

This is the first cell; it uses highlight level 0. This is the same as specifying the HP0 tag.	<i>This cell is defined to be in highlight level 1.</i>	<b>This cell is defined to be in highlight level 2.</b>	<b>The HP attribute contains only 3 specifications, so this cell, the fourth cell, uses the highlight level used in the previous cell.</b>
--	---	---	--

Here are the tags used to define and create this table:

```

:rdef id=more cwidths='1i * * *'
      hp='0 1 2'.
:table refid=more column.
:row.
:c.This is the first cell; it uses highlight level 0.
This is the same as specifying the HP0 tag.
:c.This cell is defined to be in highlight level 1.
:c.This cell is defined to be in highlight level 2.
:c.The HP attribute contains only 3 specifications, so this
cell, the fourth cell, uses the highlight level used in the
previous cell.
:etable.

```

## Using the ALIGN, VALIGN, and CONCAT Attributes

Other cell characteristics you can specify on the RDEF tag are: horizontal alignment (ALIGN attribute), vertical alignment (VALIGN attribute), and concatenation (CONCAT attribute). The following table uses all three of these attributes:

<p>This cell uses highlight level 0. Concatenation is on in this cell. All the lines are left aligned, and the cell contents are aligned to the top of the cell.</p>	<p><i>Highlight level 1; text lines centered and not concatenated; bottom vertical alignment is used.</i></p>	<p><b>This cell uses highlight level 2. Concatenation is on in this cell. All the lines are right aligned, and the cell contents are aligned to the top of the cell.</b></p>	<p>The HP attribute contains 3 values, so the fourth cell uses the highlight level from the previous cell. Input lines are joined, and space is added between words to make the left and right edges even. The text is centered vertically.</p>
--	---	--	---

Here are the tags used to define and create the previous table:

```

:rdef id=more cwidths='1i * * *'
      hp='0 1 2' align='l c r jus'
      valign='top bottom top center' concat='yes no yes'.
:table refid=more column.
:row.

:c.This cell uses highlight level 0.
Concatenation is on in this cell. All the lines are left
aligned, and the cell contents are aligned to the top
of the cell.
:c.Highlight
level 1;
text lines
centered and
not
concatenated;
bottom vertical
alignment is used.
:c.This cell uses highlight level 2.
Concatenation is on in this cell. All the lines are right
aligned, and the cell contents are aligned to the top
of the cell.
:c.The HP attribute contains 3 values, so
the fourth cell uses the highlight level from the
previous cell.
Input lines are joined, and space is added between words to
make the left and right edges even.
The text is centered vertically.
:etable.

```

If the NO value on the CONCAT attribute is specified for a cell, the normal formatting of text (justification and concatenation) is suspended within the cell; each input line becomes a separate output line. Other GML tags should be used with caution within the cell, because concatenation of input lines is suppressed. Unexpected line breaks or concatenation of source text lines may occur when GML tags are used within cells.

## Using the ROTATE and MINDEPTH Attributes

When formatting for the 3800 Print Subsystem Model 3 and Model 6, the 3820 Page Printer, and the PostScript devices, you can rotate cells within a table by using the ROTATE attribute on the RDEF tag. When specifying a rotation of 90° or 270°, you must also specify a minimum depth with the MINDEPTH attribute. If you specify a cell rotation of 90, -270, 270, or -90, the MINDEPTH attribute determines the width of the formatted text within the cell. Just for fun, the cell number attribute is used on the C tags to fill the cells in a nonsequential order.

<p>This cell is rotated 90°. The MINDEPTH value is 3 inches, so the depth of the cell is 3 inches, which in this case is the width of the text in the cell. The text is centered vertically in the cell.</p>	<p>This cell is rotated 0°. The text is centered vertically in the cell.</p>	<p>This cell is rotated 180°. The text is centered vertically in the cell.</p>	<p>This cell is rotated 270°. The width of the text in the cell is 3 inches. The text is centered vertically in the cell.</p>
--	--	--	---

Here are the tags used to define and create the previous table:

```

:rdef id=bbob rotate='90 0 180 270'
      mindepth='3i * * 3i'
      valign=center
      cwidths='* * * *'.
:table column refid=bbob.
:row.
:c. This cell is rotated 90°.
The MINDEPTH value is 3 inches,
so the depth of the cell is 3 inches,
which in this case is the width of the text in the cell.
The text is centered vertically in the cell.
:c 3. This cell is rotated 180°.
The text is centered vertically in the cell.
:c 2. This cell is rotated 0°.
The text is centered vertically in the cell.
:c 4. This cell is rotated 270°. The width of the text in
the cell is 3 inches.
The text is centered vertically in the cell.
:etable.
    
```

## Creating a Table Note, Caption, and Description

In the following table the TNOTE, TCAP, and TDESC tags were used to create a table note, caption, and description:

This is the first cell; the second and third cells are empty.		
<b>Note:</b> This is a table note. It is placed in a row containing one cell that is as wide as the table.		
		This is the third cell in this row. (The first and second cells are empty.)

**Table 1. Example of a Table Note, Caption, and Description.** This is a rather short example. Probably it does not represent a table you might create for a real purpose.

Here are the tags we used to define and create the previous table:

```
:rdef id=geef cwidths='3* 2* *'
:table refid=geef.
:row.
:c.This is the first cell; the second and third
cells are empty.
:tnote.This is a table note. It is placed in a row
containing one cell that is as wide as the table.
:etnote.
:row.
:c 3.This is the third cell in this row. (The first and second
cells are empty.)
:tcap.Example of a Table Note, Caption, and Description
:tdesc.This is a rather short example. Probably
it does not represent a table you might create for
a real purpose.
:etable.
```

### Using the ARRANGE and CWIDTHS Attributes

In the preceding examples, we didn't use the ARRANGE attribute. If all the cells in a row extend from the top of the row to the bottom of the row, the ARRANGE attribute is not necessary, because the CWIDTHS attribute determines the width of each cell. The cells are numbered from left to right, starting with cell number 1. The ARRANGE attribute is necessary when you define a row that contains at least one cell that does not extend from the top of the row to the bottom of the row. Following is the simplest example of such a row:


Consider this table.

This cell is as wide as the two cells below it.	This cell extends from the top to the bottom of the row and is the only cell to do so in this table.	An- other cell	An- other cell
An- other cell	An- other cell	What more can I say?	
This cell is as wide as the upper-leftmost cell in this table.			

Determining the correct ARRANGE attribute values to create the previous table can be tricky. The following steps should make it a little easier.

#### Step 1. Sketch the Row

Sketch your row and number each cell using positive integers. You do not have to number the cells sequentially, but the process is easier if you do.

1		6	7
2	3	5	
4		8	

#### Step 2. Form a Grid

Extend all the vertical and horizontal rules to the edges of the sketch to form a grid.


The extensions are shown in thin rules, and the cells from Step 1 are shown in thick rules.

**Step 3. Determine the Widths of the Cells**

Determine the horizontal width of each rectangle in the grid from Step 2. These are the values to use on the CWIDTHS attribute.

3p	3p	12p	3p	3p

The CWIDTHS attribute for this row would be:

```
cwidths='3p 3p 12p 3p 3p'
```

The CWIDTHS values add up to 24 picas, so the following WIDTH attribute should be specified on the TABLE tag:

```
width=24p
```

**Step 4. Number Each Box**

Number each rectangle in the grid with the number of the cell from Step 1. In the first line of the grid, place a '1' in the first and second rectangles (because they make up cell 1 in Step 1). Your grid should look like this:

1	1	5	6	7
2	3	5	8	8
4	4	5	8	8

In the previous example the numbers from 1 to 8 were used in sequential order. Remember, you can use any integers, in any order, as long as the cell composed of grid elements with the same number forms a rectangle.

**Step 5. List the Numbers**

List the numbers as they appear in the grid elements from Step 4:

```
1 1 5 6 7
2 3 5 8 8
4 4 5 8 8
```

Then, use these numbers with the ARRANGE attribute:

```
arrange='1 1 5 6 7'
arrange='2 3 5 8 8'
arrange='4 4 5 8 8'
```

or

```
arrange='1 1 5 6 7 / 2 3 5 8 8 / 4 4 5 8 8'
```

**Step 6. Enter the Row Definition**

The complete row definition looks like this:

```
:rdef id=myrow arrange='1 1 5 6 7'
           arrange='2 3 5 8 8'
           arrange='4 4 5 8 8'
           cwidths='3p 3p 12p 3p 3p'.
```



## Using the ARRANGE Attribute

Following is an example of a table requiring the ARRANGE attribute:

Selected major industry and selected country	Number of U.S. corporation returns	Controlled foreign corporations			
		Number of foreign corporations	Total assets of corp.	Foreign corporations with earnings and profits (+) before taxes	
				Earnings and profits before taxes	Foreign income taxes (net)

**Table 2. Example Table Showing the ARRANGE Attribute.**

Here are the tags used to define and create this table:

```

:rdef id=corptbl cwidths='1j * * * * *'
      arrange='1 2 3 3 3 3'
      arrange='1 2 4 5 6 6'
      arrange='1 2 4 5 7 8'
      align='left left center center center left center'
      concat='yes yes no yes'
      valign='center center top center center center top bottom'.
:table refid=corptbl column.
:row.
:c.Selected major industry and selected country
:c.Number of U.S. corporation returns
:c.Controlled foreign corporations
:c.Number of foreign corporations
:c.Total assets of corp.
:c.Foreign corporations with earnings and
profits (+) before taxes
:c.Earnings and
profits before taxes
:c.Foreign income taxes (net)
:tcap.Example Table Showing the ARRANGE Attribute.
:etable.
    
```

## Creating a Table Header

A table header is an effective way of defining a row that describes the contents of the cells in the rows below it. Table 3 on page 69 also uses the TCAP and TDESC tags to create a table caption and description at the end of the table:

Type of Offense	Number of Offenders	Number of Offenses	Distance from Offender's Residence				
			Less than 1 Block	1 to 2 Blocks	2 to 3 Blocks	3 to 4 Blocks	Unknown
Trespassing	02	03	40	20			
Carrying Weapons	05	04	29	30			
Intoxication, Disorderly Conduct, Open Container, Creating a Disturbance	07		10		40		30
Subtotal	14	07	79	50	40		30

**Table 3. Example Table Showing a Table Header.** This table indicates the correlation between different types of criminal offenses and the distance from the offender's residence to the scene of the offense.

Here are the tags used to define and create the previous table:

```

:rdef id=hdr cwidths='1.5i' concat=n hp=2
      arrange='1 2 3 4 4 4 4 / 1 2 3 5 6 7 8 9'
      align='center left left center left' valign=center
      rotate='0 270 270 0 270' mindepth='* 2i 2i * 1.5i'.
:rdef id=body cwidths='1.5i * * * * *'
      hp='2 0' align='left center'.
:table refid=body column.
:thd refid=hdr.
:c.Type of Offense:c.Number of Offenders
:c.Number of Offenses:c.Distance from Offender's Residence
:c.Less than 1 Block:c.1 to 2 Blocks
:c.2 to 3 Blocks:c.3 to 4 Blocks:c.Unknown
:ethd.
:row refid=body.
:c.Trespassing:c.02:c.03:c.40:c.20
:row.
:c.Carrying Weapons:c.05:c.04:c.29:c.30
:row.
:c.Intoxication, Disorderly Conduct, Open Container, Creating a Disturbance
:c.07:c 4.10:c 6.40:c 8.30
:row.
:c.Subtotal
:c.14:c.07:c.79:c.50
:c.40:c 8.30
:tcap.Example Table Showing a Table Header
:tdesc.This table indicates the correlation between different types of
criminal offenses and the distance from the offender's residence to the scene of
the offense.
:etable.

```

---

## Exercises: Tables

On the next few pages are two exercises for you to practice, using the starter set table tags.

### First Exercise

See if you can create a table that looks like the one below (call the file EXER5). The markup is shown in "First Table Exercise" on page 173.

Year	Passenger Cars	Trucks and Buses	Total
1900	4,192		4,192
1905	24,250	750	25,000
1910	181,000	6,000	187,000
1915	895,930	74,000	969,930
1920	1,905,560	321,789	2,227,349

**Table 4. Example Table Showing U. S. Automobile Production.** The above table shows the number of automobiles produced in the United States from the years 1900 through 1920.

To see the results back at your terminal (in single-column format rather than the *offset style* we show you here):

**TSO users enter:**<sup>16</sup>

```
script exer5 prof(dsmprof3) cont
```

EXER5 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**CMS users enter:**<sup>16</sup>

```
script exer5 (prof(dsmprof3) cont
```

EXER5 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**ATMS-III users enter:**<sup>16</sup>

```
script * prof(dsmprof3) cont
```

The document to be formatted (EXER5—referred to by the \* in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

---

<sup>16</sup> Check with your supervisor or your document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.

**Second Exercise**

Now create another table. Call your file EXER6. The markup is shown in "Second Table Exercise" on page 173.

Flag	Connecticut
	Delaware
	Georgia
	Maryland
	Massachusetts
	New Hampshire
	New Jersey
	New York
	North Carolina
	Pennsylvania
Rhode Island	
South Carolina	
Virginia	

To see the results at your terminal (in single-column format rather than the *offset style* we show you here):

**TSO users enter:**<sup>17</sup>

```
script exer6 prof(dsmprof3) cont
```

EXER6 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**CMS users enter:**<sup>17</sup>

```
script exer6 (prof(dsmprof3) cont
```

EXER6 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**ATMS-III users enter:**<sup>17</sup>

```
script * prof(dsmprof3) cont
```

The document to be formatted (EXER6 – referred to by the \* in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

<sup>17</sup> Check with your supervisor or your document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.



---

## Chapter 8. Cross-References and Footnotes

The ID attribute assigns a name to sections of information (or document elements) so that the identified information can be referred to in another part of the document.

---

### To Headings

You've already seen cases in this book of cross-references to headings, although you may not have realized it at the time. For example, this is a cross-reference:

```
_____ Formatting Example _____
```

See Chapter 4, "Lists" on page 17.

```
_____ End of Formatting Example _____
```

To get the cross-reference required two steps. The first step was to enter the heading that we referred to as follows:

```
:h1 id=clst.All Kinds of Lists
```

The "clst" is a name we made up; it could be any combination of letters and numbers as long as it contains no more than 7 characters and begins with a letter; it is a good idea to use lowercase letters. (Of course, you couldn't have another heading with the same name, because the starter set wouldn't know which one you were referring to.)

The second step was to enter the cross-reference itself using the HDREF (heading reference) tag with a REFID attribute. It looks like this:

```
:p.See :hdref refid=clst..
```

Notice the two periods at the end of the example; the first one ends the markup and the second one ends the sentence. A common error is to forget that you need two periods at that point.

Because we used the same value for the REFID attribute as for the ID attribute of the heading we are referring to, the starter set knows which one we want. Because it also knows what page it is on, it supplies that, too. If the heading had been on the same page as the reference, the starter set would know that also and would not give the page number.

You can use HDREFs to refer to any level heading as long as the heading you refer to is given an ID attribute when it's entered.

Picking descriptive names like *summary* and *intro* for your ID attributes is a good idea. Then when you insert, delete, and replace material, you will find that descriptive names help you keep track of what is going on.

Sometimes you might want to omit the page reference, even though the heading is on a different page. The starter set lets you do this with the PAGE attribute on the HDREF tag. Your markup would look like this:

```
:p.See :href page=no refid=clst..
```

On the other hand, if you want to include the page reference (even though the reference is on the same page as the heading), you can enter the PAGE attribute with a value of "yes."

Information identified with the ID attribute can come before or after the cross-reference. However, if the information being referred to comes after the reference, you must specify two or more formatting passes, using the FPASSES or TWOPASS option of the SCRIPT command at run time. If you don't, the starter set fills in "-- Heading id 'nohead' unknown --" where you wanted the cross-reference. We'll tell you more about FPASSES and TWOPASS in Chapter 13, "Formatting Instructions" on page 119.

---

## To List Items

List item references are the same as heading, figure, and table references, except you use the list item reference (LIREF) tag.

We'll create a short, ordered list to demonstrate how the LIREF tag works.

---

### Formatting Example

Steps to follow when changing wallpaper:

1. Peel off the old wallpaper.
2. Clean and size the wall.
3. Prepare the paste.
4. Get help from someone who knows what to do.

---

End of Formatting Example

We've chosen an ordered list item to cross-reference, because that is the one you'd most likely refer to. You can refer to a list item from any of the lists, but because simple lists and unordered lists have no numbers, the reference wouldn't make sense.

Here is the markup for our previous example:

Steps to follow when changing wallpaper:

```
:ol.  
:li.Peel off the old wallpaper.  
:li.Clean and size the wall.  
:li.Prepare the paste.  
:li id=help.Get help from someone  
who knows what to do.  
:eol.
```

Now we can say:

```
:p.We'd better start with number
:liref refid=help. the next time.
```

and it would appear in our output like this:

Formatting Example

We'd better start with number 4. on page 74 the next time.

End of Formatting Example

LIREF allows you to use the PAGE attribute, just as HDREF does.

## To Figures

Figure references are just the same as head references, except that you use the figure reference (FIGREF) tag.

Because we haven't shown you a figure with an ID attribute, let's create one now:

Formatting Example

We're partial to figures in boxes.  
A figure that is going to have a cross-reference must also have a caption (FIGCAP).

**Figure 5. A Captioned Figure for Cross-Reference**

End of Formatting Example

The caption is needed because the figure number assigned through the FIGCAP tag is what is going to appear in the figure reference (the text of the caption is not used). If you don't have a FIGCAP tag, the cross-reference has nothing to refer to.

The only markup we are concerned with from our figure above is the FIG tag itself—you already know how to do the rest of it. The FIG tag looks like this:

```
:fig id=xreffig frame=box place=inline width=column.
```

and there you see the ID attribute.



Now we can say

`:p. See :figref refid=xreffig.` for an illustration of...

and it would come out like this:

Formatting Example

See Figure 5 on page 75 for an illustration of...

End of Formatting Example

Because our figure is not on the same page as the figure reference, the starter set includes a page number. However, if both had been on the *same* page, it would *not* have included a page number.

FIGREF also allows you to use the PAGE attribute, just as HDREF and LIREF do.

## To Tables

Table references are just the same as heading and figure references, except that you use the table reference (TREF) tag.

Now let's create a table with an ID attribute:

Selected major industry and selected country	Number of U.S. corporation returns	Controlled foreign corporations			
		Number of foreign corporations	Total assets of corp.	Foreign corporations with current earnings and profits (+) before taxes	
				Current earnings and profits before taxes	Foreign income taxes (net)

**Table 5. A Captioned Table, to Demonstrate Cross-Referencing**

The caption is needed because the table number assigned through the TCAP tag is what is going to appear in the table reference (the text of the caption is not used). If you don't have a TCAP tag, the cross-reference has nothing to refer to.

The only markup we are concerned with from our table above is the TABLE tag itself—you already know how to do the rest of it. The TABLE tag looks like this:

`:table refid=corptbl id=tbtest.`

and there you see the ID attribute.

We could now enter:

**:p.**See **:tref refid=tbtest.** for an illustration of...

and it would come out like this:

Formatting Example

See Table 4 on page 76 for an illustration of...

End of Formatting Example

Because our table is on a different page than the table reference, the starter set included a page number. However, if both had been on the *same* page, it would *not* have included a page number.

TREF also allows the PAGE attribute, just as HDREF, LIREF, and FIGREF do.

---

## To Footnotes

A footnote is a little different from the other document elements that you can cross-reference. You can have a cross-reference (callout) to a footnote without using the ID attribute. In fact, that is probably how you would mark up a footnote most often. When you don't use the ID attribute, the footnote gets referenced where it is defined.

For example:

Formatting Example

A footnote reference appears at the end of this sentence because that is where I put my footnote.<sup>18</sup>

End of Formatting Example

This is how it was entered:

**:p.**A footnote reference appears at the end of this sentence because that is where I put my footnote.

**:fn.**

This is the text of a footnote that I entered without using the ID attribute to give the footnote a name.

**:efn.**

Footnotes can contain paragraphs, lists, highlighted phrases, and tables. They *cannot* contain examples, figures, or other footnotes.

The footnote is placed at the bottom of the page on which it is entered if there is room for it. If only one line of the footnote can fit on the current page, all of the footnote is placed on the next page. If at least two lines fit on the current page, the footnote is split between the two pages.

---

<sup>18</sup> This is the text of a footnote that I entered without using the ID attribute to give the footnote a name.

You'll need to give your footnote a name by using the ID attribute on the footnote tag if you want to make a cross-reference to a footnote from within an example, a figure, or a table. The starter set issues an error message if you put the footnote itself inside a figure, an example, or a table. The other place you would use the ID attribute is when you want to make more than one callout to a footnote.

Here's how to create a footnote using the ID attribute of the FN tag:

```
:fn id=fnxmp.
```

This is the text of the footnote.

It starts with an implied paragraph.

```
:p.It could have several paragraphs, but it  
can't contain examples, figures, or another  
footnote.
```

```
:efn.
```

Check the bottom of the page to see what the starter set did with our footnote.

Formatting Example

And here is our reference.<sup>19</sup>

End of Formatting Example

Although you've probably figured out what the FNREF tag looks like, here it is, just in case:

```
:p.And here is our reference.:fnref refid=fnxmp.
```

You need to use the FNREF tag when a footnote reference callout is inside an example, figure, or table. Place the FN tag just before that example, figure, or table.

---

## Unmatched References

When the starter set cannot find an ID value that matches the REFID value, it lets you know—at the place where you entered the REFID—that no match exists. We've deliberately entered a HDREF, a FIGREF, a LIREF, a FNREF, and a TREF that have unmatched REFIDs ("nohead," "nofig," "nolist," "nofoot," and "notable," respectively) so you can see the output that results.

This markup and text:

```
See :hdref refid=nohead., :figref refid=nofig.,  
:liref refid=nolist., :fnref refid=nofoot., and  
this nonexistent table :tref refid=notable..
```

---

<sup>19</sup> This is the text of the footnote. It starts with an implied paragraph.

It could have several paragraphs, but it can't contain examples, figures, or another footnote.



## Exercise: Cross-References and Footnotes

See if you can create a document that looks like the document on the next page. Call this one EXER7. Because we promised we would make our examples real, you'll have to look at the bottom of the page for the footnote instead of inside the lines around the exercise.

The starter set also gives you a cross-reference listing for this exercise (see Chapter 15, "Cross-Reference Listing" on page 141). Actually, it gives you one every time (unless you ask it not to), but in our previous exercises, there was nothing to put into it.

The markup is shown in "Markup for Cross-References and Footnotes" on page 174.

**Note:** The previous sentence made a cross-reference to a heading and was entered like this:

The markup is shown in `:hhref refid='xrefx'..`

Exercise

This example shows cross-references to:

1. A figure
2. A footnote
3. A list item
4. A table

Here's a cross-reference to Figure 6, so remember to run with the FPASSES or the TWOPASS option to see the correct results. And here we have a footnote reference.<sup>20</sup>

This is the figure being referred to.

**Figure 6. Figure Being Referred To**

Don't forget to put an ID attribute on item 3. above so that this reference to it resolves correctly.

Finally, we want to refer to Table 6.

Here's	the	table
being	referred	to

**Table 6. Table Cross Reference Exercise**

End of Exercise

<sup>20</sup> Of course, when you run the exercise, the footnote will be numbered 1, because it is the first footnote in the document.

To see the results at your terminal (in single-column format rather than the *offset style* we show you here):

**TSO users enter:**<sup>21</sup>

```
script exer7 prof(dsmprof3) co twopass
```

EXER7 and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**CMS users enter:**<sup>21</sup>

```
script exer7 (prof(dsmprof3) co twopass
```

EXER7 and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**ATMS-III users enter:**<sup>21</sup>

```
script * prof(dsmprof3) co twopass
```

The document to be formatted (EXER7—referred to by the \* in the command) is in your ATMS working storage. DSMPROF3 is in the SYSOP (system operator) permanent storage.

---

<sup>21</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization. (Notice that there are three references to this footnote but it appears only once. This is another example of when to use the footnote id attribute rather than to let the footnote reference be placed automatically.)



## Chapter 9. Overall Document Structure

So far we've been telling you how to deal with the elements in the *body* of your document. Now we'll show you how to create title pages, tables of contents, lists of figures, and lists of tables. Perhaps you also need a security classification on every page of your document.

The basic document structure tags have more to do with the overall structure of the document than with the elements of text within it. These tags have no text associated with the tag; rather, they identify a part of the structure of the document.

Let's look for a minute at the overall structure of a general document, which contains these parts:

<b>front matter</b>	The front matter contains the title page, abstract, preface, table of contents, list of figures, and list of tables.
<b>body</b>	The body of the document is the main portion of the document.
<b>appendixes</b>	The appendix section follows the body and contains information supplemental to the material in the body of the document.
<b>back matter</b>	The back matter contains the glossary and the index.

The *first* item you put in a general document is the GDOC tag. It identifies the document type as *general document*.

The first line in the source document for this book contains this tag:

```
:gdoc.
```

If you want a security classification placed at the top of every page, use the SEC attribute, specifying as its value the exact text that you want to appear as the security classification. The text is centered in level 2 highlighting or in a special font for page printers. The security classification also appears on the title page, following the address. The markup might look like this:

```
:gdoc sec='ABC Company Confidential'.
```

While we're on the subject, we might as well tell you about the *very last* tag you put in a general document. Because you already know the starter set rules for end tags, it must be :EGDOC, right?

Now that we've looked at the beginning and at the end of our document, let's look at each of the major pieces in turn.



---

## The Front Matter

The front matter, specified by the FRONTM tag, contains any or all of these elements, as needed:

- Title page (TITLEP tag)
- Abstract (ABSTRACT tag)
- Preface (PREFACE tag)
- Table of contents (TOC tag)
- List of figures (FIGLIST tag)
- List of tables (TLIST tag).

The starter set is smart about the FRONTM tag: it numbers your front matter pages with lowercase roman numerals, and knows not to put any headings that occur in the front matter into the table of contents.

## Title Page

The title page tag (TITLEP) immediately follows the front matter tag. To make a title page, you enter the TITLEP tag, followed by the elements you want on your title page, followed by the TITLEP end tag.

The markup for the sample title page of this book (which appears behind the IBM cover page in the front of this book) looks like this:

```
:titlep.  
:title stitle='GML Starter Set User's Guide'.  
:title.DCF: Generalized Markup Language Starter Set User's Guide (Sample Title Page  
:docnum.SH20-9186-06  
:date.October 3rd, 1989  
:author.Author's Name  
:address.  
:aline.Author's Address  
:eaddress.  
:etitlep.
```

All the elements of the title page, except for the title page tag (TITLEP and its matching end tag), are optional and appear on the title page in the following order, regardless of how you enter them:

1. Title
2. Number
3. Date
4. Author
5. Address.

**Title**

The document title is entered with the TITLE tag, as follows:

```
:title.Document Title
```

The text of the title can't have any tags within it.

The title prints on the title page and is used as a running footing on even-numbered pages throughout the document when the document is being formatted for printing in duplex mode (something you specify when you request formatting).

If your title is too long for the even-page running footing, use the STITLE attribute on the TITLE tag, just as we showed for H0 and H1, to get a short title for the runningfooting. An example of this is shown on the previous page.

**Document Number**

The document number, if any, is entered with the DOCNUM tag. All the text must be either on the same line as the tag or on the next line. The text of the DOCNUM tag can't have any tags within it.

**Date**

The DATE tag causes the starter set to do one of two things:

1. If you enter the DATE tag with no date, the system-supplied date of processing is placed on the title page each time you format the document.
2. If you enter the DATE tag followed by a date, that date is placed on the title page.

While this book was being developed, the date tag was used by itself, as follows:

```
:date.
```

Therefore, we could always tell whether a draft was the latest run by checking the date on the title page.

When we were ready to publish (and thus would want the same date on all copies, regardless of when they were formatted), we put a date following the date tag, as follows:

```
:date.February 25, 1988
```

All the text must be either on the same line as the tag or on the next line. The text of the DATE tag can't have any tags within it.

