Note

Before using this information and the product it supports, read the information in "Notices" on page 591.

This edition applies to Page Printer Formatting Aid/370 Version 1 Release 1 Modification 0, Program Number 5688-190, and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces S544-5284-12.

© Copyright IBM Corporation 1985, 2016.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Figures ........................................ vii
Tables .......................................... xi

About this publication ........................ xiii
Who Should Use This Publication? ........ xiii
Abbreviations ................................. xiii
Related Publications .......................... xiv
Reading Syntax Diagrams ..................... xv
Style Rules: ..................................... xv
Symbols: ......................................... xv
Required Parameters: .......................... xvi
Optional Parameters: ............................ xvi
Repeating Parameters: .......................... xvii