**Author**

The author's name is entered with the AUTHOR tag. All the text must be either on the same line as the tag or on the next line. The text of the AUTHOR tag can't have any tags within it. If you have more than one author, use an AUTHOR tag for each one. All the authors' names print on the title page in the order they were entered.

### Address

The address of the author or publisher is entered using as many address line (ALINE) tags as needed, surrounded by ADDRESS and its matching end tag. All the text for each address line must be either on the same line as the tag or on the next line. The text of the ALINE tag can't have any tags within it.

Addresses can also be used elsewhere in the document. For example, if you needed to write the following:

```
_____ Formatting Example _____  
  
Address technical comments concerning this specification to:  
  
    Technical Specifications Department  
    Leading Edge Laboratory  
    2000 State of the Art Boulevard  
    Sans-Serif-by-the-Sea, California  
  
_____ End of Formatting Example _____
```

you would enter it like this:

```
:p.Address technical comments concerning  
this specification to:  
:address.  
:aline.Technical Specifications Department  
:aline.Leading Edge Laboratory  
:aline.2000 State of the Art Boulevard  
:aline.Sans-Serif-by-the-Sea, California  
:address.
```

### Abstract

If your document has an abstract, enter it with the ABSTRACT tag. The ABSTRACT tag has the same effect as a level 1 heading: it begins a new page (an odd-numbered page if you specify duplexing when you request formatting) and generates the heading "Abstract."

An abstract would be entered like this:

```
:abstract.  
:p.This report....
```

### Preface

If your document has a preface, enter it with the PREFACE tag. The PREFACE tag has the same effect as a level 1 heading: it begins on a new page (an odd-numbered page if you request duplexing when you request formatting) and generates the heading "Preface."

The preface in this book was entered as follows:

```
:preface.  
:p.The IBM Document Composition Facility....
```

## Table of Contents, List of Figures, and List of Tables

The table of contents, list of figures, and list of tables are produced automatically by the starter set. Just enter the TOC, FIGLIST, and TLIST tags at those points where you want the table of contents, list of figures, and list of tables to appear.

The front matter of this book was entered as follows:

```
...
:frontm.
:titlep.
...
:preface.
...
:toc.
:figlist.
:tlist.
...
```

One other thing you need to know about the table of contents, list of figures, and list of tables: SCRIPT/VS collects the information that goes into them on every pass through your document. It then puts the final results into your document on the last pass where you have your TOC, FIGLIST, and TLIST tags. If you run with only one pass, the table of contents, the list of figures, and the list of tables will be empty—you'll get the headings for them but nothing else. For more information on the FPASSES or the TWOPASS option, see "FPASSES(n)" on page 126 and "TWOPASS" on page 129.

For review drafts, you could put the TOC, FIGLIST, and TLIST tags at the end of your document (just before the GDOC end tag) so they would be printed on the first pass and you could avoid the need to process the document twice. However, if you do put them in the back matter, they will be in two-column format.

**Note:** For information on what to do if the page numbers in the table of contents, the list of figures, and the list of tables don't match the actual pages on which the headings and figures or tables appear, see "Common Problems – Symptoms and Solutions" on page 136.

---

## The Body

When you finish the front matter, you need to identify the body of your document. You do this with the BODY tag. The starter set begins the body of your document, resets the page number to an arabic numeral 1, and puts the headings into the table of contents.

**Note:** Previously we discussed the basic document elements that you would use in the body of a document.

Here's how these elements look for this book:

```
...
:toc.
:figlist.
:tlist.
:body.
:h0 id='part1'.Part One: Getting Started
...
```

---

## The Appendixes

If you have appendixes in your document, use the APPENDIX tag after the body of your document to start the appendix section.

Each H1 tag following the APPENDIX tag is considered to be the start of a new appendix, so the starter set generates the words "Appendix A" for the first one, "Appendix B" for the second, and so forth.

The markup for this book looks like this:

```
:appendix.  
:h1 id=samp2.A Sample GML Document  
...  
:h1 id=solve.Solutions for Exercises  
...
```

You enter the heading in the appendix just as you would in the body; the starter set makes it an appendix for you.

Actually, in writing this book, we also used the tag:

```
:h0.Part Five: Appendixes
```

just before our APPENDIX tag. We did this because we wanted a part divider with the heading "Appendixes" on it before our appendixes.

---

## The Back Matter

The back matter of a document typically contains the glossary and the index.

Use the BACKM tag to specify that you've reached the back matter. The starter set turns off the generation of "Appendix" on your level 1 headings (if you had them on) and sets the back matter into two columns.

In a general document, the index is assumed to be the last element in the document. Running headings and footings are suppressed on pages following the index.

In this book, the back matter is entered as follows:

```
:backm.  
:h1.Glossary  
...  
:index.  
:egdoc.
```

We'll tell you about the index in Chapter 10, "Indexing" on page 93.

---

## Document Structure Summary

Here are all the major elements we've talked about shown in an overall structure. You can use this as a model for creating your own document structures. Anything you don't need, just omit.

```

:gdcc sec='security classification'.
:frontm.
:titlep.
:title stitle='short title'.
Text of Title
:docnum.document number, if needed
:date.either nothing or a specific date
:author.writer's name
:address.
:aline.first line of address
:aline.and as many more as you need
:eaddress.
:etitlep.
:abstract.
...
:preface.
...
:toc.
:figlist.
:tlist.
:body.
...
:appendix.
...
:backm.
...
:index.
:egdoc.

```

---

## Combining Input Files

SCRIPT/VS allows you to create a single output document from more than one source file.<sup>22</sup> This book was produced using approximately 25 separate source files.

You will want to combine source files, especially for large documents, for several reasons:

1. Different people can work on different parts of the document at the same time if the parts are in separate files.
2. Shorter files are more efficient to edit.
3. You can format pieces of the document, saving the cost of processing the whole document when you need to check only a piece of it.
4. The same file can be used in numerous output documents. For example, standard text that you put into all documents of a certain type can exist in

---

<sup>22</sup> What we call "files" here are also known (in TSO, MVS, and VSE) as "data sets" and (in ATMS-III) as "documents." For our purposes, these terms all mean the same thing.

one source file. If you change the standard, you need change it in only one place.

5. The same file can be used more than once in a single document.
6. If your files represent logical segments, such as chapters, it is easier to rearrange your material if you change your mind about the sequence. If the document were all one file, you'd actually have to move the text to reorganize the document. With this book, when we changed our mind about the sequence of chapters (which we did quite often), all we had to do was change the sequence in which the files were processed.

In particular, it is a good idea to put your major figures into separate files, which makes it easier to move them around when you are fine-tuning your document.

Creating a document from more than one input file is easy. Just use the SCRIPT/VS control word `.IM` (for *imbed*) and the name of the file you want imbedded. For example, using CMS,

```
.im gmlssch4
```

is the control word line we used to imbed the chapter on lists (filename GMLSSCH4). Because `.IM` is a control word, like all SCRIPT/VS control words it must start with the period in column 1.

How you name the file you want imbedded can vary, depending on your computing system and how the file you want imbedded is identified in that system. Consult your document administrator or supervisor for details on how you should specify the name for the file you want imbedded.

Your files should have names that are meaningful to you. For example, we knew (after several revisions) that the organization of this book would not change, so we identified our files by chapter. If you know you will change the organization of a document, you might want to identify your files by subject.

A typical way to construct a large document is to have a base file (called the *primary input file*) that consists of the following:

- Major element tags that form the structure of the document
- Imbed controls for the files that actually contain the text
- Comments that explain what the files contain.

For example, a portion of the base file for this book, which was developed using CMS, looks like this:

```
...  
.* GMLSSAXA is a sample GML document  
.im gmlssaxa  
.* GMLSSAXB is solutions for exercises  
.im gmlssaxb  
...
```

As you can see, we started all our filenames with "GMLSS" so we could distinguish them from the files that go into other books, and we used names that were meaningful to us.

The lines beginning with “.” are called comment lines. *Comment lines* are a way of writing notes to yourself (or to other people who may need to look at or perhaps update your source file); they appear in the source file, but they never appear in the formatted output. When SCRIPT/VS sees a line starting with a period in the first position and an asterisk in the second, it ignores anything on that line and goes on to the next line.

Imbedded files can have other files imbedded in them. An imbed trace is included with the cross-reference listing at the back of this book, following the index. If you look at it, you’ll see it lists the files that were used to create this book, in the order in which they were imbedded.

---

## Exercise: Overall Document Structure

This time we want you to create a document (call it EXERALL) on your own. Create the front matter and the body. Put at least one level 1 heading in the body, and imbed the other exercise files you have created.

If you like, you might put some of the exercises in as appendixes. Don’t forget to put in the GDOC end tag.

**Note:** You can use either the FPASSES(*n*) option (where *n* is the *number* of passes specified) or the TWOPASS option of the SCRIPT command to produce correct formatting.

To see the results at your terminal (in single-column format rather than the *offset style* we show you here):

**TSO users enter:**<sup>23</sup>

```
script exerall prof(dsmprof3) cont twopass
```

EXERALL and DSMPROF3 are assumed to be qualified with your TSO userid and TEXT.

**CMS users enter:**<sup>23</sup>

```
script exerall (prof(dsmprof3) cont twopass
```

EXERALL and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**ATMS-III users enter:**<sup>23</sup>

```
script * prof(dsmprof3) cont twopass
```

The document to be formatted (EXERALL—referred to by the \* in the command) is in your working storage. DSMPROF3 is in SYSOP permanent storage.

---

<sup>23</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.





---

## Chapter 10. Indexing

Indexing is more complicated than using the other starter set tags. You might want to skip this section until you need to create an index.

Creating an index using GML is somewhat like creating a table of contents. SCRIPT/VS builds a table of contents for you from the items in the text that you tag as headings. Similarly, it builds an index for you from the items in the text that you tag as index entries. Just as you use the TOC tag to show that you want a table of contents, you use the INDEX tag in the back matter of your document to show that you want an index included.

The main difference between the two is that the headings appear in the body of the document (where you entered them), whereas the items you tag as index entries appear only in the index (sorted for you by SCRIPT/VS).

By putting the index entries into the text at the point where the information you are indexing occurs, you can ensure that, just as with the table of contents, the page references in the index will be correct whenever you format your document. You will never have to update an old-fashioned index by looking up and changing the page numbers! Also, you can tell from looking at the source document whether something is indexed or not. This provides a means of checking the quality of your index.

Figure 1 on page 94 illustrates the terminology we use in this chapter to describe the elements of an index.

**Note:** The indexing tags are acted upon *only* when the INDEX option is specified in the SCRIPT command (as described in “Instructions to SCRIPT/VS” on page 123); otherwise, they are ignored. This saves the time required to process the index when an index is not needed – for instance, on drafts.

---

### Creating Index Entries

The simplest index entry, a primary entry with a page number, is entered with the l1 tag, which says, “This is an index subject at the first level.” It would look like this:

```
:i1.sauces
```

However, most indexes run to more elegant structures, with primary and secondary and sometimes tertiary entries.

Primary entries with secondary entries and secondary entries with tertiary entries may or may not also have page references of their own. The primary and secondary entries that have no page references associated with them are called “index entry headings.”

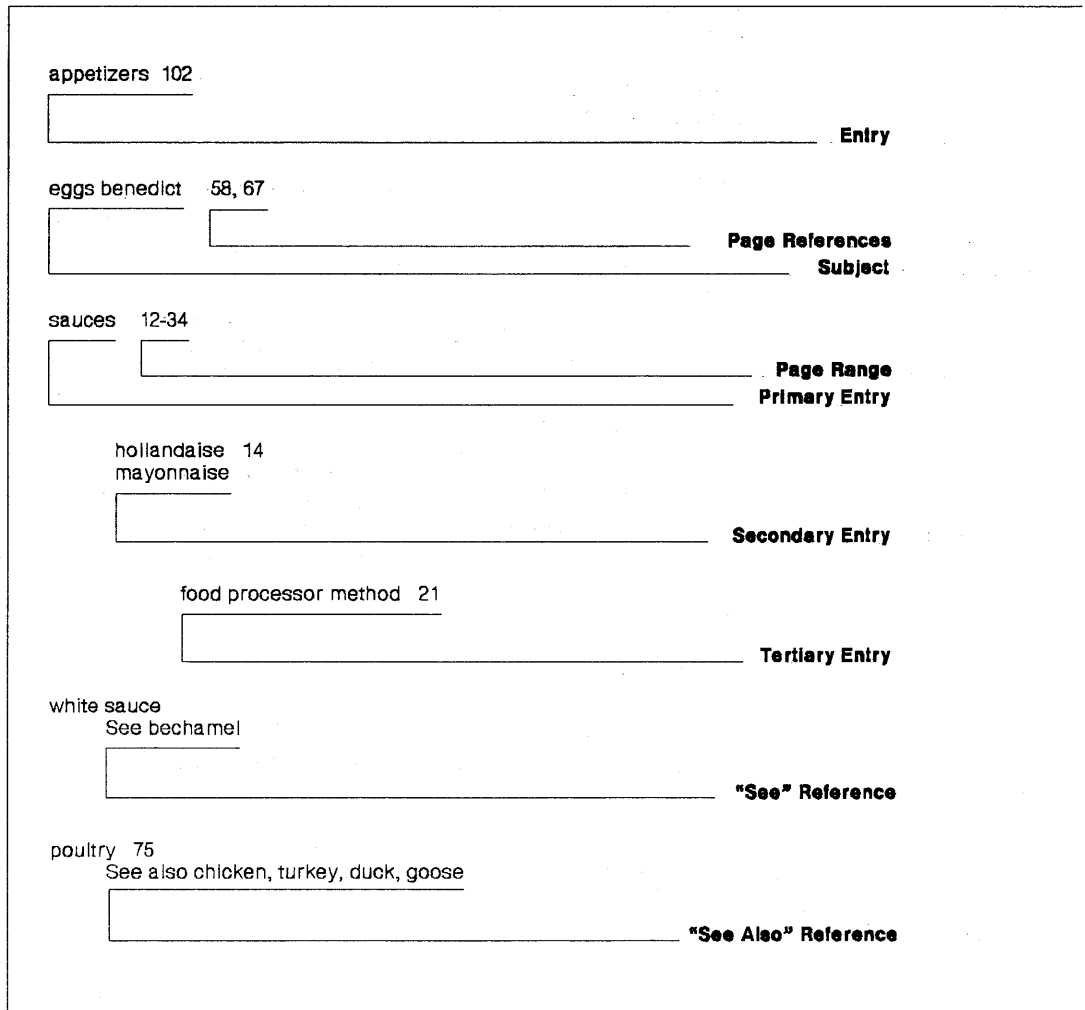


Figure 1. Terminology Used in a Discussion of Indexing

You can associate secondaries with their primaries and tertiaries with their secondaries in two ways:

- By their position in the source file
- Through use of the ID and REFID attributes.

We'll tell you all about the position method first.

Our index entry:

**:i1.sauces**

will, as we mentioned, give us a primary subject with a page number.

Elsewhere in our document, we may want some subentries under “sauces,” but we don’t want another page reference added to the “sauces” line. We could do it like this:

```
:ih1.sauces
:i2.hollandaise
...
:ih1.sauces
:i2.bearnaise
...
:ih1.sauces
:i2.bechamel
```

The IH1 tag says, “this is a primary subject with no page number” —in other words, an index entry heading. The I2 tag says, “this is a secondary subject with a page number.”

When SCRIPT/VS processes the index entries, it matches all the primaries, regardless of whether they were entered with I1 or IH1 tags, which is just what you want it to do. Thus, our entries so far would give us this result:

```
sauces 12
    bearnaise 14
    bechamel 17
    hollandaise 14
```

If you didn’t want a page reference on the “sauces” entry, you would just leave out the I1 tag, and you’d get:

```
sauces
    bearnaise 14
    bechamel 17
    hollandaise 14
```

If you didn’t want to say exactly which sauces are where, you would just enter:

```
:i1.sauces
...
:i1.sauces
...
:i1.sauces
...
:i1.sauces
```

and you would get:

```
sauces 12, 14, 17
```

Because both hollandaise and bearnaise are on page 14, SCRIPT/VS has eliminated the duplicate page reference for you.

The same general principle applies to the tertiary level as well. For instance, these tags:

```
:ih1.sauces
:ih2.mayonnaise
:i3.manual method
...
:ih1.sauces
:ih2.mayonnaise
:i3.blender method
...
:ih1.sauces
:ih2.mayonnaise
:i3.food-processor method
```

added to our existing tags would give us:

```
sauces 12
  bearnaise 14
  bechamel 17
  hollandaise 14
  mayonnaise
    blender method 20
    food-processor method 21
    manual method 18
```

If the food-processor method and the blender method are discussed together, you can cover them both in a single set of entries, as follows:

```
:ih1.sauces
:ih2.mayonnaise
:i3.food-processor method
:i3.blender method
```

When the ID/REFID attributes are not used and the index entries are purely positional, the I3 is associated with the last level-2 entry (either I2 or IH2) that occurred, no matter what has come between (except an I1 or IH1, in which case you have a markup error). The same rule applies to associating secondaries with primaries.

In fact, much text could have intervened between our IH2 in the above example and the two I3 entries. However, we do not recommend that you do this, because in an update to the document some new entries might be inserted between the two, and then your subentries could get listed under chicken soup instead of mayonnaise.

As you can see from the previous examples, repeating the entire structure of primary and secondary to enter a tertiary entry can be tedious. Therefore, the starter set has the ID and REFID attributes on the indexing tags to allow you to get at the structure with just the name you put on the ID. You will remember ID and REFID from our discussion of cross-references; they work in much the same way with the indexing tags.

When you put an ID attribute on an I1, I2, I3, IH1, IH2, or IH3 tag, the starter set “remembers” that entry *and* any higher level entries associated with it.

For example, if you had these entries:

```
:ih1.sauces
:ih2 id=mayo.mayonnaise
```

The starter set associates the ID “mayo” with both mayonnaise *and* sauces.

Then you can enter:

```
:i3 refid=mayo.manual method
...
:i3 refid=mayo.blender method
...
:i3 refid=mayo.food-processor method
```

and get exactly the same results we showed you earlier.

When you use REFID attributes on I2 and I3 tags, they pick up the *higher levels* needed from the structure identified by associating the REFID value to the ID with the same value. (You can’t use REFID on an I1 tag, because there are no higher levels.)

If you want to pick up *all* the levels (that is, you have an identical structure to the one identified with the ID attribute), you use the IREF tag. Because you are picking up all the levels, the IREF tag doesn’t need a level indicator of its own.

So if we had many different ways of making mayonnaise with a blender, we could enter:

```
:ih1.sauces
:ih2.mayonnaise
:i3 id=mayoblnd.blender method
...
:iref refid=mayoblnd.
...
:iref refid=mayoblnd.
```

and we would get this result:

```
sauces
  mayonnaise
    blender method 20, 23, 26
```

## Page Ranges – and Other Things You Can Do with Page References

If you want to show that the sauces section covers a lot of territory, you can do it with the PG attribute. PG can be used on I1, I2, I3, and IREF tags.

To use it to show a page range, you enter:

```
:i1 pg=start.sauces
...
:i1 pg=end.sauces
```

and you will get:

```
sauces 12–34
```

The starter set pairs the value “start” with the value “end” on a matching entry at the same level and produces the range of pages.

This can also be done (and more easily when you have a structure of more than one level) with the IREF tag, as follows:

```
:ih1.sauces
:i2 id=mayo pg=start.mayonnaise
...
:iiref refid=mayo pg=end.
```

which would give you:

```
sauces
  mayonnaise 18-26
```

If you want to point your reader to the most important reference in a string of page references, you can use the PG attribute to say this is the “major” entry. When you tell the starter set that a reference is major, it puts it first in the string of references. The markup might look like this:

```
:i3 refid=mayobl n pg=major.blender method
```

The other thing you can do with the PG attribute is specify some text you want used in place of a page number. It might look like this (because our document is so big it takes two volumes):

```
:i1 pg='(Volume II)'.marinara sauce
```

which would give you:

```
marinara sauce (Volume II)
```

## See and See Also

Another common need in indexes is for “see” and “see also” references.

For this, the starter set gives you the SEE and SEEID attributes on the IH1, IH2, and IREF tags. SEE allows you to specify the text you want in the reference. SEEID is like REFID—it points to a name given in an ID attribute.

You should use SEEID whenever possible, because it ensures that the entry you are pointing your reader to in the index exists. (You’ll get a message in your cross-reference listing if it doesn’t.) But sometimes that’s not feasible, so you have the SEE attribute also.

Your markup would look like this:

```
:i1 id=bech.bechamel sauce
...
:ih1 seeid=bech.white sauce
```

Now if white sauce doesn’t have any page references of its own (no I1 tag for white sauce anywhere) and doesn’t have any secondary entries, the previous markup gives you:

```
white sauce
  See bechamel sauce
```

But if there were other white sauce entries, then you would get:

```
white sauce 32, 33
  See also bechamel sauce
```

SCRIPT/VS figures out whether the reference should be a “see” or a “see also.” You don’t have to worry about it.

SEE works the same way, except that you supply the text you want for the reference, as follows:

```
:ih1 see='chicken, turkey, duck, goose'.poultry
```

and, assuming poultry has another index entry, you get:

```
poultry
  See also chicken, turkey, duck, goose
  how to buy 56
```

In this case, it’s up to you to ensure that chicken, turkey, duck, and goose can all be found in your index.

The “see also” reference is always placed first in the subentries.

When either SEEID or SEE is used on an IREF tag, the REFID must point to a primary or secondary entry. Tertiary entries cannot have “see” or “see also” references, if the SEEID points to an ID on a secondary or tertiary entry, the starter set constructs the full cross-reference for you, as follows:

```
:ih1.sauces
:i2 id='vinaig'.vinaigrette
:ih1 seeid='vinaig'.oil and vinegar dressing
```

which would give you:

```
oil and vinegar dressing
  See sauces, vinaigrette
  ...
sauces
  vinaigrette 47
```

Most of the time, this will be what you want. If not, use the SEE attribute in place of SEEID and spell out exactly what you want.

---

## Sorting Index Entries

The following list defines the basic sort sequence that is common to all of the IBM-supplied languages. SCRIPT/VS sorts the index entries in this order:

1. Blank (code point 40).
2. Required blank (code point 41), numeric space (code point E1), and characters to be omitted. Omitted characters such as the hyphen and the apostrophe do not have unique sort values. They sort as if they were required blanks.
3. Alphabetic characters including the associated accented characters (sorted alphabetically).
4. Numeric characters (sorted numerically).
5. Special characters (sorted according to their code point values).

In addition to the common sorting conventions shown, the sort sequence for each language includes the special sorting requirements of that language. See the *Document Composition Facility: Text Programmer’s Guide* for additional



information on the sort sequence for each supported language. See your supervisor or document administrator if you need to modify any of the sort sequences.

Sometimes the automatic sort provided by SCRIPT/VS might not be suitable to your needs. Some examples are:

- You want to index titles without regard to leading articles, such as “The” and “An”; for instance, when you want to put “The Wind in the Willows” under “wind.”
- You have index entries that start with a special character, but you want them sorted on the first alphabetic character rather than on the special character; for instance, you want to put “\$IBSYS” under the I’s instead of the \$’s.
- You want to put numeric subjects in the alphabetic section as if they were spelled out; for instance, “8-bean salad” sorted as if it were “eight-bean salad.”

The text that goes with an index tag (that is, the text following the markup-content separator) is *always* used for sorting. But you can use the PRINT attribute (on the IH tags) to have something else actually printed in your index, as follows:

1. Use an IH1, IH2, or IH3 tag with the PRINT attribute to show what you want printed when an entry with matching text (or a REFID pointing to the IH) is used. The IH tag with the PRINT attribute *must* precede any other index entries you want to appear under the heading created with that PRINT attribute.
2. Use an I1, I2, I3 (same level as the matching IH) tag with the same text as the IH tag, or use an IREF tag that points to an ID on the IH tag.

For example, suppose we wanted to put “8-bean salad” in the E’s. Our markup would look like this:

```
:ih1 id=eight print='8-bean salad'.eight-bean salad
...
:i1.eight-bean salad
...
:iref refid=eight.
```

The I1 tag and the IREF tag would each have the same result; either one or both could be used. Your output would look something like this:

```
...
eggs benedict 58
8-bean salad 82, 106
endive, Belgian 75
...
```

Remember, what you want printed goes with the PRINT attribute. How you want it sorted depends on the text that follows the markup; this is sometimes called the “sort key.”

The sort key needs to be only long enough to guarantee that the entry will sort as you want. In our example above, all we really would need is:

```
:ih1 id=eight print='8-bean salad'.ei
```

because “ei” is sufficient to ensure the sorting sequence we want. We recommend, however, that you make the key somewhat longer, to ensure that you will still get what you want when someone adds “eiderdown quilts” to this incredible cookbook.

### Notes on the Normal Sort

Blanks are always sorted first (unless your organization has modified the index sort sequence). Thus, “egg whites” comes before “eggs benedict.”

When SCRIPT/VS sorts index entries, it treats them as if they were entered in uppercase. Thus, if you have two entries that are identical except for the case of the letters (for instance, “time” and “TIME”), SCRIPT/VS sorts them as if they were the same. When printed, their order depends on which one is processed first.

If it is necessary to have them both in your index, use the PRINT attribute to control the sort. For example, if you wanted “TIME” to come after “time,” you could use tags like this to get the sorting you want:

```
:ih1 print='TIME'.timez
:i1.timez
...
:i1.time
```

and you would get:

```
time 78
TIME 72
```

because “timez” sorts after “time.”

### Where to Put Index Entries

Your index entries can be entered just about anywhere; they don’t cause any variation in how the text around them is formatted. Here is a list of suitable places to put index entries to ensure the correct page reference:

- After a heading
- After the first input line of a paragraph or paragraph continuation
- After the first input line of a list item or definition description
- After an XMP tag, FIG tag, or TABLE tag. (In particular, if you are indexing a figure that is going to float, the index tag must be inside the figure).

Because the starter set keeps the first few lines of a paragraph on the same page, using the index tags in these places ensures that the page reference picked up for the entry is the same page on which the paragraph starts. If the index entries precede the paragraph, they might be processed (and the page number picked up) before the starter set discovers that it has to start a new page for the paragraph.

Put index entries that have no page references associated with them (see and see also, index headers with no immediate subentries with page numbers, index headers with the PRINT attribute) in one place in the front of your source document. If you scatter them through your document, you will have trouble finding them when you want to change them, because the index itself won’t give you a page number to help you find them.

---

## Getting the Index

The INDEX tag shows where you want your index placed in the back matter.

It would look like this:

```
:backm.
...
:index.
:egdoc.
```

That's all there is to it, except that you must use the INDEX option on your SCRIPT command.

When you don't need the index for a particular run, omit the INDEX option on the SCRIPT command and save the processing time involved in sorting and formatting the index. (Your INDEX tag will give you a heading for the index, but the index will be empty.)

Running headings and running footings are suppressed on pages following the index.

To get an idea of what your final index will look like, look at the index in this book; it was done using these tags.

---

## Summarizing the Indexing Tags

At this point, you're probably having trouble keeping track of what attributes go on which indexing tags, so we will summarize them for you here. First we show you the tags; then we discuss the attributes.

### ***Index Entry Term (I1, I2, and I3)***

```
:Ix          (x is 1, 2, or 3)
  ID=name
  PG=page reference
  REFID=name          (only I2 and I3)
```

All the text of the index entry must be either on the same line as the end of the markup or on the next line.

### ***Index Entry Heading (IH1, IH2, IH3)***

```
:IHx        (x is 1, 2, or 3)
  ID=name
  SEEID=name          (only IH1 and IH2)
  SEE='text for see/see also' (only IH1 and IH2)
  PRINT='text to be printed'
```

The text of the index entry heading must follow the same rule as the text for index entries.

### ***Index Entry Reference (IREF)***

**:IREF**  
**REFID**=name (required)  
**PG**=page reference  
**SEEID**=name  
**SEE**='text for see/see also'

No text is associated with the IREF tag.

### **Index**

**:INDEX.**

No text is associated with the INDEX tag.

### **Attributes:**

All of the attributes are optional, except for the REFID attribute on the IREF tag.

**ID**=name

This attribute gives a name to this level of index subject and all higher levels associated with it. The name can be up to seven alphanumeric characters; the first character should be a letter.

**REFID**=name

This attribute is used to refer to an index structure of the same name, identified with an ID attribute. When used on the I2 or I3 tag, it refers to the levels higher than the level of this tag; when used on the IREF tag, it refers to all the levels in the structure named.

REFID is required for the IREF tag.

**PG**=page reference

Used on the I1, I2, I3, and IREF tags, this attribute allows for a page reference other than just the page number. The value can be any one of the following:

- MAJOR** Identifies this entry as the major entry in a string of entries. DSMPROF3 puts the page number of this entry first.
- START** Identifies this entry as the start of a page range.
- END** Identifies this entry as the end of a page range.
- 'string'** Identifies a string of characters (text) to be used in place of a page number; it must be enclosed in single quotation marks.

**SEEID**=name

This attribute identifies the entry with a matching name in its ID attribute that is to be used as the text for a "see" or "see also" entry.

**SEE**='text for see/see also'

This attribute identifies the text (enclosed in single quotation marks) that is to be used for a "see" or "see also" entry.

**PRINT**='text to be printed'

The text following the end of the markup content separator (.) is the text that is *always* used for sorting the entry.

This attribute identifies the text (enclosed in single quotation marks) to be printed in the index in place of the text of the entry that follows the end of the markup. When it is used, it must come before any other use of the entry.

---

## Readable Source Files

Before we leave the topic of indexing, we want to tell you one more thing.

When documents are heavily indexed, the index entries can make the source file hard to read when you are updating it. You might find it desirable, especially where you have big blocks of index entries (there are some places in this book where the index entries take up 20 or more lines at a time), to set them off with comment lines around the block.

If you'll recall, a comment line begins with a period and an asterisk in the first two positions of the line. When SCRIPT/VS sees that, it ignores the rest of the line, so you can put anything else on it that you please.

So you can, if you want to, enter lines like this around your index entries:

```
. * * * * *
. * * * * *
(block of index entries)
. * * * * *
. * * * * *
```

to help you read your source file more easily.

---

## Exercise: Indexing

Create your own index for EXERALL (the file you created earlier), so you get a feel for how the indexing tags work. (Put the index entries into your EXER1 – EXER5 files.) Put a BACKM tag and an INDEX tag in EXERALL, just before your GDOC end tag.

**Note:** You can use either the FPASSES or the TWOPASS option of the SCRIPT command to produce correct formatting.

To see the results at your terminal:

**TSO users enter:**<sup>24</sup>

```
script exerall prof(dsmprof3) co twopass index
```

EXERALL and DSMPROF3 are assumed to be qualified with your TSO userid and a type of TEXT.

**CMS users enter:**<sup>24</sup>

```
script exerall (prof(dsmprof3) co twopass index
```

EXERALL and DSMPROF3 are assumed to have a filetype of SCRIPT and to be on a disk to which you are linked and accessed.

**ATMS-III users enter:**<sup>24</sup>

```
script * prof(dsmprof3) co twopass index
```

The document to be formatted (EXERALL – referred to by the \* in the command) is in your ATMS working storage. DSMPROF3 is in the SYSOP (system operator) permanent storage.

---

<sup>24</sup> Check with your supervisor or the document administrator to see if there are any special requirements for specifying DSMPROF3 in your organization.



## Chapter 11. Process-Specific Controls

The last tag is the process-specific control (PSC) tag.

This tag name might be a little alarming compared to the other tag names in the starter set. Here's what it means:

Between the PSC tag and its end tag, we are going to include controls (such as SCRIPT/VS control words) that are intended for a *specific* process, rather than the *generalized* markup in the rest of the document. The intent with generalized markup is that it can be processed by any output device that is supported.

PSCs are used:

- When the formatting produced by the starter set is unsatisfactory (for instance, when formatting breaks text at a bad place). We call these *patch* elements, because they are generally temporary until the document is updated.

In many cases, patch elements depend on the device being used. The PSC tag has an attribute that allows us to state, "Do this only when formatting for this particular device with these characteristics."

For example, if you put in a patch to force a column eject when printing on the 1403 at 8 lines per inch, you wouldn't want the same column eject when printing at 6 lines per inch. PSC allows you to control whether the column eject is processed.

- For document elements for which no GML tags are available. We refer to these as *graphic* elements. Graphic elements are often required in the body of figures and can be charts, diagrams, pictures, or any kind of image.

This case, like the patch elements, can also depend on the characteristics of the output device.

- For special purposes or applications as defined by your organization.

For example, it is possible to use SCRIPT/VS to process GML and to produce output that is, rather than a formatted page, a file containing the processing controls for another use of the text. An example of this would be to use SCRIPT/VS to produce processing controls for a second text processor that formats text for a photocomposer. In such cases, it is sometimes necessary to include direct controls for that second text processor. Here, we would use PSC tags to tell SCRIPT/VS to pass these controls directly to the output file without doing any kind of processing on them.

When you use SCRIPT/VS to format this text directly (instead of producing controls for the second text processor), the PSC tag tells SCRIPT/VS to ignore these controls.

The PSC tag has a PROC attribute for specifying the circumstances under which the lines between the PSC and its end tag are to be processed. In the PROC attribute, you can specify one or more physical or logical device types, one or more conditions, or both.



### Physical Devices

You specify a *physical device* (the actual piece of equipment or hardware—such as a 3270 or a 3800) when you are concerned with the type of output device used, but don't care about the special characteristics of that output device. The physical devices you can specify on the PROC attribute are as follows:

3800	The IBM 3800 Printing Subsystem Model 1
1403	The IBM 1403 Printer or other printers that look the same to SCRIPT/VS
3270	The IBM 3270 Information Display family
2741	The IBM Communication terminal that looks like a typewriter
4250	The IBM 4250 Printer
38PP	IBM 3800 Printing Subsystem Model 3 and Model 6
3820	The IBM 3820 Page Printer.
PS	PostScript devices such as the IBM 4216 Personal Pageprinter

### Logical Devices

*Logical devices* are a bit more involved; they take into account not only which piece of equipment you want your output to go to, but also such things as the lines per inch and the size of the paper. Here's a sample of a few of the logical devices you can specify:

3800n8	The 3800 at 8 lines per inch with narrow paper.
1403w6	The 1403 at 6 lines per inch with wide paper.
term	The terminal.

Tables for logical line device types, page device types, and PostScript device types start on page 130.

Now, if you had this PSC tag sequence as part of your document:

```
:psc proc='3800n8'.  
(specify control words here)  
:epsc.
```

the control words between PSC and its end tag would be processed only if "3800n8" were specified as the logical device on the SCRIPT command when you requested formatting for your document.

### Conditions

*Conditions* are simply names you make up (eight alphanumeric characters; the first character should be a letter). A condition specified in the PROC attribute is considered to be "true" when that condition name is paired with the SYSVAR P in the SCRIPT command to format the document. (SYSVARs are explained in "Instructions to the Starter Set (SYSVAR)" on page 120.)

The PSC tag looks like this:

```
:psc proc='value1 value2 ...'.  
...  
:epsc.
```

where *value1*, and so on, are the conditions under which the lines between this PSC tag and its end tag are to be processed. If any of the conditions is *true*, the lines are processed. Otherwise, they are ignored, the starter set skips to the PSC end tag, and processing continues.

To make a condition true and have the control words processed, you must include this SYSVAR option as part of the SCRIPT command line:

```
sysvar (p value1)
```

0 That SYSVAR option says to SCRIPT/VS, "format the control words between the PSC tag and its end tag that have the 'PROC=value1' attribute and value specified as part of the PSC tag."

If you don't specify

```
sysvar (p value1)
```

then any control words or text included within the PSC tag identified by "value1" and its end tag are ignored.

As we mentioned, conditions that you name yourself are considered to be true when you specify the PROC attribute value with the SYSVAR P in your SCRIPT command. Because SCRIPT/VS knows what physical and logical device type it is formatting for, SCRIPT/VS knows which of those conditions is true.

Getting back to how PSC is used, here's an example of a case where you want a page eject on the 1403 at 6 lines per inch (and narrow paper) and on the terminal, but not on the 3800, 4250, 3820, or the 1403 at 8 lines per inch. Your markup would look like this:

```
:psc proc='1403n6 term'.  
.cm page break for 1403n6 or terminal format  
.pa  
:epsc.
```

The .PA (page eject) control word is processed only when formatting for the terminal or for the 1403 at 6 lines per inch on narrow paper.

Notice that we used a comment to remind ourselves (or the next person who has to update this document) what the PSC is all about. In this case, we used the .CM (comment) control word instead of the "." we normally use for comments. The reason we do this is so that your organization can, if it wants to, provide special processing for the comment. (It couldn't do this if you used the ".", because comments entered that way are not processed.) Don't put any semicolons on the .CM line; semicolons are special characters to SCRIPT/VS.

If your document had controls in it for some other text processor, such as TERMTEXT, your markup might look like this:

```
:psc proc='ttx'.  
.cm graphic only for TERMTEXT processing  
  (TERMTEXT controls)  
:epsc.
```

Again we've used .CM, this time followed by the word "graphic," to remind us of any special processing for graphics.

The PSC tag can also be used without the PROC attribute. In such a case, the starter set doesn't actually do anything with the PSC and its end tag – the lines are processed as if the PSC weren't there.

However, the existence of the PSC tag in your markup allows your organization to provide its own processing for the PSC tag, either now or in the future. Documents marked up with SCRIPT/VS control words that are enclosed in PSC tags can be processed by a processor that does not recognize SCRIPT/VS control words.

This use of PSC is to extend or modify the processing provided by the profile (that is, DSMPROF3). A common use of PSC elements is to set your own symbols, which we describe in Chapter 12, "Symbols" on page 113.

The markup would look like this:

```
:psc.  
.cm symbols for names  
  (SCRIPT/VS control words to set symbols)  
:epsc.
```

You should consult your supervisor or document administrator to find out what the rules in your organization are for entering SCRIPT/VS control words (that is, whether your organization requires that you use PSC tags around them). However, in the absence of any specific instructions to do otherwise, you should use PSC tags for every occurrence of control words (except .IM) to ensure the generality of your document and your text data base.

**When using SCRIPT/VS control words, you must be careful that they don't interfere with the control words that make the tags work.** See "Symptom: Unexpected Formatting Results" on page 136.

---

## Part Three: Document Formatting

In addition to the GML tags, there are a number of other important items to know about using the starter set. This information is found in the following chapters:

- Chapter 12, "Symbols" on page 113
- Chapter 13, "Formatting Instructions" on page 119
- Chapter 14, "Error Handling" on page 135
- Chapter 15, "Cross-Reference Listing" on page 141.



---

## Chapter 12. Symbols

In a source document, a symbol is a name that can be replaced with something else during formatting.

Symbols are commonly used to specify characters that can be printed on the printer but that can't be entered at your keyboard. An example might be the round bullet (**•**), which is not available on all keyboards.

Because symbols for this purpose vary from organization to organization, the starter set does not include them. However, your organization may have set up symbols for you to use; check with the document administrator or your supervisor if you need to enter characters that don't appear on your keyboard. We mention them here, even though the starter set doesn't include them, so that you'll know how to enter them if your organization provides them.

All symbols are entered the same way: with an ampersand (&), followed by the symbol name, followed by a period. So a symbol for the round bullet that is named "bul" would look like this:

**&bul.**

(Don't try to use this particular symbol unless you have checked and know it is available in your organization.)

The period here is just like the period you use at the end of your GML markup; it won't print in your output document.

As we mentioned earlier, all your GML tags can be entered in either uppercase or lowercase letters. We recommend, however, that you always use lowercase letters.

However, symbols are what we call "case sensitive"; that is, the symbol **&ABC.** is a different symbol from the symbol **&abc.** This means that when you use symbols, you must know the case of the name as well as the spelling.

---

### Symbols Provided by the Starter Set

The starter set provides these symbols:

<b>&amp;rbl.</b>	is replaced by a required blank
<b>&amp;gml.</b>	is replaced by a GML delimiter(:)
<b>&amp;amp.</b>	is replaced by an ampersand (&)
<b>&amp;semi.</b>	is replaced by a semicolon (;)
<b>&amp;date.</b>	is replaced by the current date
<b>&amp;time.</b>	is replaced by the current time

Here's more information about these symbols:

**&rbl.** Required blank.

This symbol is used when you want to force SCRIPT/VS to give you a blank under conditions where it otherwise might not. For instance, **&rbl.** will prevent SCRIPT/VS from splitting a line between two words when formatting. For example, if you were formatting a paragraph

with a number in the international style (blanks instead of commas), you could enter the number like this:

```
1&rb1.000&rb1.000
```

and SCRIPT/VS would keep it all on one line with just one blank where each &rb1. was requested. It would look like this:

```
1 000 000
```

Using &rb1. will also prevent SCRIPT/VS from inserting extra space between two words when it is justifying text. For instance, you could enter:

```
2&rb1.x&rb1.4
```

to ensure that the space on either side of the "x" is only one space wide.

The &rb1.'s would ensure that your output would look like this:

```
2 x 4
```

This symbol can also be used to insert blank lines in figures and examples; just put the &rb1. symbol on a line by itself.

Sometimes SCRIPT/VS takes out extra blanks in figures and examples; if you replace the first blank with an &rb1. symbol, they won't be taken out.

**&gml.** The GML delimiter.

You need to use this symbol only when you are writing a document about the GML tags.

In normal text, all colons are printed exactly as they are entered. However, if the colon is followed by a valid GML tag name, SCRIPT/VS reads it as a tag. Thus, if you are writing a book about GML tags and you want to illustrate a tag, you can't enter it with a colon. Instead, you must use the &gml. symbol. This book contains many examples of tags that were entered this way:

```
&gml.tagname.
```

**&ampamp.** The ampersand.

This symbol is similar to the &gml. symbol, except that it is used when you want to illustrate a valid symbol.

Like the colon, ampersands occurring in normal text will print as ampersands. However, if they are followed by a valid symbol name, they are considered symbols, and the appropriate substitution is made. If you are writing a document about valid symbols, then you must use this symbol in place of the &. This chapter contains many examples of this kind of entry:

```
&ampamp.symbolname.
```

In fact, to illustrate the symbol "&ampamp." itself, we entered:

```
&ampamp.amp.
```

**&semi.** The semicolon.

The semicolon is used by the starter set as the SCRIPT/VS *control word separator*. Normally, you are not concerned with this. However,

if you are illustrating the use of control word separators, you need to use this symbol in place of the actual semicolon.

Also, you need the `&semi.` symbol when you want to use a semicolon in your own symbol values, as described in the next section, or in the value of an attribute.

Sometimes SCRIPT/VS misinterprets a semicolon as the control word separator and does not print it; simply replace that semicolon with the `&semi.` symbol, which always prints.

**&date.** The date.

Use the symbol `&date.` whenever you want to use the date of the document in places other than the title page. If the DATE tag is followed by a date, then that date is printed when you use this symbol. Otherwise, the system-supplied date of processing is printed.

For example, if you enter the following sentence in your source document:

```
:p.This document was printed on &date..
```

it would appear in your output like this:

Formatting Example

This document was printed on October 3, 1989.

End of Formatting Example

**&time.** The time.

Use the `&time.` symbol whenever you want the system-supplied current time to appear in your document.

For example, if you enter the following sentence in your source document:

```
:p.This document was processed at &time..
```

it would appear in your output like this:

Formatting Example

This document was processed at 2:34 p.m.

End of Formatting Example



---

## Your Own Symbols

You can set up symbols for your own use to keep from having to enter the same lengthy phrase over and over. Symbols are also useful if the name of something you're writing about is likely to change. If you define a symbol name and then use the symbol name in the text, you have to change only the symbol value if and when the name changes.

Here's how you define a symbol name:

```
.se symbolname = 'text you want substituted'
```

The SCRIPT/VS "Set Symbol" control word (.SE) must start with the period in column 1. The *symbolname* can be anything you like, up to ten alphanumeric characters. The *text you want substituted* (the symbol value) is the text that appears whenever you use *&symbolname*. in your document. The symbol value must be enclosed in single quotation marks if it contains any blanks or special characters, such as punctuation marks. If it has a single quotation mark within it, then you must use two single quotation marks in a row, to signal SCRIPT/VS that you do not intend that single quotation mark to be the end of the symbol value.

So you could define a symbol like this at the beginning of your document:<sup>25</sup>

```
.se dcf = 'Document Composition Facility'
```

and then everywhere you used *&dcf*. in your document, like this

```
:p.SCRIPT/VS is the formatter component of the &dcf..
```

you would get:

Formatting Example
SCRIPT/VS is the formatter component of the Document Composition Facility.
End of Formatting Example

Notice that you *do not use the & in the .SE control word, but you must use the & when you actually use the symbol*. Also notice that two periods are at the end of the markup: the first period ends the symbol, and the second period ends the sentence.

Here's an example of how to specify a symbol value that uses a symbol name and has a single quotation mark in it:

```
.se dcfpg = '&dcf.: SCRIPT/VS Text Programmer''s Guide'
```

We used two single quotation marks together (*not* a double quotation mark) in "Programmer's" to signal SCRIPT/VS that we didn't intend that quotation mark to be the end of the symbol value.

---

<sup>25</sup> Of course, you should follow the rules in your organization for using SCRIPT/VS control words in your document; see Chapter 11, "Process-Specific Controls" on page 107.

So now we could enter:

```
:p.Consult the :cit.&dcfpg.:ecit.
for more information on the &dcf..
```

and we would get:

Formatting Example

Consult the *Document Composition Facility: SCRIPT/VS Text Programmer's Guide* for more information on the Document Composition Facility.

End of Formatting Example

If we had mistakenly typed &DCFPG. and &DCF. instead (forgetting that we had defined our symbols with lowercase names), we would get the following:

Formatting Example

Consult the &DCFPG. for more information on the &DCF..

End of Formatting Example

So now you know what happens if you enter a symbol name incorrectly – SCRIPT/VS treats it as text. If you follow the rule of *always* using lowercase letters for symbols, IDs, and so forth, then you never have to worry about remembering what case they are in.

**Long Input Records:** Defining your own symbols is also a way of getting 256 characters worth of information into an input record.

You might have to do this if you need to enter long titles, headings, or figure captions, but the editor program or word processing equipment you are using does not allow you to create input records that long.

For example, to produce the title of this book, we could have entered the following control words:

```
.se dcf = 'Document Composition Facility'
.se gmul = 'Generalized Markup Language'
.se ssug = 'Starter Set User''s Guide'
.se stpg = '(Sample Title Page)'
```

and then entered our TITLE tag like this:

```
:title stitle='GML &ssug.'.
:title.&dcf.: &gmul. &ssug. &stpg.
```

which is short enough to fit anywhere.



## Chapter 13. Formatting Instructions

Many times in this book we've mentioned instructions you can specify when you request formatting for your document.

Basically, you can specify two types of instructions at run time:

1. Instructions to the starter set, which specify such things as:

- Whether H0 and H1 force an odd-numbered page (the default is no odd-numbered page)
- Whether headings are to be numbered (the default is no head-level numbering)
- Whether the document is to be
  - Single column (the default)
  - Double column
  - Offset style (as used in this book)
- Whether a title page is to be printed, and how it should look (the default is like the sample title page in this book)
- Whether a cross-reference listing is to be produced (the default is yes)
- A value to test for in the PROC attribute of the PSC tag (as discussed in Chapter 11, "Process-Specific Controls" on page 107).

These instructions are specified as SYSVARs (system variables) either in the options on the SCRIPT command or in an options file called by the SCRIPT command.

2. Instructions to SCRIPT/VS itself, which specify such things as:

- The profile to be used (in our case, DSMPROF3)
- The size of the left margin
- The device for which formatting is to be done
- Whether the output is to be printed, displayed, or written to a disk file
- Whether the index tags are to be processed
- Whether spelling verification is to be done
- Whether read and write files (SYSVAR R and W) are to be created to correctly resolve page references.

The SCRIPT command options that are most commonly used are described in "Instructions to SCRIPT/VS" on page 123. For a complete description of all the options that can be specified, see *Document Composition Facility: SCRIPT/VS Language Reference*.

---

## Instructions to the Starter Set (SYSVAR)

System variables (SYSVARs) are values that can be passed to the starter set on the SCRIPT command (or, for CMS and ATMS-III, an options file called by that command) that allow you to control certain aspects of the starter set processing.

SYSVARs are specified in your command options as follows:

SYSVAR(*c* *value* *c* *value* ...)

where *c* is the single-character designation of the system variable you specify and *value* is the value you assign to it.

The SYSVARs recognized by the starter set are as follows:

**C**            **Value and Meaning**

**S**            (Style) can be set to:

1                    single column (default)

2                    double column

OFFSET            offset

Single column (1) sets both text and headings flush left.

Double column (2) sets two columns.

Offset (OFFSET) is the style used for this publication, where the text of the body and appendix is set on a line offset from the left margin, with heading levels 0–4 flush left.

As an example, if you want your document formatted in double columns, you would enter

sysvar(s 2)

as an option on the SCRIPT command.

**T**            (Title Page) can be set to:

RIGHT            title page with elements aligned on the right (default).

CENTER           title page with elements centered

LEFT             title page with elements aligned on the left

YES              title page with elements aligned on the right (same as RIGHT)

NO                no title page

As an example, if you do not want your title page to print, you would enter

sysvar(t no)

as an option on the SCRIPT command.

**D**            (Duplexing) can be set to:

NO                duplexing off; level 0 and level 1 headings start on the next new page with no blank pages (default).

**YES**                   duplexing on; level 0 and level 1 headings begin on the next odd-numbered page and are right aligned for one- and two-column formats, leaving blank pages if necessary.

Duplexing on (YES) is intended for documents that are to be printed on two sides of the paper. (YES) causes the running footings to be formatted differently on odd and even pages.

Duplexing off (NO) is intended for documents that are to be printed on only one side of the paper. (NO) causes the running footings to be formatted the same on all pages, and no blank pages are produced.

For example, if you want your document printed with duplexing on, you would enter:

```
sysvar(d yes)
```

as an option on the SCRIPT command.

**W**                   (Write) can be set to:

filename           (up to eight characters)

This may be any name that you want to assign to a file of ID information that is created when the document is processed.

This file, newly created as the result of specifying SYSVAR W, contains all the IDs specified on the heading, figure, footnote, and list item tags.

For example,

```
sysvar(w myids)
```

creates a file, myids, that contains all the IDs (except the indexing IDs) included in the document being processed.

When this newly created file is read (using the SYSVAR R option), the collected IDs are used to resolve forward references in a document so that page references resolve correctly.

SYSVAR R (for Read) and W (for Write) can both be specified using the same name. If, for example, you enter:

```
sysvar(r myids w myids)
```

then changes made to IDs or IDs added to the document being processed are included in a new file (specified with the SYSVAR W) while the earlier version (specified with the SYSVAR R) is being read.

**Note:** Write is valid *only* in CMS and TSO.

**R**                   (Read) can be set to:

The name of the file that was created using SYSVAR W.

The named file holds the collected IDs from the document that was processed when SYSVAR W was specified.

If, for example, you read the file by entering:

```
sysvar(r myids)
```

this will resolve forward references to the IDs within the document, so that page references resolve correctly on the first pass.

SYSVARs R and W can both be specified using the same name.

When you specify both R and W, changes made to existing IDs or new IDs added to the document being processed are included in the newly written file while the earlier version is being read. See the W (Write) command option.

Once the ID file has stabilized and no new ID entries are being added, a document can be formatted with correctly resolved forward references with only one pass necessary when SYSVAR R is used.<sup>28</sup>

You can also format parts of a document and get correctly resolved references to parts not being formatted. You can do this by specifying SYSVAR R and the file (containing all the IDs) that was created when the entire document was formatted previously.

**Note:** Read is valid *only* in CMS and TSO.

**H** (Head Numbering) can be set to:

NO do not number headings (default)

YES number headings 1 through 4 in the body

value number headings 1 through 4 in the body starting with this value

This allows you to process pieces of the document and to get the headings numbered as if the whole document were being processed. For example, if the first heading in the piece is an H3, and you want it numbered 3.6.2, you would enter:

```
sysvar(h 3.6.2)
```

**X** (Cross-Reference Listing) can be set to:

YES produce cross-reference listing (default)

NO do not produce cross-reference listing

DSMPROF3 generates an extremely useful listing at the end of your document. It contains information about all the ID and REFID attributes used (HDREF, FIGREF, FNREF, and so on) and an imbed trace. You will definitely want this listing while you are developing your document. However, if you are printing many copies (say, for distribution as review drafts), you will want to suppress this listing. This SYSVAR parameter lets you do that.

If you do not want the cross-reference listing to print, enter:

```
sysvar(x no)
```

as an option on the SCRIPT command.

**P** (Process Condition) can be set to:{

value sets *value* on (that is, the condition is true)

---

<sup>28</sup> You will need to use either the FPASSES or the TWOPASS option to resolve forward index references and to fill in the table of contents and the list of illustrations.

This SYSVAR parameter allows you to set the condition that is to be true when processing PSC tags with the PROC attribute specified. (See Chapter 11, "Process-Specific Controls" on page 107.)

For example, if you had a document with two versions of a figure and the version you want printed changes, depending on who will receive it, then you might set up your figures like this:

```
:psc proc='figold'.
...fig tags and text...
:epsc.
:psc proc='fignew'.
...fig tags and text...
:epsc.
```

Then, if you want *fignew* printed rather than *figold*, you would enter the following with your SCRIPT command:

```
sysvar(p fignew)
```

If neither FIGNEW nor FIGOLD were specified, both of the PSC/EPSC groupings would be ignored during processing.

You can specify as many SYSVAR parameters as you need—your SYSVAR option would look like the following if you wanted the file named myids to be read to resolve IDs, the figure identified by PSC PROC=FIGNEW printed, with double columns, duplexing on, headings numbered, and no cross-reference listing.

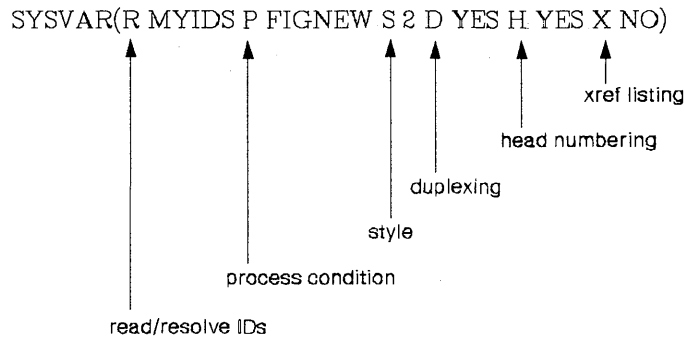


Figure 2. Example of SYSVAR Parameters

The order in which you enter the pairs does not matter.

## Instructions to SCRIPT/VS

To process your documents, you need to know how to invoke the formatter, SCRIPT/VS. You also need to be familiar with some of the SCRIPT/VS command options. We discuss some of the commonly used command options later in this chapter. A complete description of the command options is given in the *Document Composition Facility: SCRIPT/VS Language Reference*.

Some subjects we discuss here are independent of the computing system you use and the way your organization uses it. These generally deal with the formatting options (such as the SYSVARs we talked about before, whether the



index is to be produced, and how many passes you want) and other services SCRIPT/VS provides, such as spelling checking and error-message handling.

Other subjects are not; these generally deal with how data sets, files, or documents are identified, what output-device specifications can be used, and how output is routed to a device. Your organization may have set up special commands or procedures for you to use instead of what we show you here.

### Identifying the Document to Be Processed

The command that invokes SCRIPT/VS is "SCRIPT."<sup>27</sup> The name of the document to be processed is entered immediately after the command, separated from the command name by at least one blank:

```
script mydoc
```

If you are using TSO, a data set name that is not fully qualified is assumed to have the default qualifiers of your userid and TEXT.

If you are using CMS, the default filetype is SCRIPT.

If you are using ATMS-III and the document is in your working storage, use an asterisk (\*) for the document name.

If you are using any other computing environment, consult your supervisor or the document administrator as to what to do.

The document name is followed by any options to be used. The SCRIPT command options that are most commonly used in GML processing are described below.

### SCRIPT Command Options

The first parameter of the SCRIPT command is an identifier of the file or data set that contains your document. Options follow the document name.

In CMS, a left parenthesis "(" must separate the options from the document name.

```
script mydoc (options
```

In TSO and ATMS-III, options must be separated from the document name by at least one blank:<sup>28</sup>

```
script mydoc options
```

Separate options from each other by at least one blank. Some options have values or "suboptions"; enclose these values in parentheses.

---

<sup>27</sup> Even this may have been altered by your organization. For instance, if you have more than one release of SCRIPT/VS available, then each one of them has to be named differently.

<sup>28</sup> If you use ATMS-III, refer to the *ATMS-III: Terminal Operator's Guide* for additional information about using SCRIPT/VS in your environment, especially when you use the LIB option of the SCRIPT command or when you want your output sent to an off line printer.

In the following descriptions, options are shown in uppercase letters and values are shown in lowercase letters. You may enter the options in either uppercase or lowercase letters.

### Processing Options:

#### **BIND(obind ebind):**

Specifies that the printed portion of the output page is to be shifted to one side to leave room for binding. It is shifted “obind” space units to the right on odd pages, and “ebind” space units on even pages. (You would normally make “ebind” smaller than “obind” for duplex printing, where the output is ultimately to be printed on two sides of the paper, so even pages would have a large enough right margin for binding.)

The “ebind” value may be omitted when you want all pages shifted the same amount (as you normally would for printing on one side of the paper).

If you don't specify any bind, the default depends on the device type (see page 130 through page 132 for the bind values used for each device type).

#### **CHARS(font1 font2 ...):**

Specifies the fonts to be used when formatting. If CHARS is not specified, a default font is used. See “Overriding the Default Font Using the CHARS Option” on page 134 for more information.

**For the 3800, Model 1:** You can specify two uppercase and lowercase fonts—the first is the normal text font, and the second is the font used for headings, highlighting, and so forth. For example, if you were formatting for the 3800 Model 1, you might specify

```
CHARS(GT12 GB12)
```

for “gothic text, 12-pitch” and “gothic bold, 12-pitch,” respectively. The fonts available for the 3800 Model 1 are shown in *Document Composition Facility: SCRIPT/VS Text Programmer's Guide*. Your organization may have added its own fonts to the ones available.

**Note:** In addition to the font specification on the SCRIPT command, a compatible font specification must be given in the job control statements for printing the document on the 3800 Model 1. Consult your supervisor or the document administrator in your organization to ensure the correct font specifications have been made.

**For Page Devices and PostScript Devices:** You should consult the document administrator or your supervisor to see what fonts are available when you request formatting for page devices or PostScript devices. See “Font Requirements for Page Printers” on page 133 and “Font Requirements for PostScript Devices” on page 133 for more information concerning fonts.

#### **FONTLIB(libname):**

Specifies the font library to be used when formatting for page printers or PostScript devices. The font library contains character-set information and image patterns of these characters. The font library requested on the SCRIPT

command must be available on the system when the formatted document is printed.

The fonts available vary from one organization to another. Refer to "Font Requirements for Page Printers" on page 133 or "Font Requirements for PostScript Devices" on page 133. Also, see your document administrator or supervisor for the fonts used with each device at your location.

### **FPASSES(n):**

Allows you to specify more than two formatting passes on documents. When two passes are inadequate to ensure that all page number references are resolved correctly, the FPASSES option may be used to specify more passes. The IBM-supplied maximum number of formatting passes is four. (This may have been changed, however, at your location. Check with your supervisor or the document administrator.)

The FPASSES option causes SCRIPT/VS to process your input document the specified number of times, but it produces output only on the last pass.<sup>29</sup>

**Note:** FPASSES and TWOPASS are mutually exclusive options. See "TWOPASS" on page 129 for related information.

### **INDEX:**

Causes SCRIPT/VS to create an index from index entries specified in the text of the document.

If you omit the INDEX option, any indexing tags in the document are ignored. See Chapter 10, "Indexing" on page 93 for information on the indexing tags.

### **NOSEGLIB:**

Specifies that no library search is to be made for page segments. Use NOSEGLIB when you have specified page segments within your document with the .SI [Segment Include] control word, but those page segments are not yet within a segment library when you issue the SCRIPT command. This prevents an error message from being issued.

### **OPTIONS(name):**

Specifies the name of a file containing more SCRIPT/VS command options. The options are processed as if they had been entered where the OPTIONS option appears.

**Note:** The OPTIONS option is available only in CMS and ATMS-III.<sup>30</sup> In CMS the file containing the options is assumed to have a filetype of OPTIONS.

---

<sup>29</sup> Keep in mind that use of this option will significantly increase the processing time for your document.

<sup>30</sup> If you use ATMS-III, refer to the *ATMS-III: Terminal Operator's Guide* for additional information on using SCRIPT/VS in your environment, especially when you use the LIB option of the SCRIPT command or if you need instructions in getting your output.

**PAGE:**

Allows you to print only the specific pages of formatted output that you want. The PAGE option can be specified in different ways, depending on the range of pages that you want printed. You can specify the PAGE option like this:

PAGE [[FROM] frompage [TO] topage]

PAGE [[FROM] frompage FOR n]

PAGE [[FROM] page ONLY]

If no suboption is given with the PAGE option, you are asked for the page you want printed (except under ATMS-III, where it is ignored.)

Here are some examples of how to use this option when you issue the SCRIPT command:

page (from 8 to 16)	this would print pages 8 through, and including, 16
page (from 8 for 2)	this would print 2 pages — 8 and 9
page (6 only)	this would print only page 6

**PROFILE(name):**

Specifies the name of the file you want used as the document profile.<sup>31</sup> The profile provided with SCRIPT/VS for use with the GML starter set is named DSMPROF3.

**Note:** Because the name "PROFILE" is assigned as the default for the SCRIPT/VS profile, you should avoid using this name for a fileid, data set name, qualifier, or member name.

**PSOUT:**

Specifies whether SCRIPT/VS generates PostScript output in ASCII or EBCDIC when a PostScript device is specified with the DEVICE option. If you specify a PostScript device and do *not* specify PSOUT, the default is to produce an ASCII file.

If you specify PSOUT(ASCII), SCRIPT/VS produces an ASCII file. You can then download the file to an IBM PC and print the file on a locally-attached PostScript device, such as the IBM 4216 Personal Page Printer.

If you specify PSOUT(EBCDIC), SCRIPT/VS produces an EBCDIC file. You can look at the EBCDIC file (for diagnostic purposes) on the host system, but you cannot download and print the file.

**Note:** The PSOUT option is valid *only* for PostScript devices.

**SEGLIB(name):**

Specifies the name of the segment library you want searched when formatting

---

<sup>31</sup> ATMS-III users should refer to the *ATMS-III: Terminal Operator's Guide* for additional information on using SCRIPT/VS in your environment, especially when you use the LIB option of the SCRIPT command or if you need instructions in getting your output.

for page devices. SCRIPT/VS searches the segment library for any page segments you have included in your document with the .SI [Segment Include] control word:

If SEGLIB is not specified, SCRIPT/VS automatically searches the default library for requested page segments. You must specify the SEGLIB option if the page segment you request resides in a different library. Remember, SCRIPT/VS searches only one segment library, either the default library or the one you specify with SEGLIB.

The default segment libraries for the IBM 4250 Printer are:

<b>System</b>	<b>Default Library</b>
CMS	SEGLIB(PSEG4250)
TSO	SEGLIB(SYS1.PSEG4250)
ATMS-III	NOSEGLIB
MVS/DLF	SEGLIB(PSEG4250)
VSE/DLF	See the note below.

The default segment libraries for the IBM 3800 Printing Subsystem Model 3 are:

<b>System</b>	<b>Default Library</b>
CMS	SEGLIB(PSEG38PP)
TSO	SEGLIB(SYS1.PSEG38PP)
ATMS-III	NOSEGLIB
MVS/DLF	SEGLIB(PSEG38PP)
VSE/DLF	See the note below.

The default segment libraries for the IBM 3820 Page Printer are:

<b>System</b>	<b>Default Library</b>
CMS	SEGLIB(PSEG3820)
TSO	SEGLIB(SYS1.PSEG3820)
ATMS-III	NOSEGLIB
MVS/DLF	SEGLIB(PSEG3820)
VSE/DLF	See the note below.

**Note:** For more information about the .SI [Segment Include] control word in the VSE environment, see the *Document Composition Facility: SCRIPT/VS Language Reference*.

If you know a page segment does not yet exist or there is no segment library, specify the NOSEGLIB option of the SCRIPT command. SCRIPT/VS will not search for a segment library, and no error message will be issued. (See href refile=noseg.)

**SPELLCHK:**

Causes SCRIPT/VS to perform spelling verification.

Because spelling verification requires additional processing time (each word is checked), you should use this option only when your document is almost complete. You won't want to use it on every draft unless you feel it is absolutely necessary.

If you request spelling verification when processing with the starter set, everything is checked except the contents of examples, figures, running headings, and running footings. (The words that SPELLCHK locates that are not a part of its spelling dictionary are listed with the error messages.)

**SYSVAR(x value ....):**

Specifies system variables that can be used to control the processing of a document.

See "Instructions to the Starter Set (SYSVAR)" on page 120 for information on the SYSVAR options supported by the starter set and how to specify them.

**TWOPASS:**

Causes SCRIPT/VS to process your input document twice, producing output only on the second pass.

TWOPASS is used to resolve forward references (for example, when an automatically generated table of contents or list of illustrations is in the front of a book). Not unexpectedly, this option nearly doubles computer processing time for the document.

**Note:** FPASSES and TWOPASS are mutually exclusive options. See "FPASSES(n)" on page 126 for related information.

**Logical Device and Output Destination**

**DEVICE (type):**

Specifies the “logical” device for which formatting is to be performed. In this context, “logical” means a combination of physical device type, page size, and number of lines per vertical inch. The logical device types supported in the starter set include line devices, page devices, and PostScript devices and are shown starting on page 130.

**Note:** See the document administrator for procedures to get your document printed at your location.

Below are the logical line devices you can specify with the DEVICE option and the defaults for each device.

Logical Device Type	Real Device Type	Lines per Inch	Page Size		Margins			Line Length
			Width	Length	Bind	Top	Bottom	
TERM 2741 3270	( <sup>1</sup> ) 2741 3270	6	132 204	11i 11i	2	.5i	.5i	6i
1403N6 1403N8 1403W6 1403W8 1403W6 1403W8S 1403SW <sup>2</sup> STAIRS	1403	6 8 6 8 6 8 6 6	8.5i 8.5i 13.5i 13.5i 13.5i 13.5i 8.5i 13.5i	11i 11i 11i 11i 8.5i 8.5i 11i 11i	1i	.5i	.5i	6i
3800N6 3800N8 3800N12 3800W6 3800W8 3800W12 3800N6S 3800N8S 3800N12S 3800W6S 3800W8S 3800W12S	3800	6 8 12 6 8 12 6 8 12 6 8 12	8.5i 8.5i 8.5i 13.5i 13.5i 13.5i 11i 11i 11i 13.5i 13.5i 13.5i	10i 10i 10i 10i 10i 10i 7.5i 7.5i 7.5i 7.5i 7.5i 7.5i	1i	0	0	6i
<sup>1</sup> The physical device type corresponding to the TERM logical device can be either 2741 or 3270, depending upon the actual terminal type. Under ATMS, the physical device type is always 2741. <sup>2</sup> This is a 12-pitch device; all other 1403 devices are 10-pitch.								

**Table 7. SCRIPT/VS Logical Line Devices**

Below are the logical page devices you can specify with the DEVICE option and the defaults for each device.

Logical Device Type	Real Device Type	Lines per Inch	Page Size		Margins			Line Length
			Width	Length	Bind	Top	Bottom	
38PPN 38PPW 38PPNS 38PPWS	3800-3	(³)	8.5i	10i	1i	0	.125i	6i
			13.5i	10i				
			11i	7.5i				
			13.5i	7.5i				
38PPW90 38PPNS90 38PPW270			10i	13.5i	.5i	.5i	.5i	6i
			7.5i	11i				
			10i	13.5i				
3820A 3820A90 3820A180 3820A270 3820L 3820A4 3820B4 3820B5	3820	(³)	8.5i	11i	1i	.5i	.5i	6i
			11i	8.5i				
			8.5i	11i				
			11i	8.5i				
			8.5i	14i				
			210mm	297mm				
			257mm	364mm				
			182mm	257mm				
4250A 4250B 4250L 4250A3 4250A4	4250	(³)	8.5i	11i	1i	.5i	.5i	6i
			11i	17i				
			8.5i	14i				
			297mm	420mm				
			210mm	297mm				
PG1A PG1A90 PG1A180 PG1A270 PG1L PG1A4 PG1B4 PG1B5	3820 ⁴	(³)	8.5i	11i	1i	.5i	.5i	6i
			11i	8.5i				
			8.5i	11i				
			11i	8.5i				
			8.5i	14i				
			210mm	297mm				
			257mm	364mm				
			182mm	257mm				
PG2A PG2A90 PG2A180 PG2A270 PG2L PG2A4 PG2B4 PG2B5	3820 ⁴	(³)	8.5i	11i	1i	.5i	.5i	6i
			11i	8.5i				
			8.5i	11i				
			11i	8.5i				
			8.5i	14i				
			210mm	297mm				
			257mm	364mm				
			182mm	257mm				
PG3A PG3W PG3NS PG3L PG3A4 PG3B4 PG3B5	4224 ⁵	(³)	8.5i	11i	1i	.5i	.5i	6i
			13.5i	11i				
			11i	8.5i				
			8.5i	14i				
			210mm	297mm				
			257mm	364mm				
			182mm	257mm				

³ The linespacing for these is determined by the .LS [Line Spacing] control word and by the fonts used in the document.

⁴ The DCF physical device type specified is 3820. The actual device may be 3820, 3812, or other printers compatible at the data stream level. Refer to "Using Generic Logical Device Types" in the *Document Composition Facility: SCRIPT/VS User's Guide* for more information.

⁵ The DCF physical device type specified is 4224. The actual device may be a 4224-2xx or other printer compatible at the data stream level.

Table 8. SCRIPT/VS Logical Page Devices



## Formatting Instructions

Below are the logical PostScript devices you can specify with the DEVICE option and the defaults for each device.

Logical Device Type	Real Device Type	Lines per inch	Page Size		Margins			Line Length
			Width	Length	Bind	Top	Bottom	
PSA	PostScript	(*)	8.5i	11i	1i	.5i	.5i	6i
PSA90			11i	8.5i				
PSA180			8.5i	11i				
PSA270			11i	8.5i				
PSB			11i	17i				
PSL			8.5i	14i				
PSA3			297mm	420mm				
PSA4			210mm	297mm				
PSB5			182mm	257mm				

\* The linespacing for PostScript logical devices is determined by the .LS [Line Spacing] control word and 1.1 times the pointsize of the fonts used in the document.

**Table 9. SCRIPT/VS Logical PostScript Devices**

### **FILE (name):**

Causes output to be formatted for the specified logical device and stored in the named file or data set, instead of being displayed or printed.

If the name (and its surrounding parentheses) is omitted, SCRIPT/VS creates a name for the output document, depending on the device type. For more information about default file names in your operating environment, refer to the *Document Composition Facility: SCRIPT/VS Language Reference*.

**Note:** The FILE option is not available in ATMS-III.

### **PRINT:**

Causes output to be formatted for the specified logical line device and printed on the system printer (in CMS and TSO).<sup>32</sup> If no logical device is specified, "DEVICE(1403W6)" is assumed. For more information on using the PRINT option in your operating environment, refer to the *Document Composition Facility: SCRIPT/VS Language Reference*.

### **TERM:**

Causes output to be formatted for the specified logical device and displayed at a terminal, instead of on the physical device. If no logical device is specified, "DEVICE(TERM)" is assumed.<sup>32</sup>

## **Error Handling**

### **CONTINUE:**

Normally, SCRIPT/VS stops processing when it finds an error. This option directs it to continue processing unless the error is a severe one.

<sup>32</sup> If you are an ATMS-III user, refer to the *ATMS-III: Terminal Operator's Guide* for additional information on using SCRIPT/VS in your environment, especially when you use the LIB option of the SCRIPT command or if you need instructions in getting your output.

**MESSAGE(DELAY ID TRACE):**

This option, with the DELAY parameter specified, causes error messages to be printed at the end of the output document instead of being displayed at a terminal during processing.<sup>33</sup>

The ID parameter causes SCRIPT/VS to include the error-message identifier along with the error message.

The TRACE parameter provides additional information pointing to where the error occurred by tracing through all the imbedded files from the one where the error occurred back to the primary input file.

---

## Font Requirements for Page Printers

SCRIPT/VS has as the original default font Monotype Times New Roman (5771-AAR)(\*\*) for the IBM 4250 Printer, and Sonoran Serif (5771-ABA)<sup>34</sup> for the IBM 3820 Page Printer and the IBM 3800 Printing Subsystem Model 3 and Model 6. These default fonts may have been changed at your installation.

In addition to requiring a typographic font as the initial (body) font, the starter set also requires the Typewriter and Pi Specials (5771-AAW) font for the IBM 4250 Printer, and the Pi and Specials (5771-ABC) font for the IBM 3820 Page Printer and IBM 3800 Printing Subsystem Model 3 and Model 6 for use in formatting unordered list bullets.

---

## Font Requirements for PostScript Devices

SCRIPT/VS requires that the following typeface families be available to PostScript devices in order to format GML starter set documents:

- Times Roman
- Courier.

Each font that is used in the document must be available to the PostScript device and must have a corresponding Adobe Font Metrics (AFM) file in the PostScript font library. For example, for you to format a GML starter set document with Times Roman and Courier fonts, the Times Roman and Courier AFM files must be available in the PostScript font library.

For more information about PostScript fonts, see the *Document Composition Facility: SCRIPT/VS User's Guide*.

---

<sup>33</sup> If you are an ATMS-III user, refer to the *ATMS-III: Terminal Operator's Guide* for additional information on using SCRIPT/VS in your environment, especially when you use the LIB option of the SCRIPT command or when you want your output.

<sup>34</sup> Sonoran Serif is a functional equivalent to Monotype Times New Roman.

---

## Overriding the Default Font Using the CHARS Option

If you wish to override the default typeface when formatting for a page printer or a PostScript device, specify a different font in the CHARS option of the SCRIPT command. Although you can specify up to 31 fonts on CHARS, the starter set uses only the first one specified.

The initial font, specified either by the default font or on CHARS option, must be a typographic font available in a variety of sizes and weights. See your document administrator to learn which fonts are available at your location and how to access them.

---

## Chapter 14. Error Handling

When you format a document and problems occur, you can expect to get two kinds of error messages: starter set messages and SCRIPT/VS messages. The first part of this chapter contains some general information about error messages; however, you should refer to *Document Composition Facility Messages* for a list of the messages and their explanations.

Sometimes your output is wrong and you get no messages at all. In this case, SCRIPT/VS has no way of knowing that you aren't getting what you want and, like all programs, does just as it's told. When this kind of problem occurs, see "Common Problems – Symptoms and Solutions" on page 136 to see if the problem is one we already know how to solve. If this doesn't work—you've checked your markup and are sure that the markup is correct—go to your trouble shooter with the problem.

---

### Starter Set and SCRIPT/VS Messages

**Starter Set Messages:** These messages begin with + + +, tell you about markup errors that have been detected, are self-explanatory, and are followed by the number of the page on which the problem occurred.

**SCRIPT/VS Messages:** These messages fall into two categories:

- Messages resulting from markup errors that the starter set didn't detect

Because the starter set cannot check for all possible markup errors, some errors slip by it. However, because these errors can cause problems when a document is being formatted, SCRIPT/VS gives you a message. Included with the message (if you have the TRACE option in effect on the SCRIPT command) is an indication of the line number in the source file where the problem was found. That's the place to look to see if your markup is okay.

- Messages resulting from an error in the code that interprets your markup

Unfortunately, no immediately apparent way exists to distinguish this kind of message from the kind described above.

When you have checked your markup and are sure that the error is not the result of something you've done wrong, take these messages *and* a copy of your source file to your document administrator or whoever in your organization is responsible for trouble shooting problems in formatting documents.

These messages may be confusing, because they generally involve an error in the use of a SCRIPT/VS control word. Because your markup consists of tags and not control words, you may wonder what is going on.

The secret is that all the formatting work is done with control words, which you don't see (unless something goes wrong). In most cases when SCRIPT/VS finds a tag, it actually interprets the GML by checking the profile (DSMPROF3) to see what is wanted. SCRIPT/VS then goes to another file, called a "macro library," and gets the set of control words that do the work for that tag. This set of control words is called an *application processing function* (APF).

Because of this process, it is easy for an organization to change the way things are formatted without having to change the markup in the source files. Just change the profile to point to a different APF or, alternatively, use a different profile to point to a different APF.

---

## Common Problems – Symptoms and Solutions

This section does not include problems we think you'll be able to figure out for yourself. For example, if a huge chunk of the document comes out highlighted, it may be traumatic, but you can still figure out for yourself that you omitted (or misspelled) an end tag for a highlighted phrase.

This section describes problems that are a little more subtle. Everything we tell you about here is something that has happened to us, and that's how we know.

---

### Symptom: Unexpected Formatting Results

*If you use **SCRIPT/VS** control words within your GML documents, you must be careful that they do not counteract the **SCRIPT/VS** control words that make the tags work as they do.*

For example, if you set indentation within a list, the indentation would interfere with the indentation already set by DSMPROF3 for lists. (See the section entitled "Control Words and Macros" in *DCF: Generalized Markup Language Starter Set Reference* for a list of SCRIPT/VS control words that can be used safely in GML documents.)

So, if you have unexpected results in your formatting, look for any control words (other than those in the list) you may have added to your source file and remove them.

---

### Symptom: Text Disappears

If pieces of text disappear from your output document, the most likely reason is that you have omitted a markup-content separator (the period at the end of the markup).

Many tags don't need the period, but some of them do. If a tag needs an ending period and you omit it, then any text following that tag is treated as an attribute and disappears from your document. What's even worse, if you leave out the period, some things that format correctly today may not work in the future because of changes in the profile you use. This is why you should *always* put in the period at the end of the markup.

If a lot of text disappears, you may be missing a PSC end tag.

Another thing that can cause text to disappear is an attribute value that contains blanks or special characters and is not enclosed in single quotation marks. The rest of the attribute value after the blank or special character doesn't get picked up as part of that attribute value.

---

**Symptom: Text Doesn't Format*****Throughout the Document:***

If your text and tags came out formatted in huge blocks, then your SCRIPT command did not specify the profile correctly and the tags weren't recognized at all.

If your tags don't show up in the text but the text still formats as one big block, then your SCRIPT command did not specify the library correctly. In this case, the tags were recognized, but SCRIPT/VS couldn't find the processing instructions to do the work.

This won't happen when you use DSMPROF3, because it checks to ensure the right library is there; if it isn't, it gives you a message and stops processing. Therefore, you don't waste a processing run that is sure to give you bad results. (We mention it here because it could happen if you use a profile other than DSMPROF3.)

***Starting in the Middle of the Document:***

When your formatting suddenly stops and the text comes out in big blocks, you probably have an extra XMP or FIG end tag in your input document. You will have to find it and delete it.

---

**Symptom: Column Is Narrow**

If you have this problem, it is almost certain that somewhere you have forgotten an end tag — usually for a list.

The starter set does a bit of “housekeeping” for head tags, and if it finds that a list, an example, or a figure is still going on, it ends it and gives you a message. However, if there are no more headings or sections in your document, this check can't be done for you.

It is a good practice to check the last couple of pages of your document to ensure that the column hasn't shrunk; if it has, then you have this problem. Look for an end tag near where the problem begins. If the end tag is missing, put one in.

---

**Symptom: The Colon after a Level 5 or 6 Heading Is Missing**

If you don't get a colon (:) after a level 5 or 6 heading, it's because you haven't included the P tag for the first paragraph following the heading. It is the P tag that tells the starter set to put in the colon. If your heading text is too long, or if other tags follow the heading tag, the colon may appear in an unpredictable location.

---

## Symptom: Page Number References Are Wrong

Sometimes the page numbers in the table of contents, the list of illustrations, the list of tables, or cross-references don't correspond to the pages where the headings, figures, or tables actually appear. If you are using two formatting passes, it is likely that the output pages from the first SCRIPT/VS pass through the document are numbered differently from the output pages from the second pass. Because the table of contents, list of illustrations, and list of tables are built on the first pass and then printed in your document at the beginning of the last pass, they have the page numbers from the first pass rather than the last pass.

The mismatch happens when things such as cross-references to headings are used before they are defined. For example, if you specify two formatting passes and you use a HDREF tag to refer to a heading that comes *later* in the book, the starter set doesn't know how much space to leave for that heading reference when it encounters the HDREF tag. When the starter set processes the second pass, it then knows how much space to leave for that heading reference and can place it correctly. A heading reference that is shorter or longer than the space left for it can cause the page numbers to be different from one pass to the next.

A quick fix for this problem is to put the TOC, FIGLIST, and TLIST tags at the end of the document (just before the GDOC end tag). This way, these elements are assigned the page numbers that were generated on the last pass rather than the first, and they will be correct. However, if you put these elements in the back matter, they will print in two-column format.

Another approach to this problem is to use FPASSES to specify more than two formatting passes. (See "FPASSES(n)" on page 126.)

---

## Symptom: Figures Are Out of Order

When a number of figures with different PLACE and WIDTH attributes get bunched together, they are sometimes printed out of order. This is because the figure numbers are assigned in the sequence in which the figures are found in the source file, but other things might cause the figures to be printed in a different sequence. For example, if you specify a figure with a PLACE attribute of TOP, and then follow it with a figure with a PLACE attribute of INLINE, the first figure can be floated past the second figure and thus be out of order.

The fix for this problem depends on your document. You may want to move some figures around or specify different PLACE attribute values on them.

For this reason, you should keep your figures in separate files. This makes them easy to move when necessary.

---

**Symptom: Blanks Disappear**

Sometimes, in an example or a figure, you have text followed by a string of blanks, which are followed by more text. If the last item in the first section of text was a tag (for example, to turn off highlighting), SCRIPT/VS may take out your blanks and move the last section of text up against the first.

You can fix this problem by replacing the first blank in the string of blanks with the &rbl. (required blank) symbol.

---

**Symptom: Semicolon Won't Print**

If a semicolon in your source document doesn't print and you get a line break instead, it's because SCRIPT/VS has misinterpreted this semicolon as a control-word separator.

Use the &semi. symbol in place of the semicolon that won't print and your problem is solved.

---

**Symptom: Front Matter Lists Are Empty**

If the table of contents, list of illustrations, or list of tables are empty, you've forgotten to specify FPASSES or TWOPASS in your SCRIPT command options.

---

**Symptom: Index Is Empty**

If you get a heading for your index but no entries, you've forgotten the INDEX option on the SCRIPT command.

Of course, you won't get an index if you forget your INDEX tag either, because without this tag there is no way for the starter set to know that you want an index, and it will make no place for it in your document.

---

**Symptom: Blank Inserted at GML Tag**

Sometimes an additional blank may be inserted in a line at the location of a GML tag. The blank might appear between the text before the tag and the text after the tag, or it might appear between text or white space that was inserted by the tag and text from the input line.

You can usually eliminate the blank by rearranging the input line or by breaking the line into two or more lines (moving the text that falls either to the left or the right of a tag to another line). If the line contains two tags, move one of the tags to a new line. If there are two tags on the same input line and the last tag on a line is followed by no text, try using a required blank (&rbl.) to the right of the tag.





---

## Chapter 15. Cross-Reference Listing

As we mentioned in Chapter 9, "Overall Document Structure" on page 83, DSMPROF3 produces a cross-reference listing at the end of your document unless you ask it not to at run time (SYSVAR X, as described in "Instructions to the Starter Set (SYSVAR)" on page 120). Normally, you would tear it off and throw it away before you published a book, but we have left it here (following the index) for you to examine.

The cross-reference listing includes headings, figures, footnotes, list items, tables, index entries, and an imbed trace.

The first six listings are similar. They show the following information for every case of a heading, figure, footnote, list item, table, or index entry that was entered with an ID attribute:

1. The name used on the ID attribute
2. The source file that contained it
3. The output page on which it appears
4. The output pages on which any references to it appear.

This assists you in checking for any markup errors in your cross-references.

If you'll recall, we had some unmatched REFID attributes in our chapter on cross-references; you'll see these listed with question marks in our cross-reference listing.

---

### The Imbed Trace

The imbed trace shows the files that were used in the formatting of your document and, through indentation, the level at which the file is imbedded. (See "Combining Input Files" on page 89 for information on imbedding files.)

The page number shown in the imbed trace is the page that SCRIPT/VS was processing when it encountered the .IM control word. This may be different from the page on which the output of that file occurs; for example, if the first thing in the imbedded file is an H1 tag, DSMPROF3 skips to the next page before it starts to print the contents of the imbedded file.

The primary input file (the document you named on your SCRIPT command) is not shown in the imbed trace.

If you look at the imbed trace for this book, you'll see the files that were imbedded when the document was formatted.



---

## Part Four: Tag Reference

This part contains additional information on how to enter GML tags, and it provides a detailed summary of the tags. This information is found in the following chapters:

- Chapter 16, "General Guidelines for Markup Entry" on page 145
- Chapter 17, "Alphabetic Summary of Tags" on page 147
- Chapter 18, "Summary by Document Elements" on page 151.



## Chapter 16. General Guidelines for Markup Entry

Following are some helpful guidelines for markup entry, based on the experience of many users of the GML starter set:

- Document elements are identified with GML tags. The tags begin with the GML delimiter, which in the starter set is the colon (:). End tags use the GML end tag delimiter, which in the starter set is the two characters ":e" (colon followed by a lowercase letter e).
- End the markup with a period.
- Start all input lines at the left margin.
- Begin a new input line for all tags except inline quotes, title references, cross-references, and highlighted phrases.
- When you end a sentence, do not enter any more text on that line.
- Do not end input lines with a hyphen.
- When using labeled attributes, enclose values that include blanks or special characters in single quotation marks. For example:

```
:gdoc sec='ABC Company Confidential'.
```

If the value itself includes a single quotation mark, then you must enter two of them so that SCRIPT/VS does not think the first one ends the value, for example:

```
:title stitle='Mrs. O''Leary''s Cow'.
```

The actual value that will then be used by SCRIPT/VS is

```
Mrs. O'Leary's Cow
```

- When you enter *decimal values* on attributes, enclose the values within a set of single quotation marks. If you don't, the starter set interprets the period as a *markup content separator* (not as a decimal point) and does not process the attribute value correctly.  
If you'd rather, you can enter such decimal values using a *comma* to mean a decimal point, and the starter set then processes the attribute appropriately.
- Do not start text lines with a period. If you do, SCRIPT/VS assumes the line is either a control word or a macro, gives you an error message if it is not, and does not print the line.
- Maximum input line length is 256 characters after all symbols are substituted.
- Do not mark an element with a tag that does not correctly describe it, even if it apparently results in the correct processing. The document administrator or text programmer may someday change the way that particular tag formats, and you may find that your document is formatted in a way you did not want.
- Enter tags and attribute labels in lowercase letters.
- Do not enter blank lines.

If you need a blank line in a figure or an example, just put an &rb1. symbol on the line you want to be blank.

## Guidelines for Markup Entry

- When storing parts of a document in separate files, be sure each file starts at the beginning of a paragraph unit or higher-level element and finishes at the end of one.

---

## Chapter 17. Alphabetic Summary of Tags

The table Table 10 on page 148 is an alphabetic summary of GML tags. The following are shown: each tag, its meaning, its attributes (if any), the allowable text contents, and the end tag (if any).

Required attributes are in italics. The other attributes are optional.

In the contents column, you will find one of these entries:

**Text on same line or next line; no other tags.**

All the text must be entered either on the same line as the end of the markup or on the next line; no tags, such as those for highlighting or quotes, can be used within the text.

The tag should appear at the beginning of the input line.

**No immediate text.**

There is no text directly associated with this tag; that is, the tag is always followed by another tag.

The tag should appear at the beginning of the input line.

**Implied paragraph.**

The text associated with this tag is an implied paragraph. The text can start either on the same line as the tag or on the next line and can continue for as many lines as needed.

The tag should appear at the beginning of the input line.

**Any text; anywhere on line.**

These tags can appear anywhere within the input line, and the text can be anything that is reasonable in the context of that tag. (You wouldn't, for example, put a figure in the middle of a title reference.)

**Self-contained.**

These tags can appear anywhere within the input line, and the content of the tag is contained entirely within the tag (plus attributes) itself; that is, it has no effect on the following text.

The tags are shown with the starter set default delimiters. (Your organization may use different delimiters.)



## Summary of Tags

Tag	Element	Attributes	Content	Termination
:ABSTRACT	abstract		No immediate text.	
:ADDRESS	address		No immediate text.	:EADDRESS
:ALINE	address line		Text on same line or next line; no other tags.	
:APPENDIX	appendix section		No immediate text.	
:AUTHOR	author of document		Text on same line or next line; no other tags.	
:BACKM	back matter		No immediate text.	
:BODY	body of document		No immediate text.	
:C	cell of a table	(cell number)	Implied paragraph.	:EC
:CIT	title citation		Any text; anywhere on line.	:ECIT
:DATE	date		If any text, must be on same line or next line; no other tags.	
:DD	definition description		Implied paragraph.	
:DDHD	definition description heading		Text on same line or next line; no other tags.	
:DL	definition list	BREAK COMPACT HEADHI = TERMHI = TSIZE =	No immediate text.	:EDL
:DOCNUM	document number		Text on same line or next line; no other tags.	
:DT	definition term		Text on same line or next line; no other tags.	
:DTHD	definition term heading		Text on same line or next line; no other tags.	
:FIG	figure	DEPTH = FRAME = ID = PLACE = WIDTH =	No immediate text.	:EFIG
:FIGCAP	figure caption		Text on same line or next line; no other tags.	
:FIGDESC	figure description		Implied paragraph.	
:FIGLIST	figure list		No immediate text.	
:FIGREF	figure reference	PAGE = REFID =	Self-contained.	
:FN	footnote	ID =	Implied paragraph.	:EFN
:FNREF	footnote reference	REFID =	Self-contained.	
:FRONTM	front matter		No immediate text.	
:GD	glossary definition		Implied paragraph.	
:GDOC	general document	SEC =	No immediate text.	:EGDOC
:GL	glossary list	COMPACT TERMHI =	No immediate text.	:EGL

Tag	Element	Attributes	Content	Termination
:GT	glossary term		Text on same line or next line; no other tags.	
:HDREF	head reference	PAGE= REFID=	Self-contained.	
:HP0--:HP3	highlighted phrases 0-3		Any text; anywhere on line.	:EHP0--:EHP3
:H0--:H1	head levels 0-1	ID= STITLE=	Text on same line or next line; no other tags.	
:H2--:H6	head levels 2-6	ID=	Text on same line or next line; no other tags.	
:IH1--:IH3	index entry heading levels 1-3	ID= PRINT= SEE= SEEID=	Text on same line or next line; no other tags.	
:INDEX	index		No immediate text.	
:IREF	index entry reference	REFID= PG= SEE= SEEID=	Self-contained.	
:I1	index entry term level 1	ID= PG=	Text on same line or next line; no other tags.	
:I2--:I3	index entry term levels 2-3	ID= REFID= PG=	Text on same line or next line; no other tags.	
:LI	list item	ID=	Implied paragraph.	
:LIREF	list item reference	PAGE= REFID=	Self-contained.	
:LP	list part		Implied paragraph.	
:LQ	long quote (excerpt)		No immediate text.	:ELQ
:NOTE	note		Implied paragraph.	
:OL	ordered list	COMPACT	No immediate text.	:EOL
:P	paragraph		Paragraph.	
:PC	paragraph continuation		Implied paragraph.	
:PREFACE	preface		No immediate text.	
:PSC	process specific controls	PROC=	No immediate text.	:EPSC
:Q	quote		Any text; anywhere on line.	:EQ
:RDEF	row definition	ID= HP= ALIGN= CONCAT= VALIGN= ROTATE= MINDEPTH= ARRANGE= CWIDTHS=	No immediate text.	

## Summary of Tags

Tag	Element	Attributes	Content	Termination
:ROW	row	REFID = SPLIT =	No immediate text.	:EROW
:SL	simple list	COMPACT	No immediate text.	:ESL
:TABLE : c.table	REFID = ID = COLUMN PAGE SPLIT = ROTATE = WIDTH =	No immediate text.	:ETABLE	
:TCAP	table caption		Text on same line or next line; no other tags.	
:TDESC	table description		Implied paragraph.	
:TFT	table footing	REFID =	No immediate text.	:ETFT
:THD	table heading	REFID =	No immediate text.	:ETHD
:TITLE	title of document	STITLE =	Text on same line or next line; no other tags.	
:TITLEP	title page		No immediate text.	:ETITLEP
:TLIST	table list		No immediate text.	
:TNOTE	table notes		Implied paragraph.	:ETNOTE
:TOC	table of contents		No immediate text.	
:TREF	table reference	REFID = PAGE =	Self-contained.	
:UL	unordered list	COMPACT	No immediate text.	:EUL
:XMP	example	DEPTH =	No immediate text.	:EXMP

Table 10. Alphabetic Summary of Starter Set Tags.

## Chapter 18. Summary by Document Elements

<i>Major Document Elements</i>	<i>Document Type</i>		
		general document	:GDOC SEC= :EGDOC
	<b>Front Matter</b>	front matter	:FRONTM
		title page	:TITLEP :ETITLEP
		title	:TITLE STITLE=
		document number	:DOCNUM
		date	:DATE
		author	:AUTHOR
		address	:ADDRESS :EADDRESS
		address line	:ALINE
		abstract	:ABSTRACT
		preface	:PREFACE
		table of contents	:TOC
		figure list	:FIGLIST
		table list	:TLIST
	<b>Body</b>	body	:BODY
	<b>Appendixes</b>	appendix section	:APPENDIX
	<b>Back Matter</b>	back matter	:BACKM

Table 11. Major Document Elements

<i>Indexing</i>		
	index	:INDEX
	index entry term (levels 1-3)	:I1 - :I3 ID = PG = REFID =
	index entry heading (levels 1-3)	:IH1 - :IH3 ID = PRINT = SEE = SEED =
	index entry reference	:IREF REFID = PG = SEE = SEED =

Table 12. Indexing

## Summary of Document Elements

<b>Headings</b>	head level 0-1	:H0 – :H1 ID = STITLE =
	head level 2 – 6	:H2 – :H6 ID =
	head reference	:HDREF PAGE = REFID =
	table of contents	:TOC

Table 13. Headings

<b>Basic Text</b>	paragraph	:P
	paragraph continuation	:PC
	highlighted phrase 0 – 3	:HP0 – :HP3 :EHP0 – :EHP3
	title citation	:CIT :ECIT
	address	:ADDRESS :EADDRESS
	address line	:ALINE
	note	:NOTE
	quotation (inline)	:Q :EQ
	quotation (excerpt)	:LQ :ELQ

Table 14. Basic Text

<b>Examples and Figures</b>	<b>Examples</b>	example	:XMP DEPTH = :EXMP
	<b>Figures</b>	figure	:FIG DEPTH = FRAME = ID = PLACE = WIDTH = :EFIG
		figure caption	:FIGCAP
	figure description	:FIGDESC	
	figure reference	:FIGREF PAGE = REFID =	
	figure list	:FIGLIST	

Table 15. Examples and Figures

<b>Tables</b>	defining the table	:RDEF ID= HP= ALIGN= CONCAT= VALIGN= ROTATE= MINDEPTH= ARRANGE= CWIDTHS=
	building the table	:TABLE REFID= ID= COLUMN PAGE SPLIT= ROTATE= WIDTH= :ETABLE
	row	:ROW REFID= SPLIT= :EROW
	cell	:C (cell number) :EC
	table heading	:THD REFID= :ETHD
	table footing	:TFT REFID= :ETFT
	table reference	:TREF REFID= PAGE=
	table caption	:TCAP
	table description	:TDESC
	table notes	:TNOTE :ETNOTE
	table list	:TLIST

Table 16. Tables

<b>Footnotes</b>	footnote	:FN ID= :EFN
	footnote reference	:FNREF REFID=

Table 17. Footnotes

# Summary of Document Elements

<b>Process-Specific</b>	process-specific controls	:PSC PROC= :EPSC
-------------------------	---------------------------	------------------------

**Table 18. Process Specific Controls**

<b>Lists</b>	<b>Simple</b>	<b>simple list</b>	<b>:SL COMPACT :ESL</b>
		list item	:LI ID=
	<b>Unordered</b>	unordered list	:UL COMPACT :EUL
		list item	:LI ID=
	<b>Ordered</b>	ordered list	:OL COMPACT :EOL
		list item	:LI ID=
	<b>Definition</b>	definition list	:DL COMPACT BREAK HEADHI= TERMHI= TSIZE= :EDL
		definition term heading	:DTHD
		definition description heading	:DDHD
		definition term	:DT
		definition description	:DD
	<b>Glossary</b>	glossary list	:GL COMPACT TERMHI= :EGL
		glossary term	:GT
		glossary description	:GD
	<b>List Part</b>	list part	:LP
	<b>List Item Reference</b>	list item reference	:LIREF REFID=

**Table 19. Lists**

---

## Part Five: Appendixes

This part contains the following appendixes:

- Appendix A, "A Sample GML Document" on page 157
- Appendix B, "Solutions for Exercises" on page 169





---

## Appendix A. A Sample GML Document

The following pages contain a sample document illustrating the use of the GML starter set. The first part of this appendix shows the text and markup of the document; the second part shows the resulting output. We didn't incorporate all of the tags described in this book, but enough of them are shown to give you an idea of how GML works.

**Note:** We used the following SCRIPT command options to run our sample document named SAMPDOC:

```
SCRIPT SAMPDOC MESSAGE(I D) TW SYS(S 0) INDEX PROF(DSMPROF3)
```

```
:gdoc sec='Company Confidential'.
:frontm.
:titlep.
:title stitle='A Short GML Document'.
A Short GML Document Demonstrating the Document Composition Facility
:date.
:author.C. H. Scott
:address.
:aline.77 Cattle Drive
:aline.Silverado, Colorado
:eaddress.
:etitlep.
:toc.
:figlist.
:tlist.
:body.
:h1.Introduction to the Generalized Markup Language
:ih1 id=gml.Generalized Markup Language
:i2.introduction to
:ih1 seeid=gml.GML
:p.The practice of generalized markup of source documents
has significant benefits not only for SCRIPT/VS processing, but also
facilitates many other applications.
:h2.What Is Generalized Markup?
:p.To mark up a source document is to add information to
it that
:ih1 id=mark.markup
:i2.definition of
enables a person or system to process it in some way.
The added information or markup can be instructions called
control words, or indications, or descriptions of some kind.
Document markup is the primary means of instructing a
computerized text processing system, such as SCRIPT/VS,
how a document is to be processed.
:p.A document can be marked up in two ways:
:ol.
:li.With specific markup, which limits text processing
:i2.specific
to a particular application.
:li.With generalized markup, which describes the contents
:i2.generalized
of a source document, without regard to particular processing.
The language used for generalized markup (GML, for Generalized
```

Markup Language) is descriptive; it provides the syntax and usage rules for developing your own vocabulary of tags for **:hp2.describing:ehp2**.

**:i1.tags :i2 refid=gml.tags**  
the parts of a document.

**:fn.**

A starter set of GML tags is provided with the Document Composition Facility to allow the user to get going.

**:efn.**

**:eol.**

**:p.**A GML tag identifies the associated text as a particular document element.

For example:

**:ul.**

**:li.**A book might have the major divisions:

**:ul.**

**:li.**front matter

**:li.**body

**:li.**back matter

**:eul.**

**:p.**Within those divisions we can have paragraphs, examples, figures, list items, and so on.

**:li.**A memo might have document elements of addressee, date, sender, address, subject, and reference, as well as other types similar to those of a book.

**:eul.**

**:p.**You might ask, **:q.**Why use generalized markup instead of specific markup:**:eq.?**

Well, thinking in terms of the content or purpose of the parts of a document is easier than thinking in terms of how the parts should be processed or how they should appear when printed.

A person doing generalized markup can concentrate on the text, without thinking about format.

**:p.**There are benefits of generalized markup when your only application will be text processing with SCRIPT/VS and others when you anticipate additional applications.

**:fig id='abc' width='4i' depth='10p6' frame='box'.**

I've just drawn a little figure here where I've left 10 picas and 6 points of space before the text of the figure started. I've set my width to four inches -- if I print out my document with a column width of four inches or larger, my figure will be column width, if I print out my document with a column width of less than four inches, my figure will extend the width of the page dimension.

**:figcap.**A Sample Figure

**:figdesc.**This figure also gives you a little bit of information on how the WIDTH attribute of the FIG tag works.

**:i1.**figure example

**:i2 refid=mark.**of a figure

**:efig.**

**:h3.**Potential Benefits of GML for SCRIPT/VS Text Processing

**:p.**Here are some potential benefits in using GML when you intend to use SCRIPT/VS only for text processing:

```

:ul.
:li.Alternative GML interpretation.
A GML tag need not be limited to a single SCRIPT/VS
interpretation.
For example, a tag might indicate that a group of
words in the text is a title.
For one application, you might want titles to be enclosed
in quotation marks.
But for another application, you might want titles
to be underlined or set in italics.
Each application could be satisfied by alternative GML
interpretations, :hp1.with no change:ehp1. to the source
document or to the markup of titles.
:i2 refid=gml.potential benefits of
As described in a later section in this chapter, you
have control over the way GML is interpreted.
:li.Ease of markup.
It is easy to remember GML tags because they can
consist of terms and abbreviations commonly
used to describe a document.
GML generally requires fewer characters to be entered
for markup than a corresponding complex sequence of
control words.
The result is faster markup and keying of the document.
Changes to the markup are also faster and may even be
eliminated with GML, because you can control how GML
is interpreted.
:li.Ease of text update.
It's easy to update text marked up with GML.
With GML, such things as the numbering of items in a
numbered list is left to the formatter, which numbers
the items automatically.
Thus, when you insert or delete an item, you don't
have to renumber subsequent items, as you would have to
if the numbers were part of the source text.
:eul.
:rdef id=corptbl cwidths='1i * * * * *'
arrange='1 2 3 3 3 3'
arrange='1 2 4 5 6 6'
arrange='1 2 4 5 7 8'
align='left left center center center left center'
concat='yes yes no yes'
valign='center center top center center top bottom'.
:table refid=corptbl id=tbtest.
:i1.table example
:i2 refid=mark.of a table
:row.
:c.Selected major industry and selected country
:c.Number of U.S. corporation returns
:c.Controlled Foreign Corporations
:c.Number of foreign corporations
:c.Total assets of corp.
:c.Foreign corporations with current earnings and
profits (+) before taxes
:c.Current
earnings and
profits before taxes
:c.Foreign income taxes (net)
:erow.

```

**:tcap.**A Sample Table  
**:etable.**  
**:backm.**  
**:index.**  
**:egdoc.**

**A SHORT GML DOCUMENT  
DEMONSTRATING THE  
DOCUMENT COMPOSITION  
FACILITY**

*September 15th, 1989*

C. H. Scott

77 Cattle Drive  
Silverado, Colorado

*Company Confidential*

Figure 3. Example of a Title Page

*COMPANY CONFIDENTIAL*

## **Table of Contents**

<b>Introduction to the Generalized Markup Language</b> .....	<b>1</b>
What Is Generalized Markup? .....	1
Potential Benefits of GML for SCRIPT/VS Text Processing .....	2
<b>Index</b> .....	<b>4</b>

Table of Contents

ii

Figure 4. Example of a Table of Contents

## **List of Illustrations**

Figure 1. A Sample Figure ..... 2

Figure 5. Example of a Figures List



## List of Tables

Table 1. A Sample Table ..... 3

Figure 6. Example of a Tables List

## Introduction to the Generalized Markup Language

The practice of generalized markup of source documents has significant benefits not only for SCRIPT/VS processing, but also facilitates many other applications.

### *What Is Generalized Markup?*

To mark up a source document is to add information to it that enables a person or system to process it in some way. The added information or markup can be instructions called control words, or indications, or descriptions of some kind. Document markup is the primary means of instructing a computerized text processing system, such as SCRIPT/VS, how a document is to be processed.

A document can be marked up in two ways:

1. With specific markup, which limits text processing to a particular application.
2. With generalized markup, which describes the contents of a source document, without regard to particular processing. The language used for generalized markup (GML, for Generalized Markup Language) is descriptive; it provides the syntax and usage rules for developing your own vocabulary of tags for **describing** the parts of a document.<sup>1</sup>

A GML tag identifies the associated text as a particular document element. For example:

- A book might have the major divisions:
  - front matter
  - body
  - back matter

Within those divisions we can have paragraphs, examples, figures, list items, and so on.

- A memo might have document elements of addressee, date, sender, address, subject, and reference, as well as other types similar to those of a book.

You might ask, "Why use generalized markup instead of specific markup"? Well, thinking in terms of the content or purpose of the parts of a document is easier than thinking in terms of how the parts should be processed or how they should appear when printed. A person doing generalized markup can concentrate on the text, without thinking about format.

There are benefits of generalized markup when your only application will be text processing with SCRIPT/VS and others when you anticipate additional applications.

---

<sup>1</sup> A starter set of GML tags is provided with the Document Composition Facility to allow the user to get going.

Figure 7. Example of Text with Headings

I've just drawn a little figure here where I've left 10 picas and 6 points of space before the text of the figure started. I've set my width to four inches -- if I print out my document with a column width of four inches or larger, my figure will be column width, if I print out my document with a column width of less than four inches, my figure will extend the width of the page dimension.

**Figure 1. A Sample Figure:** This figure also gives you a little bit of information on how the WIDTH attribute of the FIG tag works.

### Potential Benefits of GML for SCRIPT/VS Text Processing

Here are some potential benefits in using GML when you intend to use SCRIPT/VS only for text processing:

- **Alternative GML interpretation.** A GML tag need not be limited to a single SCRIPT/VS interpretation. For example, a tag might indicate that a group of words in the text is a title. For one application, you might want titles to be enclosed in quotation marks. But for another application, you might want titles to be underlined or set in italics. Each application could be satisfied by alternative GML interpretations, *with no change* to the source document or to the markup of titles. As described in a later section in this chapter, you have control over the way GML is interpreted.
- **Ease of markup.** It is easy to remember GML tags because they can consist of terms and abbreviations commonly used to describe a document. GML generally requires fewer characters to be entered for markup than a corresponding complex sequence of control words. The result is faster markup and keying of the document. Changes to the markup are also faster and may even be eliminated with GML, because you can control how GML is interpreted.
- **Ease of text update.** It's easy to update text marked up with GML. With GML, such things as the numbering of items in a numbered list is left to the formatter, which numbers the items automatically. Thus, when you insert or delete an item, you don't have to renumber subsequent items, as you would have to if the numbers were part of the source text.

Figure 8. Example of Text with a Figure

**COMPANY CONFIDENTIAL**

Selected major industry and selected country	Number of U.S. corporation returns	Controlled Foreign Corporations			
		Number of foreign corporations	Total assets of corp.	Foreign corporations with current earnings and profits (+) before taxes	
				Current earnings and profits before taxes	Foreign income taxes (net)

**Table 1. A Sample Table**

**Figure 9. Example of a Table**

## Index

### F

figure example 2

### G

Generalized Markup Language

introduction to 1

potential benefits of 2

tags 1

GML

See Generalized Markup Language

### M

markup

definition of 1

generalized 1

of a figure 2

of a table 3

specific 1

### T

table example 3

tags 1

Figure 10. Example of an Index

---

## Appendix B. Solutions for Exercises

This appendix contains the markup for the exercises that are included in this book.

---

### Markup for Paragraphs and Headings

The following markup is for "Exercise: Paragraphs and Headings" on page 15.

```
:h3.This is a Head Three
:p.It has two short paragraphs
following it.
This is the first paragraph.
Each sentence was started on a
new line.
:p.This is the second paragraph.
Look to see how DSMPROF3 has
formatted it (a blank line before,
and no indention).
:h5.This is a Head Five
:p.The paragraph following it
begins on the same line as the
head, even though it was entered
on a separate line.
:p.The next paragraph under the
head five formats like this.
```

---

## Markup for Lists

The following markup is for "Exercise: Lists" on page 29.

```
:p.Please send the following items:
:dl tsize='11'.
:dtth.Number
:ddhd.Description
:dt.A107C2
:dd.Class A Widget, 100 x 200 mm
in assorted colors (no white).
:dt.B321DJ
:dd.Brown and yellow Whuzzit,
with linkage for optional nutcracker
attachment.
:dt.BY7532
:dd.Assorted gimcracks, with the
following characteristics:
:ul.
:li.Soft
:li.Cuddly.
:eul.
:edl.
:p.When I get around to it, I will
:ol.
:li.Balance my checkbook.
:li.Prune the shrubbery.
:li.Write to the following:
:s1 compact.
:li.My mother in New Rochelle
:li.My great aunt in Detroit
:li.My niece in Ossining
:li.My friend in South Bend.
:es1.
:li.Clean out the garage.
:eol.
```

---

## Markup for Examples and Figures

The following markup is for "Exercise: Examples and Figures" on page 43.

**:p.**Here's an example of some BASIC statements:

**:xmp.**

```
10 PRINT USING 55 A, B, C
```

```
20 LET J = K + 2
```

```
30 IF J = X GO TO 80
```

**:emp.**

**:pc.**that will solve this problem.

**:fig place=inline width=page frame=box.**

AN INLINE, PAGE-WIDE FIGURE

Because the contents of a figure format EXACTLY as entered, you

can enter blanks on the line (before text) and the

lines will print exactly the same as they were entered!

**:figcap.**An Inline, Page-Wide Figure

**:figdesc.**This is the first figure I have entered myself.

**:efig.**

**:p.**This paragraph follows the FIG end tag.

Here we have another figure (inline and column wide):

**:fig place=inline width=column.**

Let's create another figure that is column wide, which will create a second item for a list of illustrations in a future exercise.

**:figcap.**A Column-Wide Figure

**:efig.**



---

## Markup for Highlighting, Citing, Noting, and Quoting

The following markup is for "Exercise: Highlighting, Citing, Noting, and Quoting" on page 50.

**:p**.Print **:hp2**.boldly:**ehp2**. on  
the 3800, **:hp2**.strikingly:**ehp2**. on the 1403.  
**:p**.Title references are **:hp1**.also:**ehp1**.  
used for movie titles, such as  
**:cit**.Casablanca:**ecit**..  
Humphrey Bogart is often misquoted  
as having said **:q**.Play it again,  
Sam:**eq**. in that movie.  
I wish people would learn to  
quote **:hp3**.accurately:**ehp3**..  
**:p**.If I said, **:q**.he said **:q**.yes:**eq**.  
to me,**eq**. that would be an example  
of a nested quote.  
**:p**.Look at this excerpt  
from the **:cit**.GML User's Guide:**ecit**.:  
**:lq**.  
**:p**.You may well ask, **:q**.Why  
use quote tags at all?:**eq**.  
**:p**.We now have a variety of output devices  
on which to print our documents.  
Some of these devices print using  
fonts that distinguish among open quotation marks,  
close quotation marks, and apostrophes.  
By using the correct markup, our files  
can be printed on any of the  
available devices with no change to the markup.  
**:elq**.  
**:note**.Notes lose their impact if used excessively.

---

## Markup for Tables

The following markup is for "Exercises: Tables" on page 70.

### First Table Exercise

```
:rdef id=autohdr cwidths=',5i * * *' valign=bottom
      hp='2 2 2 3' align='left center center right'.
:rdef id=auto cwidths=',5i * * *' align='left right'
:table column refid=auto width='3i'
:thd refid=autohdr.
:c.Year:c.Passenger Cars:c.Trucks and Buses:c.Total
:ethd.
:row refid=auto.
:c.1900:c.4,192:c 4.4,192
:c.1905:c.24,250:c.750:c.25,000
:c.1910:c.181,000:c.6,000:c.187,000
:c.1915:c.895,930:c.74,000:c.969,930
:c.1920:c.1,905,560:c.321,789:c.2,227,349
:tcap.Example Table Showing U.S. Automobile Production
:tdesc.The above table shows the number of automobiles
produced in the United States from the years 1900 through 1920.
:etable.
```

### Second Table Exercise

```
:rdef id=flag arrange='1 2 / 1 3 / 1 4 / 1 5 / 1 6 / 1 7 / 1 8 '
      arrange='9 9 / 10 10 / 11 11 / 12 12 / 13 13 / 14 14'
      align='c r' valign='c t'.
:table column refid=flag.
:row.
:c.Flag
:c.Connecticut:c.Delaware
:c.Georgia:c.Maryland
:c.Massachusetts:c.New Hampshire
:c.New Jersey:c.New York
:c.North Carolina:c.Pennsylvania
:c.Rhode Island:c.South Carolina
:c.Virginia
:etable.
```

---

## Markup for Cross-References and Footnotes

The following markup is for “Exercise: Cross-References and Footnotes” on page 80.

```
:p.This example shows cross-references to:  
:ol.  
:li.A figure  
:li.A footnote  
:li id=three.A list item  
:li.A table  
:eol.  
:p.Here's a cross-reference to  
:figref refid=exf4., so remember to run with the FPASSES  
or the TWOPASS option to see the correct results.  
:fn id=exer4.  
Of course, when you run the exercise, the footnote will  
be numbered 1, because it is the first footnote in the  
document.  
:efn.  
And here we have a footnote  
reference.:fnref refid=exer4.  
:fig place=inline id=exf4 frame=box width=18p.  
This is the figure being referred to.  
:figcap.Figure Being Referred To  
:efig.  
:p.Don't forget to put an ID attribute on item  
:liref refid=three. above so that this reference to  
it resolves correctly.  
:p.Finally, we want to refer to :tref refid=jb..  
:rdef id=xwins align=center cwidths='2i 1i 1i'.  
:table column refid=xwins id=jb width='4i'.  
:row.  
:c.Here's  
:c.the  
:c.table  
:row.  
:c.being  
:c.referred  
:c.to  
:tcap.Table Cross Reference Exercise  
:etable.
```

---

## Glossary

The glossary defines words and phrases that have special meanings in SCRIPT/VS or special meanings in a typographical sense. The terms are defined as they are used in this book. If you do not find the term you are looking for, refer to the index or to the *IBM Dictionary of Computing*, SC20-1699.

This glossary includes definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the American National Dictionary for Information Processing, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

### A

**advanced function printing (AFP).** The ability of licensed programs to use the all-points-addressable concept to print text and images on a printer.

**alignment.** The horizontal placement of text in a column or cell.

**all-points addressability.** The capability to address, reference, and position text, overlays, and images at any defined point on the printable area of a sheet. See *page device* and contrast with *line device*.

**alphanumeric string.** A sequence of characters consisting solely of the letters a through z and the numerals 0 through 9.

**ampersand.** The & character. When an ampersand begins a character string, SCRIPT/VS assumes the character string is a symbol name. If the symbol name is defined, SCRIPT/VS replaces the symbol with its value (unless symbol substitution is off). In running footings, running headings, and running titles, the ampersand is usually the page number symbol. When encountered by itself on the right side of a .SE [Set Symbol] control word, it is interpreted as the page number symbol.

**application processing function (APF).** In GML processing, the processing that is performed when a document element or attribute is recognized. In SCRIPT/VS, an APF is implemented as a sequence of control words, possibly intermixed with text and symbols, in one of three forms: macro definition, value of a symbol, or imbedded file.

**application programming interface (API).** An external, published interface that can be programmed to by another application.

**ascender.** (1) In a font, the distance from the baseline to the top of the character. See *maximum ascender*. (2) The part of a lowercase letter that rises above the body of the letter. Letters with ascenders are b, d, f, h, k, l, and t.

**ATMS-II.** Advanced Text Management System.

**attribute.** A characteristic of a document (or document element) other than its type or content. For example, the security level of a document or the depth of a figure.

**attribute label.** In GML markup, a name of an attribute that is entered in the source document when specifying the attribute's value.

### B

**back matter.** In a book, those sections (such as glossary and index) that are placed after the main chapters or sections.

**balancing.** In multicolumn formatting, the process of making column depths on a page approximately equal by re-distributing the text in the columns. See also *vertical justification*.

**baseline.** An imaginary horizontal line that most of the letters in a line of text appear to rest on.

**basic document element.** In a general document, one of a group of elements that occurs frequently; for example: note, paragraph, and definition list.

**batch environment.** The environment in which non-interactive programs are executed.

**binding edge.** The edge of a page to be bound, stapled, or drilled. Defined with the BIND option of the SCRIPT command.

**body.** (1) Of a printed page, that portion between the top and bottom margins that contains the text. (2) Of a book, that portion between the front matter and the back matter.

**boldface.** A heavy-faced type. Also, printing in this type.

**bottom margin.** On a page, the space between the body or the running footing, if any, and the bottom edge of the page.

## Glossary

**break.** An interruption in the formatting of input lines so that the next input line is printed on a new output line.

**bug.** An error in a program on in a document markup.

### C

**call.** Used in reference to macros. It means to invoke the macro.

**caps.** Capital letters. See also *initial caps*.

**caption.** Text accompanying and describing an illustration.

**case-sensitive.** Whether a group of letters is uppercase or lowercase a group of letters is uppercase or lowercase has relevance. ABC is different from Abc which is different from ABC.

**CDPF.** Composed Document Printing Facility.

**cell.** A single unit within a table in which text or other expressions may appear. A cell is always rectangular and usually bounded by horizontal and vertical rules.

**centimeter (cm).** A measurement equal to 0.39 inch. 100 cm = 1 meter (m).

**chapter.** In a general document, a name given to a first-level heading segment that occurs within the body of a document. See also *heading segment*.

**character.** A symbol used in printing. For example, a letter of the alphabet, a numeral, a punctuation mark, or any other symbol that represents information.

**character arrangement table.** Translates input data into printable characters and identifies associated character sets and graphic character modification modules.

**character set.** A finite set of different characters that is agreed to be complete for some purpose. For example, in printing, the characters that constitute a font.

**character space.** The horizontal size of a character. This size depends upon which font the character is from and on which physical device the character is printed.

**character spacing.** The space between characters in a word.

**cicero.** In the Didot point system, a unit of 0.1776 inch (4.512 millimeters) used in measuring typographical material.

**CMS.** Conversational Monitor System—an interactive processor that operates within VM/370.

**code page.** A font library member name that gives the association between code points and the character names of a font.

**code point.** An eight-bit binary code representing one of 256 possible characters.

**coded font.** (1) The combination of a code page and a font library. (2) A font that is fully described in terms of typeface, point-size, weight, width, and attribute.

**column.** A vertical arrangement of characters or other expressions on a printed page.

**column balancing.** The process of redistributing lines of text among a set of columns so that the amount of text in each column is as equal as possible.

**column width.** The width of each text column on a page. Specified with the .CL [Column Line Length] control word. (In multicolumn formatting, all columns on a page usually have the same width.)

**command.** A request from a terminal or a request specified in a batch processing job for the performance of an operation or the execution of a particular program. For example, a request given at a terminal for SCRIPT/VS to format a document or for an editor to edit a line of text.

**comment.** A control word line that is ignored by SCRIPT/VS. Such lines begin with either \* or .cm.

**composed text.** Text that has been formatted and that contains control information to direct the presentation of the text on page printers.

**composite.** The act or result of formatting a document.

**composite rotation.** The total amount of rotation done by the printer to place text in the correct orientation on the page.

To determine the composite rotation, add all the current rotations, such as the rotation of the page as specified with the DEVICE command option, the rotation of the current area (as specified on the .DA [Define Area] control word), the rotation of the current table (as specified in the TABLE parameter of the .TD [Table Definition] control word), and the rotation of the contents of the current cell (as specified in the CELL parameter of the .TD [Table Definition] control word). For example: if the page rotation is 90 and there is no current area, and the table rotation is 180, and the rotation of the contents of the cell is 180; then the composite rotation of the table would be 270 and the composite rotation of the contents of the cell would be 90.

**compositor.** A person or program that composes text.

**concatenation.** The forming of an output line that contains as many words as the column width allows, by placing the first words from an input line after the last words from the preceding input line. When words from an input line would reach beyond the right margin and hyphenation cannot be performed, they are placed at the beginning of the next output line, and so on.

**control word.** An instruction within a document that identifies its parts or tells SCRIPT/VS how to format the document. See also *macro*.

**control word line.** An input line that contains at least one control word.

**control word statement.** A control word and its parameters.

**copy group.** A portion of a FORMDEF that defines a set of modifications that can be used when printing a page.

**current left margin.** The left limit of a column that is in effect for formatting. Each column's left margin is specified with the .CD [Column Definition] control word. However, the current left margin (that is, the left boundary for an output line) might vary to the right of the column's left margin when indentation is changed with the .IN [Indent], .UN [Undent], .IL [Indent Line], and .OF [Offset] control words.

**current line.** The line in a source document at which a computer program (such as an editor or a formatter) is positioned for processing.

## D

**debug.** To detect, trace, and eliminate errors in computer programs and SCRIPT/VS documents.

**default value.** A value assumed by a computer program when a control word, command, or control statement with no parameters is processed. In GML processing, the value assumed for an attribute when none is specified.

**descender.** (1) In a font, the distance from the baseline to the bottom of the character. See *maximum descender*. (2) The part of a letter that falls below the body of the letter. Letters with descenders are g, j, p, q, y, and Q.

**destination.** The physical device to which data is sent.

**dictionary.** A collection of *word stems* that is used with the spelling verification and automatic hyphenation functions.

**Didot point system.** A standard printer's measurement system on which type sizes are based. A Didot point is 0.0148 inch (0.376 millimeter). There are 12 Didot points to a cicero. See also *cicero* and *point*.

**document.** (1) A publication or other written material. (2) A machine-readable collection of lines of text or images, usually called a source document. See also *output document* and *source document*.

**document administrator.** One who is responsible for defining markup conventions and procedures for an organization.

**document conversion processor.** A computer program that processes a machine-readable document that includes formatting controls written in one formatter language, to produce a machine-readable document that includes formatting controls appropriate for another formatter language.

**document library.** A set of VSAM data sets, accessible in a batch environment, which contain documents and related files.

**dot leader.** A set of periods that fills in the space between two pieces of split text such as a chapter title and its page number in a table of contents.

**duplex.** A mode of formatting appropriate for printing on both sides of a sheet.

## E

**EBCDIC.** Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

**edit.** To create or modify the contents of a document or file. For example, to insert, delete, change, rearrange, or copy lines.

**editor.** A computer program that processes commands to enter lines into a document or to modify it.

**eject.** In formatting, a skip to the next column or page.

**element.** Any part of a document: a single character or a word or a sentence. Also refers to any part of a document you can identify with a GML tag (tagged element), such as a paragraph or figure or heading.

**em.** A unit of measure usually equal to the width or the height of the character "m" in a particular font.

**en.** A unit of measure usually equal to one-half the width of an em. For many typefaces, lowercase characters tend to average the width of an en.

## Glossary

**enable.** Used in reference to a tag. Means that the tag is mapped to its appropriate APF.

**epifile.** The second portion of a profile (after a .EF control word) that is processed *after* a main document has been processed.

**escapement.** The space unit of movement (either vertical or horizontal) that is built into a physical device. For the 1403, with a 10-pitch train, the horizontal escapement unit is 1/10th of an inch; for the 4250, that value is 1/600th of an inch; and for the 3800 Model 3 and the 3820 Page Printer, that value is 1/240th of an inch.

**extended symbol processing.** The processing of a symbol whose value causes the remainder of the line to be stacked and later processed as a new input line.

## F

**factor.** A dimensionless scalar value used to form a product with another value. Factors can also be expressed as percentages.

**figure space.** (1) The width of the figure zero (0) is commonly used as the figure space of a given typeface. This is the definition figure space as it is used in DCF. (2) A unit of measure equal to the width of the "en" space in a particular font.

**fill character.** The character that is used to fill up a space; for example, blanks used to fill up the space left by tabbing.

**float.** (1) (noun) A keep (group of input lines kept together) whose location in the source file can vary from its location in the printed document. (2) (verb) Of a keep, to be formatted in a location different from its location in the source file.

**flush.** Having no indentation.

**fold.** (1) To translate the lowercase characters of a character string into uppercase. (2) To place that portion of a line that does not fit within a column on the next output line.

**folio.** Page number.

**font.** (1) An assortment of type, all of one size and style. (2) A font library member that contains characters that must be used in conjunction with a code page font library member.

**font object.** Refers to a member of a font library. In CMS, a font object is a file whose filetype matches the name of the font library. In MVS, a font object can be a member of a partitioned data set (PDS).

**font set.** The set of fonts to be used in formatting a source document.

**footing.** Words located at the bottom of the text area. See also *running footing*.

**footnote.** A note of reference, explanation, or comment, placed below the text of a column or page, but within the body of the page (above the running footing).

**foreground.** The environment in which interactive programs are executed. Interactive processors reside in the foreground.

**format.** (1) (noun) The shape, size, and general makeup of a printed document. (2) (verb) To prepare a document for printing in a specified format.

**formatter.** (1) A computer program that prepares a source document to be printed. (2) That part of SCRIPT/VS that formats input lines for a particular logical device type.

**formatting mode.** In document formatting, the state in which input lines are concatenated and the resulting output lines are justified.

**form definition.** A resource object that defines the characteristics of the form which include: overlays to be used (if any), text suppression, the position of page data on the form, and the number and modifications of the page. Synonymous with FORMDEF.

**FORMDEF.** Synonym for form definition.

**front matter.** In a book, those sections (such as preface, abstract, table of contents, list of illustrations) that are placed before the main chapters or sections.

## G

**general document.** A type of document whose description can apply to a variety of documents, from memoranda to technical manuals. It can be used as a catch-all category for documents that do not conform to any other type description.

**Generalized Markup Language (GML).** A language that can be used to identify the parts of a source document without respect to particular processing.

**GML delimiter.** A special character that denotes the start of GML markup. In the starter set, it is initially a colon (:).

**GML end tag delimiter.** A special character that denotes the end of GML markup. In the starter set, it is initially a period (.).

**GML interpretation.** Interpretation of GML markup consists of recognizing the start or end of an element (or an attribute label), associating it with an APF, and

executing the APF. In SCRIPT/VS, interpretation is performed jointly by SCRIPT/VS itself and by APFs.

graphic character modification (GCM) module. Modules that contain scan patterns of IBM-supplied character sets and/or user-defined character sets, without respect to particular processing.

Graphical Data Display Manager (GDDM). An IBM licensed program that creates page segments.

gutter. In multicolumn formatting, the space between columns.

## H

hanging indention. The indention of all lines of a block of text following the first line (which is not indented the same number of spaces). Specified with the .OF [Offset] or .UN [Undent] control word.

head-level. The typeface and character size associated with the words standing at the beginning of a chapter or chapter topic.

heading. Words located at the beginning of a chapter or section or at the top of a page. See also *head-level* and *running heading*.

heading segment. An element that begins with a heading, followed by basic document elements and lower-level heading segments.

hexadecimal. Pertaining to a number system based on 16, using the sixteen digits 0, 1, . . . 9, A, B, C, D, E, and F. For example, hexadecimal 1B equals decimal 27. See also *EBCDIC*.

highlighting. Emphasis associated with a document element. In formatting, highlighting is usually expressed by changing font, overstriking, underscoring, and/or capitalizing the highlighted element.

horizontal justification. The process of redistributing the extra horizontal white space at the end of a line of text in between the words and letters of the line so as to exactly fill the width of the column with the text.

## I

IEBIMAGE. Utility program that creates and maintains various 3800 Printing Subsystem Model 1 modules (for example, character arrangement table and graphic character modification (GCM) modules) and stores them in SYS1.IMAGELIB.

image. A likeness or imitation of an object, such as a picture or logo.

impact printer. A printer, such as the 1403 and the 3211, in which printing is the result of mechanical impacts.

implied paragraph structure. An element that begins with an implied paragraph; that is, one for which you do not specifically enter a paragraph tag. The existence of the paragraph is understood from the existence of the implied paragraph structure, for example, as with notes, figure captions, and footnotes.

indent. To set typographical material to the right of the left margin.

indentation. The action of indenting. The condition of being indented. The blank space produced by indenting. Specified with the .IN [Indent], .IR [Indent Right], .UN [Undent], .OF [Offset], and .IL [Indent Line] control words. See also *hanging indention*.

initial caps. Capital letters occurring as the first letter of each word in a phrase. To set a phrase in initial caps is to capitalize the first letter of each word in the phrase.

initial value. A value assumed by SCRIPT/VS for a formatting function until the value is explicitly changed with a control word. The *initial value* is assumed even before the control word is encountered, whereas the *default value* is assumed when the control word is issued without parameters. See also *default value*.

initialize. This is a general programming term which means to set everything up correctly at the the beginning before you actually do any processing. For the starter set it means doing things such as mapping tags to APFs and setting up symbol names and values.

inline space. An amount of horizontal white space in a line that usually occurs between two words.

input device. A machine used to enter information into a computer system (for example, a terminal used to create a document).

input line. A line, as entered into a source file, to be processed by a formatter.

interactive. Pertaining to an application in which entries call forth a response from a system or program, as in an inquiry system. An interactive system might also be conversational, implying a continuous dialog between the user and the system. Interactive systems are usually communicated with via terminals, and respond immediately to commands. See also *foreground*.

interactive environment. The environment in which an interactive processor operates.



## Glossary

**intercharacter space.** Extra horizontal white space inserted *between* characters of a word. This space is in addition to the space included as part of the characters by the designer of the font.

**interword space.** See *word space*.

**Interactive System Productivity Facility (ISPF).** A dialog manager for interactive applications that provides control and services to allow processing of the dialogs in different host environments.

**italic.** A typestyle with characters that slant upward to the right.

## J

**job control language (JCL).** A language of job control statements used to identify a computer job or describe its requirements to the operating system.

**job control statement (JCS).** A statement that provides an operating system with information about the job being run.

**justification.** See *horizontal justification* and *vertical justification*.

**justify.** (1) (ISO) To control the printing positions of characters on a page so that the left-hand and right-hand margins of the printing are regular. (2) See *left-justify* and *right-justify*.

## K

**Kanji.** The non-phonetic Japanese writing system. In a font representing Kanji characters, each character is represented by a double-byte code. Contrast with Katakana.

**Katakana.** A character set consisting of symbols used in one of the two common Japanese phonetic alphabets. Each character is represented by one byte. Contrast with Kanji.

**keep.** (noun) In a source document, a collection of lines of text to be printed in the same column. When the vertical space remaining in the current column is insufficient for the block of text, the text is printed in the next column. (In the case of single-column format, the next column is on the next page.)

## L

**layout.** The arrangement of matter to be printed. See also *format*.

**leader.** (1) Dots or hyphens (as in a table of contents) used to lead the eye horizontally. (2) The divider between text and footnotes on a page (usually a short line of dashes that can be redefined).

**left-hand page.** The page on the left when a book is opened; usually even-numbered.

**left-justify.** (ISO) To control the printing positions of characters on a page so that the left-hand margins of the printing are regular.

**ligature.** A single character (piece of type or font raster) that represents two or more input characters: ff and ffi are examples of characters that may be represented by (printed as) a ligature.

**line device.** Any of a class of printers that accept one line of text from the host system at a time. SCRIPT/VS supports such line devices as the 1403 and 3800 Model 1.

**line space.** The vertical distance between the baseline of the current line and the baseline of the previous line.

**line spacing.** See *line space*.

**logical output device.** The combination of a physical output device and such logical variables as page size and number of lines per vertical inch (for line devices). A specification of 1403W6 is an example of a logical output device.

**logical page.** Synonym for page.

**lowercase.** Pertaining to small letters as distinguished from capitals; for example, a, b, g rather than A, B, G.

## M

**machine-readable.** Data in a form such that a machine can acquire or interpret (read) it from a storage device, from a data medium, or from another source.

**maclib.** See *macro library*

**macro.** See *macro instruction*

**macro instruction.** An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language. In SCRIPT/VS, a macro is a sequence of one or more

control words, symbols, and input lines. A macro's definition can be recursive.

macro library. A collection of macros. The form the library takes will vary by environment, being a MACLIB in CMS, a PDS in TSO, and so on.

macro substitution. During formatting, the substitution of control words, symbols, and text for a macro.

map. Associate a tag with an APF using the .AA [Associate APF] control word.

margin. (1) The space above, below, and on either side of the body of a page. (2) The left or right limit of a column.

mark up (verb). (1) To determine the markup for a document. (2) To insert markup into a source document.

markup. (noun) Information added to a document that enables a person or system to process it. Markup can describe the document's characteristics, or it can specify the actual processing to be performed. In SCRIPT/VS, markup consists of GML tags, attribute labels and values, and control words.

markup-content separator. A delimiter used in GML markup which indicates the end of the markup and the beginning of the text. The default markup content separator for GML is a period (.).

maximum ascender. The maximum height from the baseline to the top of the character in the font character set.

maximum descender. The maximum depth from the baseline to the bottom of the character in the font character set.

meter (m). Basic unit of linear measurement.

MCS. Markup/content separator.

One thousandth of a meter. There are 10 millimeters in one centimeter. (25.4 millimeters = 1 inch.)

## N

nonimpact printer. A printer, such as the IBM 3800 Printing Subsystem, in which printing is not the result of mechanical impacts, but is instead produced by another process such as laser beam, ink-jet, or electro-erosion. The IBM 3800 Printing Subsystem, for example, uses a laser based technology and the 4250 Printer uses an electro-erosion process.

## O

object. A sequential collection of control records that represent documents, pages, fonts, and so on.

offset. (1) (verb) To indent all lines of a block of text, except the first line. (2) (noun) The indentation of all lines of a block of text following the first line.

option. Information entered with a SCRIPT command to control the execution of SCRIPT/VS.

orientation. The angle between the top or bottom edge of the page and the baselines within a column, measured in a clockwise direction.

output device. A machine used to print, display, or store the result of processing.

output document. A machine-readable collection of lines of text or images that have been formatted, or otherwise processed, by a document processor. The output document can be printed or it can be filed for future processing.

output line. A line of text produced by a formatter.

## P

page. A collection of data that can be printed on a physical sheet of paper. Synonymous with logical page.

PAGEDEF. Synonym for page definition.

page definition. An object containing a set of formatting controls for printing pages of data. Includes controls for number of lines per printer form, font selection, print direction, and for mapping individual fields in the data to positions on the forms. Synonymous with PAGEDEF.

page device. A device that prints a formatted page that has graphics and text merged.

page printer. Any of a class of printers that accept composed pages, constructed of composed text and images, among other things. SCRIPT/VS supports such page printers as the 4250 Printer, the 3800 Model 3, and the 3820 Page Printer.

page segment. See *segment*.

paginate. To number pages.

paragraph unit. An element that has the same structure as a paragraph. In a General Document, the paragraph units are: paragraph, note, and paragraph continuation.

## Glossary

**parameter.** Any one of a set of properties whose values determine the characteristics or behavior of something. The syntax of some SCRIPT/VS control words includes parameters, which establish the properties of a formatting function or a printed page.

**part.** In a general document, a part is a zero-level heading segment. See also *heading segment*.

**patch PSC element.** A PSC element that is used temporarily to modify the normal output.

**pel.** The unit of horizontal measurement for the IBM 3800 Printing Subsystem and the 4250 Printer. On the IBM 3800 Printing Subsystem Model 1, one pel equals approximately 1/180th inch. On the 3800 Model 3 and the 3820 Page Printer, one pel equals approximately 1/240th inch. On the 4250 Printer, one pel equals approximately 1/600th inch.

**physical output device.** A physical device, such as a terminal, a disk file, a line printer, or a nonimpact printer. The 1403 Printer is an example of a physical output device.

**pica.** A unit of about 1/6 inch used in measuring typographical material. Similar to a cicero in the Didot point system.

**pitch.** A number that represents the amount of horizontal space a font's character occupies on a line. For example, 10-pitch means 10 characters per inch, or each character is 0.1 (1/10) inch wide. 12-pitch means 12 characters per inch, and 15-pitch means 15 characters per inch.

**point.** (1) A unit of about 1/72 of an inch used in measuring typographical material. There are twelve points to the pica. (2) In the Didot point system, a unit of 0.0148 inch. There are twelve Didot points to the cicero.

**PostScript language.** A programming language designed to convey a description of virtually any desired page to a printer capable of interpreting the page description.

**PostScript devices.** Any of a class of devices, such as the IBM 4216 Personal Pageprinter, that are configured to accept 8-bit ASCII and include PostScript files in DCF documents.

**PostScript image file.** Any file containing encapsulated PostScript that is imbedded in a DCF document by means of the .PO [PostScript] control word. PostScript image files can include any combination of images or text.

**profile.** (1) In SCRIPT/VS processing, a file that is imbedded before the primary file is processed. It can

be used to control the formatting of a class of source documents. When processing GML markup, the profile usually contains the mapping from GML to APFs and the symbol settings that define the formatting style. (2) In the DLF library, a collection of information that identifies a batch SCRIPT/VS user (user profile) or a document processor (attribute profile) or that defines certain library parameters (system profile).

**proportional spacing.** The spacing of characters in a printed line so that each character is allotted a space proportional to the character's width.

## R

**ragged right.** The unjustified right edge of text lines. See also *left-justify*.

**ragged left.** The unjustified right edge of text lines. See also *right-justify*.

**reference element.** In a general document, an element whose content is a reference to another element that is generated by an APF. There are five: figure reference, footnote reference, heading reference, index entry reference, and list item reference.

**required blank.** A character that prints as a blank, but does not act as a word separator.

**residual text.** The line of text following the markup/content separator of a GML tag.

**right-hand page.** The page on the right when a book is opened; usually odd-numbered.

**right-justify.** (ISO) To control the printing positions of characters on a page so that the right-hand margins of the printing are regular.

**row.** A horizontal arrangement of characters or other expressions on a printed page.

**rule.** (1) A straight horizontal or vertical line used, for example, to separate or border the parts of a figure or box. (2) A solid black rectangle of a given width, extending horizontally across the column or vertically down the column.

**running footing.** A footing that is repeated above the bottom margin area on consecutive pages (or consecutive odd- or even-numbered pages) in the body of the page (text area).

**running heading.** A heading that is repeated below the top margin area on consecutive pages (or consecutive odd- or even-numbered pages) in the body of the page (text area).

## S

**SCRIPT/VS.** The formatter component of the Document Composition Facility. SCRIPT/VS provides capabilities for text formatting and document management, macro processing and symbol substitution, and GML tag recognition and processing.

**section.** When an output page has two or more single-column parts with the same or different column-widths, or a single-column part and a multicolumn part, or two or more different multicolumn parts, each part of the output page is called a section.

**segment.** An object containing composed text and images, prepared before formatting and included in a document when it is printed.

**set.** This term is used in reference to a symbol. It implies the .SE [Set Symbol] control word.

**set size.** The set size of a given typeface determines the number of characters that will fit in a line of a given width when it is printed or set.

**small caps.** Capital letters in the same style as the normal capital letters in a font, but approximately the size of the lowercase letters.

**source document.** A machine-readable collection of lines of text or images that is used for input to a computer program.

**space.** A blank area separating words or lines.

**space unit.** A unit of measure of horizontal or vertical space. In GML markup, the em is used when a measure that is relative to the current font size is required. When an absolute measure is required, as in specifying the depth of a figure, recommended space units are inches (nnI), millimeters (nnW), picas/points (nnPnn), or Ciceros/Didot points (nnCnn), where nn is the number of units. See also *em*, *pica*, *point*, *Cicero*, and *Didot point system*.

**starter set.** An example of GML support that is provided with the Document Composition Facility. It consists of a document type description for general documents, a profile, and a library of APFs.

**SYSVAR.** An option of the SCRIPT command that permits the user to specify values for symbols. In the starter set, SYSVAR symbol values determine whether certain processing variations will occur, such as heading numbering, duplex formatting, and two-column printing.

**structure.** A characteristic of a document (or element) that expresses the type and relationship of the elements of the content. See also *element*.

**structured field.** A self-identifying string of bytes, analogous to a logical record. A structured field consists of an introducer, which identifies and characterizes the structured field, and data or parameters.

**symbol.** A name in a source document that can be replaced with something else. In SCRIPT/VS, a symbol is replaced with a character string. SCRIPT/VS can interpret the character string as a number, a character string, a control word, or another symbol.

**symbol substitution.** During formatting, the replacement of a symbol with a character string that SCRIPT/VS can interpret as a value (numeric, character string, or control word) or as another symbol.

**string.** A linear sequence of entities such as characters or physical elements.

## T

**tab.** (1) (noun) A preset point in the typing line of a typewriter-like terminal. A preset point in an output line. (2) (verb) To advance to a tab for printing or typing. (3) (noun) a tab character, X'05'.

**table (ISO).** (1) An array of data each item of which may be unambiguously identified by means of one or more arguments. (2) An arrangement of cells in rows and columns.

**tag.** In GML markup, a name for a type of document (or document element) that is entered in the source document to identify it. For example, .p. might be the tag used to identify each paragraph.

**terminal.** A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel.

**text item.** Explicitly marked (tagged) elements that occur within text, such as within a paragraph unit. In a general document, for example, quotations and phrases are text items.

**text programmer.** One who implements APFs that provide the processing specified by the document administrator. In SCRIPT/VS, this involves writing SCRIPT/VS macros and organizing macro libraries and profile files so that the appropriate composition will be done for each tag.

**text line.** An input line that contains only text.

**text variable.** A symbol whose final value is to be treated as text only.

## Glossary

**token.** A string of characters that is treated as a single entity. In SCRIPT/VS, a parameter passed to a macro in one of the local variables &\*1, ... &\*n.

**top margin.** On a page, the space between the body or running heading and the top edge of the page.

**translate table.** That 256-byte portion of the character arrangement table that translates the user's data code for a character recognizable by the 3800 Printing Subsystem Model 1.

**TRC.** Table reference character. In printer SYSOUT data sets, a second control byte, following the carriage control byte, which indicates which font the record is to be printed in. The presence of TRCs is indicated by the JCL parameter DCB=OPTCD=J.

**TSO.** An interactive processor within OS/VS2.

**type posture.** A typeface style variation indicating whether a typeface is upright (as in roman) or slanted to the right (as in italic or cursive).

**type size.** The vertical height (point size) of a given typeface, such as 10 point.

**type style.** Style variations in a typeface. Among these variations are posture, weight, and width.

**type weight.** The relative thickness of the strokes of a typeface. Usually described in such terms as light, demi bold, bold, and so on.

**type width.** The horizontal size (set size) of a given typeface. The width may be given in units of measurement, such as set 9 point, or it may be descriptive: ultra condensed, condensed, expanded, and so on.

**typeface.** All type of a single style. There might be several fonts (different sizes) with the same typeface or style.

**typeface family.** A collection of fonts of a common typeface that vary in size and style.

**typeset.** (1) (verb) To arrange the type on a page for printing. (2) (adjective) Pertaining to material that has been set in type.

## U

**underscore.** (1) (noun) A line printed under a character. (2) (verb) To place a line under a character.

**unformatted mode.** (1) In document formatting, the state in which each input line is processed and

printed without formatting. Other SCRIPT/VS control words remain in effect and are recognized. (2) In document printing using the UNFORMAT option, the state in which each input line (control words as well as text) is printed as it exists in the input, in the order in which it is processed. No formatting is done.

**unique identifier (ID).** In a general document, an attribute whose value serves as a name which can be used to refer to the element. See also *reference element*.

**unit space.** The minimum amount of additional spacing acceptable for purposes of horizontal justification, as specified by the font designer.

**uppercase.** Pertaining to capital letters, as distinguished from small letters; for example, A, B, G rather than a, b, g.

## V

**variable.** A quantity that can assume any of a given set of values.

**variable text.** For the .VT [Variable Text] control word, text to be inserted in a formatted document by a post processor.

**vertical justification.** The process of redistributing the extra vertical white space at the end of a column in between the lines of text, so as to make each column in a set appear to be equal in depth.

## W

**widow.** One or two lines or words at the end of a paragraph that are printed separately from the rest of the paragraph.

**word space.** The horizontal white space placed between words; referred to as an interword blank.

**word spacing.** The space between words in a line. See also *word space*.

**Writable Character Generation Module (WCGM).** A 64-position portion of the 3800 Printing Subsystem Model 1's character generation storage that holds the scan elements of one character set. There are two WCGMs in the basic 3800, and optional added storage provides two more.

# Index

## A

ABSTRACT tag 86  
 address line (ALINE) tag 86  
 ADDRESS tag 86  
 addressability, all-points  
   definition 175  
 administrator, document  
   definition 177  
 advanced function printing  
   definition 175  
 AFP  
   definition 175  
 ALIGN attribute  
   on RDEF tag 52  
 alignment  
   definition 175  
 ALINE (address line) tag 86  
 all-points addressability  
   definition 175  
 alphabetic tag summary 148  
 alphanumeric string  
   definition 175  
 ampersand  
   definition 175  
 ampersand (&) 113  
 ampersand (&) symbol 113  
 APF  
   definition 175  
 API  
   definition 175  
 appendix section 88  
   description of 83  
 APPENDIX tag 88  
 Application Programming Interface  
   See API  
 arabic numerals  
   body of document 87  
 ARRANGE attribute  
   on RDEF tag 54  
 ascender  
   definition 175  
 ascender, maximum  
   definition 181  
 ATMS-II  
   acronym for 175  
 ATMS-III 7  
 attributes  
   BREAK (on DL tag) 22  
   COMPACT  
     on DL tag 22  
     on list tags 17  
   definition 175  
   DEPTH  
     on FIG tag 34  
     on XMP tag 31

attributes (*continued*)  
   FRAME (on FIG tag) 34  
   HEADHI (on DL tag)  
     type of highlighting 45  
   how to specify  
     labeled 23  
     value (single-word) 20  
 ID  
   on FIG tag 75  
   on FN tag 77  
   on H0-H6 tag 73  
   on indexing tags 102  
   on LI tag 19  
   on LIREF tag 74  
   on TABLE tag 76  
   rules for 73  
 labeled, how to specify 23  
 label, definition 175  
 PAGE  
   on FIGREF tag 75  
   on HDREF tag 73  
   on TREF tag 76  
 PG (on indexing tags) 102  
 PLACE (on FIG tag) 34  
 PRINT (on indexing tags) 102  
 PROC (on PSC tag) 107  
 REFID  
   on FIGREF tag 75  
   on FNREF tag 77  
   on HDREF tag 73  
   on indexing tags 102  
   on LIREF tag 74  
   on TREF tag 76  
 SEC (on GDOC tag) 83  
 SEE (on indexing tags) 102  
 SEEID (on indexing tags) 102  
 STITLE (short title)  
   on TITLE tag 85  
 TERMHI (on DL tag)  
   type of highlighting 45  
 TSIZE (on DL tag) 22  
 value, how to specify 20  
 WIDTH (on FIG tag) 34  
   with an equal sign 23  
 AUTHOR tag 85

## B

back matter 88  
   BACKM tag 88  
   definition 175  
   description of 83  
 balancing  
   definition 175

## Index

- baseline
  - definition 175
- basic document element
  - definition 175
- batch environment
  - definition 175
- BIND (SCRIPT command option) 125
- binding edge
  - definition 175
- blank pages 120
- blanks, disappearing 139
- blanks, sorting (index) 101
- blank, required
  - definition 182
- blank, required (symbol) 113
- body 87
  - definition 175
  - description of 83
- BODY tag 87
- boldface
  - definition 175
- bottom margin
  - definition 175
- break
  - definition 176
- bug
  - definition 176
- C**
- C [Cell] tag 57
- C tag
  - cell number attribute 57
- call
  - definition 176
- caps
  - definition 176
- caps, initial
  - definition 179
- caption
  - definition 176
- case-sensitive
  - definition 176
- CDPF
  - acronym 176
- cell
  - definition 176
- cell number
  - of a table, current 57
- cell number attribute
  - on C tag 57
- cell, in rows of a table
  - cell [C] tag 57
  - definition of 51
  - how to specify 57
- centimeter (cm)
  - definition 176
- centimeters 39
- changes with Release 3.2 xvii
- chapter
  - definition 176
- character
  - definition 176
- character arrangement table
  - definition 176
- character set
  - definition 176
- character space
  - definition 176
- character spacing
  - definition 176
- characters, horizontal space units 39
- characters, non-keyable 113
- character, fill
  - definition 178
- CHARS (SCRIPT command option) 125, 134
- checking spelling 129
- cicero
  - definition 176
- ciceros 39
- CIT (title citation) tag 46
- CM (comment) control word 109
- CMS 7
  - acronym 176
- code page
  - definition 176
- code point
  - definition 176
- coded font
  - definition 176
- colon missing after level 5 or 6 heading 137
- colon (:)
  - illustrating a GML tag 113
  - &gml. symbol 113
- COLUMN attribute
  - on TABLE tag 55
- column balancing
  - definition 176
- column is narrow 137
- column width
  - definition 176
- columns
  - definition 176
- combining text and graphics 38
- command
  - definition 176
- comment
  - definition 176
- comments in the source document 91
  - with PSC tag 109
- COMPACT attribute (on list tags) 20
- compartment
  - in a table
  - See cell
- composed text
  - definition 176

- composite rotation
    - definition 176
  - composition
    - definition 176
  - compositor
    - definition 177
  - computing system 7
  - CONCAT attribute
    - on RDEF tag 53
  - concatenation
    - definition 177
  - CONTINUE (SCRIPT command option) 132
  - control word line
    - definition 177
  - control word separator
    - illustrating (&semi. symbol) 113
    - semicolon misinterpreted as 139
  - control word statement
    - definition 177
  - control words
    - and PSC tag 107
    - CM (comment) 109
    - definition 177
    - IM (imbed) 89
    - SE (set symbol) 116
  - conversion processor, document
    - definition 177
  - copy group
    - definition 177
  - courses
    - self-study v
    - training v
  - cross-reference listing 79
    - example (following index)
    - explanation of 141
    - X SYSVAR 122
  - cross-references 73
    - direction 74
    - figures 75
      - FIGCAP required for 75
    - footnotes 78
    - headings 73
    - listing 141
    - tables 76
      - TCAP required for 76
    - to a table 52
    - unmatched REFIDs 78
  - current left margin
    - definition 177
  - current line
    - definition 177
  - CWIDTHS attribute
    - on RDEF tag 54
- D**
- D (duplex) SYSVAR 120
  - data sets, multiple input 89
  - data set, identifying 124
  - DATE tag 85
  - date, symbol for 113
  - DCF publications v
    - library guide iv
  - DD (definition description) tag 22
  - DDHD (definition description heading) tag 22
  - debug
    - definition 177
  - default fonts 133, 134
  - default value
    - definition 177
  - defining rows, of a table
    - See RDEF tag
  - defining your own symbols 116
  - definition description (DD) tag 22
  - definition list 21
  - definition list (DL) tag 22
    - BREAK attribute 22
    - COMPACT attribute 22
    - HEADHI attribute 22
      - type of highlighting 45
    - TERMHI attribute 22
      - type of highlighting 45
    - TSIZE attribute 22
  - definition term (DT) tag 22
    - rule for entering 23
  - DEPTH attribute
    - on FIG tag 34
    - on XMP tag 31
  - descender
    - definition 177
  - descender, maximum
    - definition 181
  - destination
    - definition 177
  - device specification 130
  - device units, horizontal and vertical space units 39
  - DEVICE (SCRIPT command option) 130
  - device-dependent markup (PSC tag) 107
  - device, input
    - definition 179
  - device, line
    - definition 180
  - device, output
    - definition 181
  - device, output, physical
    - definition 182
  - dictionary
    - definition 177
  - Didot point system
    - definition 177
  - disappearing blanks 139
  - disappearing semicolon 139
  - disappearing text 136
  - DL (definition list) tag 22
    - BREAK attribute 22
    - COMPACT attribute 22



- DL (definition list) tag (*continued*)
  - HEADHI attribute 22
  - type of highlighting 45
  - TERMHI attribute 22
  - type of highlighting 45
  - TSIZE attribute 22
- DOCNUM (document number) tag 85
- document
  - definition 177
- document administrator
  - definition 177
- document conversion processor
  - definition 177
- document element group, summary of tags 151
- document element, basic
  - definition 175
- document library
  - definition 177
- document number (DOCNUM) tag 85
- document structure 83–91
  - summary 89
- document title 85
- document types 5
  - general documents
    - definition 178
- documents, general 5
- documents, multiple input 89
- document, identifying 124
- document, output
  - definition 181
- document, source
  - definition 183
- dot leader
  - definition 177
- double column 120
- DSMPROF3 6
  - how to specify 127
  - symbols provided 113
- DT (definition term) tag 22
  - rule for entering 23
- DTHD (definition term heading) tag 22
- duplex
  - definition 177
- duplex (D) SYSVAR 120

**E**

- EBCDIC
  - acronym 177
- edge, binding
  - definition 175
- EDIT
  - definition 177
- editor
  - definition 177
- editor (computer program)
  - text 6
- eject
  - definition 177

- element
  - definition 177
- element, basic document
  - definition 175
- element, reference
  - definition 182
- em
  - definition 177
- ems, horizontal and vertical space units 39
- en
  - definition 177
- enable
  - definition 178
- end tags 19
- entries, index 93
  - placement 101
- environment
  - batch 175
  - definition 175
  - interactive
    - definition 179
- epifile
  - definition 178
- error messages
  - controlling printing 133
- errors, markup
  - symptoms and solutions 136
- escapement
  - definition 178
- example (XMP) tag 31
  - DEPTH attribute 31
- examples 31
- excerpts (long quotations) 48
- exercises
  - cross-references and footnotes 80
  - examples and figures 43
  - highlighting and quoting 50
  - indexing 105
  - lists 29
  - overall document structure 91
  - solutions 169
  - tables 70
- extended symbol processing
  - definition 178

**F**

- factor
  - definition 178
- field, structured
  - definition 183
- FIG (figure) tag 34
  - DEPTH attribute 34
  - FRAME attribute 34
  - ID attribute 75
  - PLACE attribute 34
  - WIDTH attribute 34
- FIGCAP (figure caption) tag 35
  - required for cross-references 75

- FIGDESC (figure description) tag 36
  - FIGLIST (list of figures) tag 87
  - FIGREF (figure reference) tag 75
    - PAGE attribute 75
    - REFID attribute 75
  - figure caption (FIGCAP) tag 35
    - required for cross-references 75
  - figure description (FIGDESC) tag 36
  - figure list 87
  - figure reference (FIGREF) tag 75
    - PAGE attribute 75
    - REFID attribute 75
  - figure space
    - definition 178
  - figure (FIG) tag 34
    - DEPTH attribute 34
    - FRAME attribute 34
    - ID attribute 75
    - PLACE attribute 34
    - WIDTH attribute 34
  - figures
    - captions 35
    - description 36
    - floating 32, 37
    - how to enter 32
  - FILE (SCRIPT command option) 132
  - files, multiple input 89
  - file, input, identifying 124
  - file, writing output to 132
  - fill character
    - definition 178
  - float
    - definition 178
  - floating figures 32, 37
  - flush
    - definition 178
  - FN (footnote tag) 77
    - ID attribute 77
  - FNREF (footnote reference) tag 77
    - REFID attribute 77
  - fold
    - definition 178
  - folio
    - See *also* page numbers
    - definition 178
  - font
    - definition 178
  - font object
    - definition 178
  - font set
    - definition 178
  - font specification (on 3800) 125
  - fonts, default 133
  - fonts, page printers 133
  - fonts, PostScript devices 133
  - footing
    - definition 178
  - footing, running
    - definition 182
  - footnote reference (FNREF) tag 77
    - REFID attribute 77
  - footnote (FN) tag 77
    - ID attribute 77
  - footnotes 78
    - definition 178
    - placement 77
  - foreground
    - definition 178
  - form definition
    - definition 178
  - format
    - definition 178
  - formatter
    - definition 178
  - formatting doesn't work 137
  - formatting mode
    - definition 178
  - FORMDEF
    - definition 178
  - FRAME attribute
    - on FIG tag 34
  - front matter
    - contents
      - abstract 86
      - address of author or publisher 86
      - author 85
      - date 85
      - document number 85
      - document title 85
      - list of figures 87
      - preface 86
      - table of contents 87
      - title page 84
    - definition 178
    - description of 83
    - FRONTM tag 84
  - front matter (FRONTM) tag 84
  - future applications of GML 47
- G**
- GCM module
    - definition 179
  - GD (glossary definition) tag 148
  - GDOC (general document) tag 83
    - SEC attribute 83
  - general document
    - definition 178
  - general document (GDOC) tag 83
  - general documents 5
  - Generalized Markup Language (GML) 5
    - future applications 47
  - generic devices
    - page printers xvii
    - specifying xvii

## Index

generic page printer devices xvii  
GL (glossary list) tag 148  
GML  
    definition 178  
GML delimiter  
    definition 178  
GML delimiter symbol 113  
GML end tag delimiter  
    definition 178  
GML interpretation  
    definition 178  
GML tags  
    See tags  
Graphic Data Display Manager  
    definition 179  
graphic (with PSC tag) 109  
GT (glossary term) tag 148  
gutter  
    definition 179

## H

H (head level numbering) SYSVAR 122  
hanging indention  
    definition 179  
HDREF (head reference) tag 73  
    PAGE attribute 73  
    REFID attribute 73  
head level numbering (H) SYSVAR 122  
head levels 0-1 (H0-H1) tags  
    ID attribute 73  
    right-hand page 120  
    running foot (odd) 120  
head levels 2-6 (H2-H6) tags  
    ID attribute 73  
head reference (HDREF) tag 73  
    PAGE attribute 73  
    REFID attribute 73  
head-level  
    definition 179  
HEADHI attribute  
    type of highlighting 45  
headings  
    definition 179  
    for definition list 22  
    numbering 122  
    segment  
        definition 179  
heading, running  
    definition 182  
hexadecimal  
    definition 179  
highlighted phrase (HP) attribute  
    on RDEF tag 52  
highlighted phrases 45  
    nesting 45  
    on the terminal 46  
    on the 1403 printer 46  
    on the 3800 printer 45

highlighted phrases (*continued*)  
    on the 3820 Page Printer 45  
    on the 4250 printer 45  
    tags 45  
highlighting  
    definition 179  
horizontal justification  
    definition 179  
horizontal space units 39  
HP0-HP3 (highlighted phrase 0-3) attribute  
    on RDEF tag 52  
HP0-HP3 (highlighted phrase 0-3) tags 45  
    nesting 45  
H0-H1 (head levels 0-1) tags  
    ID attribute 73  
    right-hand page 120  
    running foot (odd) 120  
H2-H6 (head levels 2-6) tags  
    ID attribute 73

## I

ID attribute  
    on FIG tag 75  
    on FN tag 77  
    on H0-H6 tag 73  
    on indexing tags 102  
    on LIREF tag 74  
    on RDEF (row definition) tag 52  
    on TABLE tag 55, 76  
    rules for 73  
identifier (ID attribute), rules for 73  
IEBIMAGE  
    definition 179  
IH1-IH3 (index entry heading) tags 102  
IM (imbed) control word 89  
image  
    definition 179  
    included with text 38  
imbedding files 89  
impact printer  
    definition 179  
implied paragraph 28  
implied paragraph structure  
    definition 179  
inches 39  
including page segments 38  
indent  
    definition 179  
indentation  
    definition 179  
indentation, hanging  
    definition 179  
index entry heading (IH1-IH3) tags 102  
index entry reference (IREF) tag 102  
index entry term (I1-I3) tags 102  
INDEX tag 102  
INDEX (SCRIPT command option) 126

indexing 93–104  
 blanks, sorting 101  
 creating entries 93  
 doesn't print 139  
 placement in back matter 102  
 placement of entries 101  
 readable source files 104  
 see and see also references 98  
 sorting 99  
 summary of tags 102

initial caps  
 definition 179

initial value  
 definition 179

initialize  
 definition 179

inline space  
 definition 179

input device  
 definition 179

input line  
 definition 179

input record length  
 extending with symbols 117

interactive  
 definition 179

interactive environment  
 definition 179

intercharacter space  
 definition 180

interword space  
 definition 180

IREF (index entry reference) tag 102

ISPF  
 definition 180

italic  
 definition 180

I1-I3 (index entry term) tags 102

**J**

JCL  
 definition 180

JCS  
 definition 180

justification  
 definition 180  
 horizontal 179  
 definition 179  
 vertical  
 definition 184

justify  
 definition 180

**K**

Kanji  
 definition 180

Katakana  
 definition 180

Keep  
 definition 180

**L**

labeled attributes 23

label, attribute  
 definition 175

layout  
 definition 180

leader  
 definition 180

leader, dot  
 definition 177

left-hand page  
 definition 180

left-justify  
 definition 180

LI (list item) tag 19

library guide  
 for DCF publications iv

library, document  
 definition 177

ligature  
 definition 180

line device  
 definition 180

line space  
 definition 180

line spacing  
 definition 180

lines, vertical space units 39

line, input  
 definition 179

LIREF (list item reference) tag 74  
 ID attribute 74  
 REFID attribute 74

list item reference (LIREF) tag 74  
 REFID attribute 74

list item (LI) tag 19  
 ID attribute 74

list of figures 87

list of figures (FIGLIST) tag 87

list of illustrations  
 empty 139  
 page numbers wrong 138

list of tables  
 empty 139  
 page numbers wrong 138

list of tables (TLIST) tag 87  
 how to produce one 87

list part (LP) tag 27

listing of cross-references 79

listing, cross-reference 141

lists 17  
 compact 17  
 definition 18, 21

## Index

lists (*continued*)  
  list part (LP) tag 27  
  nesting 25  
  simple 17  
  unordered 17  
logical device 130  
logical output device  
  definition 180  
logical page  
  definition 180  
long input records 117  
long quotations 48  
long quote (LQ) tag 48  
lowercase  
  definition 180  
LP (list part) tag 27  
LQ (long quote) tag 48

## M

machine-readable  
  definition 180  
macro instruction  
  definition 180  
macro library  
  definition 181  
macro substitution  
  definition 181  
map  
  definition 181  
margin  
  bottom  
    definition 175  
  definition 181  
  top  
    definition 184  
mark up  
  definition 181  
markup  
  definition 181  
markup errors  
  symptoms and solutions 136  
markup-content separator 19  
  definition 181  
  when required 19  
markup, general guidelines 145  
MCS  
  acronym 181  
MESSAGE (SCRIPT command option) 133  
messages, error  
  controlling printing 133  
meter (m)  
  definition 181  
millimeter (mm)  
  definition 181  
millimeters 39  
MINDEPTH attribute  
  on RDEF tag 53

mode, formatting  
  definition 178  
multiple input files 89

## N

nesting highlighted phrases 45  
nesting lists 25  
nesting quotations 48  
nonimpact printer  
  definition 181  
nonkeyable characters 113  
NOSEGLIB (SCRIPT command option) 126  
NOTE tag 47  
number  
  as an attribute on C tag 57  
  document (DOCNUM tag) 85  
  of current cell, in a table 57  
  page, arabic 87  
  page, front matter 84  
numbering headings 122

## O

object  
  definition 181  
offset  
  definition 181  
offset style 120  
OL (ordered list) tag 17  
  COMPACT attribute 17  
  numbering the items 18  
option  
  definition 181  
OPTIONS (SCRIPT command option) 126  
ordered list (OL) tag 17  
  COMPACT attribute 17  
  numbering the items 18  
orientation  
  definition 181  
output device  
  definition 181  
output device, physical  
  definition 182  
output document  
  definition 181  
output file, naming 132  
output line  
  definition 181  
overall document structure 83–91  
  summary 89  
override default fonts 134

## P

P (process value) SYSVAR 122  
page  
  definition 181

- PAGE attribute
    - on FIGREF tag 75
    - on HDREF tag 73
    - on TABLE tag 55
    - on TREF tag 76
  - page definition
    - definition 181
  - page device
    - definition 181
  - page numbering
    - body 87
    - front matter 84
  - PAGE option, how to specify 127
  - page printer
    - definition 181
  - page printer fonts 133
  - page segment
    - definition 181
  - page segments
    - including in your document 38
    - reserving space for in your document 38
  - PAGE (SCRIPT command option) 127
  - PAGEDEF
    - synonym for 181
  - page, right-hand
    - definition 182
  - paginate
    - definition 181
  - paragraph
    - continuation 32
    - implied 28
  - paragraph continuation (PC) tag 32
  - paragraph unit
    - definition 181
  - parameter
    - definition 182
  - part
    - definition 182
  - patch PSC element
    - definition 182
  - patch (with PSC tag) 109
  - PC (paragraph continuation) tag 32
  - pel
    - definition 182
  - period (.)
    - entering SCRIPT/VS control words 89
    - when required 19
  - PG attribute (on indexing tags) 102
  - phrases used, descriptions of 3
  - phrases, highlighted 45
    - nesting 45
    - on the terminal 46
    - on the 1403 printer 46
    - on the 3800 printer 45
    - on the 3820 Page Printer 45
    - on the 4250 printer 45
  - physical device 130
  - physical output device
    - definition 182
  - pica
    - definition 182
    - space unit 39
  - pictures included with text 38
  - pitch
    - definition 182
  - PLACE attribute (on FIG tag) 34
  - point
    - definition 182
  - point system, Didot
    - definition 177
  - PostScript 182
  - PostScript fonts 133
  - PostScript images 39
  - PostScript support xvii
  - PostScript, using PROC attribute 108
  - PREFACE tag 86
  - primary input file, identifying 124
  - PRINT attribute (on indexing tags) 102
  - PRINT (SCRIPT command option) 132
  - printer
    - definition 179
    - impact
    - nonimpact
      - definition 181
    - page
      - definition 181
  - PROC attribute (on PSC tag) 107
    - P (process value) SYSVAR 122
  - process value (P) SYSVAR 122
  - process-specific controls (PSC) tag 107
    - P (process value) SYSVAR 122
    - PROC attribute 107
  - profile
    - definition 182
  - PROFILE (SCRIPT command option) 127
  - proportional spacing
    - definition 182
  - PSC (process-specific controls) tag 107
    - P (process value) SYSVAR 122
    - PROC attribute 107
  - PSOUT (SCRIPT command option) 127
  - publications
    - for DCF v
- Q**
- Q (quote) tag 48
  - quotations 47
    - excerpts 48
    - inline 47
    - nesting 48
  - quote (Q) tag 48

**R**

R (read) SYSVAR 121

ragged right  
definition 182

RDEF (row definition) tag 51  
ALIGN attribute 52  
ARRANGE attribute 54  
CONCAT attribute 53  
CWIDTHS attribute 54  
HP0-HP3 (highlighted phrase 0-3) attribute 52  
ID attribute 52  
MINDEPTH attribute 53  
ROTATE attribute 53  
VALIGN attribute 53

read (R) SYSVAR 121

reference element  
definition 182

REFID attribute  
on FIGREF tag 75  
on FNREF tag 77  
on HDREF tag 73  
on indexing tags 102  
on LIREF tag 74  
on ROW tag 56  
on TABLE tag 55  
on TFT (table footer) tag 58  
on THD (table header) tag 57  
on TREF tag 76  
unmatched 78

related publications  
descriptions vi

Release 3.2 changes xvii

required blank  
definition 182

required blank symbol 113

residual text  
definition 182

right-hand page  
definition 182  
duplex (D) SYSVAR 120

right-justify  
definition 182

roman numerals  
front matter 84

ROTATE attribute  
on RDEF tag 53  
on TABLE tag 56

row  
definition 182

row definition (RDEF) tag 51  
ALIGN attribute 52  
ARRANGE attribute 54  
CONCAT attribute 53  
CWIDTHS attribute 54  
HP0-HP3 (highlighted phrase 0-3) attribute 52  
ID attribute 52  
MINDEPTH attribute 53  
ROTATE attribute 53

row definition (RDEF) tag (*continued*)  
VALIGN attribute 53

ROW [Row] tag 56  
starting a row 56

ROW tag  
REFID attribute 56  
SPLIT attribute 56

rows, in a table  
definition of 51

rule  
definition 182

rules  
in FRAME attribute, 33  
on FIG (figure) tag 33

run time  
things you can specify 119

running foot  
even 85  
STITLE attribute on TITLE 85

running footing  
definition 182

running heading  
definition 182

**S**

S (style) SYSVAR 120

SCRIPT command 123  
BIND option 125  
CHARS option 125, 134  
CONTINUE option 132  
DEVICE option 130  
FILE option 132  
FPASSES option 126  
how to enter 124  
INDEX option 126  
MESSAGE option 133  
NOSEGLIB option 126  
OPTIONS option 126  
PAGE option 127  
PRINT option 132  
PROFILE option 127  
PSOUT option 127  
SEGLIB option 127  
SPELLCHK option 129  
SYSVAR option 129  
TERM option 132  
TWO PASS option 129

SCRIPT/VS  
definition 183

SCRIPT/VS control words  
and PSC tag 107  
CM (comment) 109  
IM (imbed) 89  
SE (set symbol) 116

SE (set symbol) control word 116

SEC attribute (on GDOC tag) 83

section  
definition 183

- see and see also references (indexes) 98
- SEE attribute (on indexing tags) 102
- SEEID attribute (on indexing tags) 102
- SEGLIB (SCRIPT command option) 127
- segment
  - definition 183
- segment include 38
- segment include library 127
- segment, page
  - definition 181
- self-study courses v
- semicolon
  - symbol for 113
  - won't print 139
- semicolon symbol 113
- set
  - definition 183
- set size
  - definition 183
- simple list (SL) tag 17
  - COMPACT attribute 17
- single column 120
- single quotation mark (')
  - in a symbol's value 116
  - in an attribute value 145
- single-word attributes 20
- SL (simple list) tag 17
  - COMPACT attribute 17
- small caps
  - definition 183
- solutions for exercises 169
- sorting index entries 99, 101
- source document
  - definition 183
- source files, multiple 89
- space
  - definition 183
- space unit
  - definition 183
- space units 39
- space, figure
  - definition 178
- space, inline
  - definition 179
- space, intercharacter
  - definition 180
- space, interword
  - definition 180
- space, line
  - definition 180
- space, word
  - definition 184
- spacing, proportional
  - definition 182
- spacing, word
  - definition 184
- special characters, non-keyable 113
- SPELLCHK (SCRIPT command option) 129
- spelling verification 129
- SPLIT attribute
  - on ROW tag 56
  - on TABLE tag 55
- splitting a table 56
- starter set 5
  - DSMPROF3 6
  - symbols provided 113
- starter set unit
  - definition 183
- STITLE (short title) attribute
  - on TITLE 85
  - running foot (even) 85
- string
  - definition 183
- string (definition)
- string, alphameric
  - definition 175
- structure
  - definition 183
- structure of document 83–91
  - summary 89
- structured field
  - definition 183
- style (S) SYSVAR 120
- Summary of Amendments xvii
- summary of overall document structure 89
- summary of tags 147
  - alphabetic 148
  - by document element group 151
- support
  - generic page printer devices xvii
- symbol
  - definition 183
- symbol processing, extended
  - definition 178
- symbol substitution
  - definition 183
- symbols 113
  - ampersand (&) 113
  - date 113
  - defining your own 116
  - GML delimiter 113
  - how to enter 113
  - nonkeyable characters 113
  - provided by the starter set 113
  - required blank 113
  - semicolon 113
  - time 113
- symptoms and solutions
  - blank inserted at GML tag 139
  - blanks disappear 139
  - colon after a level 5 or 6 heading is missing 137
  - column is narrow 137
  - index is empty 139
  - page numbers are wrong 138
  - semicolon won't print 139



## Index

- symptoms and solutions (*continued*)
    - table of contents, list of illustrations or tables are empty 139
    - text disappears 136
    - text doesn't format 137
    - unexpected formatting results 136
  - system variables (SYSVARs) for the starter set 120
    - cross-reference listing (X) 122
    - duplex (D) 120
    - head level numbering (H) 122
    - how to specify 120
    - process value (P) 122
    - read (R) 121
    - style (S) 120
    - title page (T) 120
    - write (W) 121
  - system, computing 7
  - SYSVAR
    - definition 183
  - SYSVARs (system variables) for the starter set 120
    - cross-reference listing (X) 122
    - duplex (D) 120
    - head level numbering (H) 122
    - how to specify 120
    - process value (P) 122
    - read (R) 121
    - style (S) 120
    - title page (T) 120
    - write (W) 121
  - SYSVAR(SCRIP command option) 129
- ## T
- T (title page) SYSVAR 120
  - tab
    - definition 183
  - table
    - definition 183
  - table caption (TCAP) tag 58
    - required for cross-references 76
  - table description (TDESC) tag 58
  - table footer (TFT) tag 58
    - REFID attribute 58
  - table header (THD) tag 57
    - REFID attribute 57
  - table of contents 87
    - empty 139
    - page numbers wrong 138
  - table of contents (TOC) tag 87
  - table reference (TREF) tag 76
    - PAGE attribute 76
    - REFID attribute 76
  - TABLE tag 55
    - C (cell) attribute 55
    - COLUMN attribute 55
    - ID attribute 55, 76
    - PAGE attribute 55
    - REFID attribute 55
    - ROTATE attribute 56
  - TABLE tag (*continued*)
    - ROW attribute 55
    - SPLIT attribute 55
    - WIDTH attribute 55
  - tables
    - building a table 55
    - defining characteristics of 51
    - definition of 51
    - starting a table 51
  - table, translate
    - definition 184
  - tag
    - definition 183
  - tags
    - ABSTRACT 86
    - ADDRESS 86
    - ALINE (address line) 86
    - APPENDIX 88
    - AUTHOR 85
    - BACKM (back matter) 88
    - BODY 87
    - C [Cell], in rows of a table 57
    - CIT (title citation) 46
    - DATE 85
    - DD (definition description) 22
    - DDHD (definition description heading) 22
    - DL (definition list) 22
    - DOCNUM (document number) 85
    - DT (definition term) 22
    - DTHD (definition term heading) 22
    - FIG (figure) 34
    - FIGCAP (figure caption) 35
    - FIGDESC (figure description) 36
    - FIGLIST (list of figures) 87
    - FIGREF (figure reference) 75
    - FN (footnote) 77
    - FNREF (footnote reference) 77
    - FRONTM (front matter) 84
    - GD (glossary definition) 148
    - GDOC (general document) 83
    - GL (glossary list) 148
    - GT (glossary term) 148
    - HDREF (head reference) 73
    - HP0-HP3 (highlighted phrase 0-3) 45
    - IH1-IH3 (index entry heading) 102
    - illustrating (&gml symbol) 113
    - INDEX 102
    - IREF (index entry reference) 102
    - I1-I3 (index entry term) 102
    - LI (list item) 19
    - LIREF (list item reference) 74
    - LP (list part) 27
    - LQ (long quote) 48
    - NOTE 47
    - OL (ordered list) 17
    - PC (paragraph continuation) 32
    - PREFACE 86
    - PSC (process-specific controls) 107

- tags (*continued*)
    - Q (quote) 48
    - RDEF (row definition) 51
    - ROW 56
    - SL (simple list) 17
    - summary 147
      - alphabetic 148
      - by document element group 151
    - TABLE 55
    - TCAP (table caption) 58
    - TDESC (table description) 58
    - TFT (table footer) 58
    - THD (table header) 57
    - TITLE 85
    - TITLEP (title page) 84
    - TOC (table of contents) 87
    - TREF (table reference) 76
    - UL (unordered list) 17
    - XMP (example) 31
  - TCAP (table caption) tag 58
    - required for cross-references 76
  - TDESC (table description) tag 58
  - TERM (SCRIPT command option) 132
  - TERMHI attribute
    - type of highlighting 45
  - terminal
    - definition 183
  - terminal formatting
    - forcing uppercase 46
  - text doesn't format 137
  - text editor
  - text item
    - definition 183
  - text line
    - definition 183
  - text programmer
    - definition 183
  - text variable
    - definition 183
  - text, composed
    - definition 176
  - text, residual
    - definition 182
  - TFT (table footer) tag 58
    - REFID attribute 58
  - THD (table header) tag 57
    - REFID attribute 57
  - the starter set
    - SYSVARs supported 120
  - time, symbol for 113
  - title citation (CIT) tag 46
  - title of document 85
  - title page (TITLEP) tag 84
  - title page (T) SYSVAR 120
  - TITLE tag 85
  - TITLEP (title page) tag 84
  - TLIST (list of tables) 87
    - how to produce one 87
  - TOC (table of contents) tag 87
  - token
    - definition 184
  - top margin
    - definition 184
  - training courses v
  - translate table
    - definition 184
  - TRC
    - definition 184
  - TREF (table reference) tag 76
    - PAGE attribute 76
    - REFID attribute 76
  - TSO 7
    - definition 184
  - TWOPASS (SCRIPT command option) 129
  - type posture
    - definition 184
  - type size
    - definition 184
  - type style
    - definition 184
  - type weight
    - definition 184
  - type width
    - definition 184
  - typeface
    - definition 184
  - typeface family
    - definition 184
  - types of documents 5
  - typesei
    - definition 184
- ## U
- UL (unordered list) tag 17
    - COMPACT attribute 17
  - underscore
    - definition 184
  - unexpected formatting results 136
  - unformatted mode
    - definition 184
  - unformatted text
    - in a figure 33
    - in an example 31
  - unique identifier (ID)
    - definition 184
  - unit space
    - definition 184
  - unit, paragraph
    - definition 181
  - unmatched REFIDs 78
  - unordered list (UL) tag 17
    - COMPACT attribute 17
  - uppercase
    - definition 184
  - uppercase, forcing (on the terminal) 46

## Index

### V

- VALIGN attribute 53
  - on RDEF tag 53
- value attributes
  - how to specify 20
- value, initial
  - definition 179
- variable
  - definition 184
- variable text
  - definition 184
- vertical justification
  - definition 184
- vertical space units 39

### W

- W (write) SYSVAR 121
- WCGM
  - definition 184
- where to put index entries 101
- WIDTH attribute
  - on TABLE tag 55
- WIDTH attribute (on FIG tag) 34
- window
  - definition 184
- word processing equipment 6
- word space
  - definition 184
- word spacing
  - definition 184
- write (W) SYSVAR 121

### X

- X (cross-reference listing) SYSVAR 122
- XMP (example) tag 31
  - DEPTH attribute 31

## Numerics

- IBM 1403 Printer
  - highlighting on 46
- IBM 3800 Printing Subsystem
  - default fonts 133
  - default page dimensions 129
  - default segment library 128
  - font specification (CHARS option) 125
  - highlighting on 45
  - using the PROC attribute 108
- IBM 3820 Page Printer
  - default fonts 133
  - default page dimensions 129
  - default segment library 128
  - font specification (CHARS option) 125
  - highlighting on 45
  - using the PROC attribute 108

- IBM 4250 Printer
  - default fonts 133
  - default page dimensions 129
  - default segment library 128
  - font specification (CHARS option) 125
  - highlighting on 45
  - using the PROC attribute 108

## Special Characters

- . (period)
  - entering SCRIPT/VS control words 89
  - when required 19
- \* (comments) 91
- & (ampersand) 113
- &amp; symbol 113
- &date. symbol 113
- &gml. symbol 113
- &rbf. symbol 113
- &semi. symbol 113
- &time. symbol 113
- : (colon)
  - illustrating a GML tag 113
  - &gml. symbol 113
- ' (single quotation mark)
  - in a symbol's value 116
  - in an attribute value 145
- ; (semicolon) 139
  - illustrating the control word separator 113
  - symbol for 113
  - won't print 139



148, 148, 148, 148, 148, 148, 149, 149, 149, 149, 149,  
 149, 149, 149, 149, 149, 149, 149, 149, 149, 149, 149,  
 149, 149, 149, 149, 149, 149, 150, 150, 150, 150, 150,  
 150, 150, 150, 150, 150, 150, 150, 150, 150

FOOT	IA6USSET	i	
4COLHD	IA6USSET	i	151, 152, 154
3COLHD	IA6USSET	i	151, 152, 152, 153, 153
C18DOC	IA6USSET	i	151, 151
C18HEAD	IA6USSET	i	151, 151, 151, 152
C18TEXT	IA6USSET	i	152, 152
C18DISP	IA6USSET	i	152, 152
C18LIST	IA6USSET	i	154, 154
C18FOOT	IA6USSET	i	153, 153, 153
C18TABL	IA6USSET	i	152, 153
HEADER	IA6UMSTR SCRIPT	iv	iv
LIBG	IA6UMSTR SCRIPT	iv	iv, iv, iv, iv, iv, iv, iv, iv, iv, iv

Figures
---------

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
IDXF	IA6UCH10	94	1 93
IDXG	IA6UCH13	123	2

Headings
----------

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
LIB	IA6UMSTR SCRIPT	iv	Publication Library Guide for the Document Composition Facility
LEGDCF	IA6UMSTR SCRIPT	v	DCF Publications iv
PART1	IA6UMSTR SCRIPT	1	Part One: Getting Started iii
INTRO	IA6UCH1	3	Chapter 1, Introduction 1
GML	IA6UCH2	5	Chapter 2, What Is GML? 1, 4
PART2	IA6UMSTR SCRIPT	9	Part Two: Tag Guide iii, 12
GS	IA6UCH3	11	Chapter 3, Paragraphs and Headings 9, 12
EXER1	IA6UCH3	15	Exercise: Paragraphs and Headings 169
LST	IA6UCH4	17	Chapter 4, Lists 9, 73
DL	IA6UCH4	21	Definition Lists 45
EXER2	IA6UCH4	29	Exercise: Lists 170
XF	IA6UCH5	31	Chapter 5, Examples and Figures 9
FGS	IA6UCH5	32	Figures

		31	
PS	IA6UCH5	39	PostScript Images 38
SPACE	IA6UCH5	39	Space Units 21, 22, 31, 34, 35, 54, 55
XFFX	IA6UCH5	43	Exercise: Examples and Figures 171
HPQ	IA6UCH6	45	Chapter 6, Highlighting, Citing, Noting, and Quoting 9, 21, 24, 25, 52
EXER5	IA6UCH6	50	Exercise: Highlighting, Citing, Noting, and Quoting 172
TABLES	IA6UCH7	51	Chapter 7, Tables 9
SROW	IA6UCH7	56	Starting a Row 56
GORP	IA6UCH7	66	Using the ARRANGE and CWIDTHS Attributes 54, 54
TBLEXER	IA6UCH7	70	Exercises: Tables 173
XREF	IA6UCH8	73	Chapter 8, Cross-References and Footnotes 9, 12, 19, 33, 34, 55
UREF	IA6UCH8	78	Unmatched References
EXER4	IA6UCH8	80	Exercise: Cross-References and Footnotes 174
WUF	IA6UCH9	83	Chapter 9, Overall Document Structure 9, 141
FMAT	IA6UCH9	84	The Front Matter
PREF	IA6UCH9	86	Preface
MIFS	IA6UCH9	89	Combining Input Files 141
IDX	IA6UCH10	93	Chapter 10, Indexing 9, 88, 126
MSC	IA6UCH11	107	Chapter 11, Process-Specific Controls 9, 116, 119, 123
PART3	IA6UMSTR SCRIPT	111	Part Three: Document Formatting iii
SYM	IA6UCH12	113	Chapter 12, Symbols 110, 111
SVAR	IA6UCH13	119	Chapter 13, Formatting Instructions 14, 74, 111
SYSV	IA6UCH13	120	Instructions to the Starter Set (SYSVAR) 108, 129, 141
SCMD	IA6UCH13	123	Instructions to SCRIPT/VS 39, 93, 119
OPTIO	IA6UCH13	124	SCRIPT Command Options
PASS	IA6UCH13	125	Processing Options:
FPASS	IA6UCH13	126	FPASSES(n) 87, 129, 138
NOSEG	IA6UCH13	126	NOSEGLIB
PSDEV	IA6UCH13	127	PSOUT
TWPASS	IA6UCH13	129	TWOPASS 87, 126
AFONT	IA6UCH13	133	Font Requirements for Page Printers 125, 126
PFONT	IA6UCH13	133	Font Requirements for PostScript Devices 125, 126

OVERID	IA6UCH13	134	Overriding the Default Font Using the CHARS Option 125
ERHAND	IA6UCH14	135	Chapter 14, Error Handling 111
SSMSG	IA6UCH14	135	Starter Set and SCRIPT/VS Messages
ERR	IA6UCH14	136	Common Problems – Symptoms and Solutions 87, 135
SYMPUN	IA6UCH14	136	Symptom: Unexpected Formatting Results 110
SYMPTOC	IA6UCH14	138	Symptom: Page Number References Are Wrong
XRL	IA6UCH15	141	Chapter 15, Cross-Reference Listing 79, 79, 80, 111
PART4	IA6UMSTR SCRIPT	143	Part Four: Tag Reference iii
REF1	IA6UCH16	145	Chapter 16, General Guidelines for Markup Entry 143
ASUMM	IA6UCH17	147	Chapter 17, Alphabetic Summary of Tags 143
SUMDE	IA6UCH18	151	Chapter 18, Summary by Document Elements 143
APPS	IA6UMSTR SCRIPT	155	Part Five: Appendixes iii
SAMP2	IA6UAPXA	157	Appendix A, A Sample GML Document 155
SOLVE	IA6UAPXB	169	Appendix B, Solutions for Exercises 155
GSX	IA6UAPXB	169	Markup for Paragraphs and Headings 15
LSTX	IA6UAPXB	170	Markup for Lists 29
XFX	IA6UAPXB	171	Markup for Examples and Figures 43
HPQX	IA6UAPXB	172	Markup for Highlighting, Citing, Noting, and Quoting 50
XTB	IA6UAPXB	173	Markup for Tables
1ST	IA6UAPXB	173	First Table Exercise 70
2ND	IA6UAPXB	173	Second Table Exercise 71
XREFX	IA6UAPXB	174	Markup for Cross-References and Footnotes 80
GLS	IA6UGLOS	175	Glossary 24

<b>Index Entries</b>
----------------------

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
SOA	IA6UMSTR SCRIPT	xvii	(1) Summary of Amendments
CHG	IA6UMSTR SCRIPT	xvii	(1) changes with Release 3.2
HI3800	IA6UCH6	45	(1) 3800 45, 108
HI4250	IA6UCH6	45	(1) 4250 45, 108
HI3820	IA6UCH6	45	(1) 3820 45, 108

HI1403	IA6UCH6	46	(1) 1403 46
TABLT	IA6UCH7	55	(1) TABLE tag 76
SV	IA6UCH13	120	(1) system variables (SYSVARs) for the starter set 120, 120, 120, 121, 121, 122, 122, 122
SVO	IA6UCH13	120	(1) SYSVARs (system variables) for the starter set 120, 120, 120, 121, 121, 122, 122, 122
SCMMD	IA6UCH13	123	(1) SCRIPT command 124, 125, 125, 126, 126, 126, 126, 127, 127, 127, 127, 129, 129, 129, 130, 132, 132, 132, 132, 133, 134
SYMPS	IA6UCH14	136	(1) symptoms and solutions 136, 136, 137, 137, 137, 138, 139, 139, 139, 139, 139
API1	IA6UGLOS	175	(1) API 175

#### List Items

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
HELP	IA6UCH8	74	4
THREE	IA6UCH8	80	3

#### Footnotes

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
SUPPORT	IA6UMSTR SCRIPT	iv	1 iv
ADM	IA6UMSTR SCRIPT	iv	2 iv, iv
DLF	IA6UMSTR SCRIPT	xvii	3 xvii
RFT	IA6UMSTR SCRIPT	xvii	4 xvii
DOCADM	IA6UCH2	5	6 5
PER	IA6UCH3	11	7 11
INPUT	IA6UCH3	12	8 12
QUOTE	IA6UCH3	13	9 13
FNEX1	IA6UCH3	15	10 15, 15, 15
ENDTAG	IA6UCH4	19	11 19
FNEX2	IA6UCH4	29	12 29, 29, 30
FNEX3	IA6UCH5	43	13 43, 43, 43
FNEX5	IA6UCH6	50	14 50, 50, 50
FNEXT1	IA6UCH7	70	16 70, 70, 70
FNEXT2	IA6UCH7	71	17 71, 71, 71



FNXMP	IA6UCH8	78	19	78
EXER4	IA6UCH8	80	20	80
FNEX4	IA6UCH8	81	21	81, 81, 81
FILES	IA6UCH9	89	22	89
FNEXALL	IA6UCH9	91	23	91, 91, 91
FNEXAL2	IA6UCH10	105	24	105, 105, 105
PSCRULE	IA6UCH12	116	25	116
XCMD	IA6UCH13	124	27	124
ATMS	IA6UCH13	124	28	124
3ATMS	IA6UCH13	126	30	126
PROF3	IA6UCH13	127	31	127
ATMS8	IA6UCH13	132	32	132, 132
ATM2	IA6UCH13	133	33	133
MONOS	IA6UCH13	133	34	133

<b>Revisions</b>
------------------

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
REV1	IA6USSET	i	ii, ii, ii, ii, xvii, xvii, 54, 54, 71, 71, 173, 173
RCF	IA6USSET	i	
BAR	IA6USSET	i	
NUM	IA6USSET	i	

<b>Spots</b>
--------------

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>
LGDCF	IA6UMSTR SCRIPT	v	(no text)
DLIST	IA6UCH4	21	(no text) 23
PSDEV	IA6UCH5	39	(no text)
LSTX	IA6UCH8	74	(no text) 75
FXREF	IA6UCH8	75	(no text) 76
TXREF	IA6UCH8	76	(no text) 77
SI	IA6UCH13	128	(no text)
DTS	IA6UCH13		128, 128, 128

		130	(no text)	108, 125, 130
LPSD	IA6UCH13	132	(no text)	125

Tables				
--------	--	--	--	--

<u>id</u>	<u>File</u>	<u>Page</u>	<u>References</u>	
TAB2	IA6UCH7	69	3	68
TBTEST	IA6UCH8	76	5	
JB	IA6UCH8	80	6	
LDUPDLT	IA6UCH13	130	7	80
LDUPDPG	IA6UCH13	131	8	
LDUPDPS	IA6UCH13	132	9	
ALSUM	IA6UCH17	148	10	147

Processing Options	
--------------------	--

Runtime values:

```

Document fileid ..... IA6UMSTR SCRIPT
Document type ..... USERDOC
Document style ..... DCFTAB
Profile ..... EDFPRF20
Service Level .. ..... 0006
SCRIPT/Vs Release ..... 3.2.1
Date ..... 89.10.06
Time ..... 10:51:44
Device ..... 3820A
Number of Passes ..... 4
Index ..... YES
SYSVAR D ..... YES
SYSVAR G ..... INLINE
SYSVAR X ..... YES

```

Formatting values used:

```

Annotation ..... NO
Cross reference listing ..... YES
Cross reference head prefix only ..... NO
Dialog ..... LABEL
Duplex ..... YES
DVCF conditions file ..... (none)
DVCF value 1 ..... (none)
DVCF value 2 ..... (none)
DVCF value 3 ..... (none)
DVCF value 4 ..... (none)
DVCF value 5 ..... (none)
DVCF value 6 ..... (none)
DVCF value 7 ..... (none)
DVCF value 8 ..... (none)
DVCF value 9 ..... (none)
Explode ..... NO
Figure list on new page ..... YES
Figure/table number separation ..... YES
Folio-by-chapter ..... NO
Head 0 body text ..... (none)
Head 1 body text ..... Chapter
Hyphenation ..... NO
Justification ..... NO
Language ..... ENGL
Layout ..... OFF
Leader dots ..... YES
Master index ..... (none)
Partial TOC (maximum level) ..... 4
Partial TOC (new page after) ..... INLINE
Print example id's ..... NO
Print cross reference page numbers ..... YES
Process value ..... (none)

```

Punctuation move characters ..... ,  
Read cross-reference file ..... (none)  
Running heading/footer rule ..... NONE  
Show index entries ..... NO  
Table of Contents (maximum level) ..... 4  
Table list on new page ..... YES  
Title page (draft) alignment ..... CENTER  
Write cross-reference file ..... (none)

**Imbed Trace**

Page 0	IA6USSET
Page 1	IA6UCH1
Page 4	IA6UCH2
Page 9	IA6UCH3
Page 16	IA6UCH4
Page 30	IA6UCH5
Page 44	IA6UCH6
Page 50	IA6UCH7
Page 71	IA6UCH8
Page 81	IA6UCH9
Page 91	IA6UCH10
Page 105	IA6UCH11
Page 111	IA6UCH12
Page 117	IA6UCH13
Page 134	IA6UCH14
Page 139	IA6UCH15
Page 143	IA6UCH16
Page 146	IA6UCH17
Page 150	IA6UCH18
Page 155	IA6UAPXA
Page 168	IA6UAPXB
Page 174	IA6UGLOS

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

**Note:** IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

- |   | Yes                      | No                       |
|---|--------------------------|--------------------------|
| • Does the publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the information:         |                          |                          |
| Accurate?                               | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to read and understand?            | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to retrieve?                       | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use?           | <input type="checkbox"/> | <input type="checkbox"/> |
| Legible?                                | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete?                               | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated?                       | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level?       | <input type="checkbox"/> | <input type="checkbox"/> |
| • Do you use this publication:          |                          |                          |
| As an introduction to the subject?      | <input type="checkbox"/> | <input type="checkbox"/> |
| As a reference manual?                  | <input type="checkbox"/> | <input type="checkbox"/> |
| For advanced knowledge of the subject?  | <input type="checkbox"/> | <input type="checkbox"/> |
| To learn about operating procedures?    | <input type="checkbox"/> | <input type="checkbox"/> |
| As an instructor in class?              | <input type="checkbox"/> | <input type="checkbox"/> |
| As a student in class?                  | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation?              |                          |                          |

**Comments:**

To receive a reply provide your name, mailing address, and date:

---

---

---

---

Thank you for your input and cooperation.

**Note:** If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

SH20-9186-06

**Reader's Comment Form**



Fold and Tape

Please Do Not Staple

Fold and Tape



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT NO.40 ARMONK, NY  
POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation  
Department 588  
P. O. Box 1900  
Boulder, Colorado U.S.A. 80301-9191



Fold and Tape

Please Do Not Staple

Fold and Tape





Program Number  
5748-XX9

File Number  
S370-40

SH20-9186-06

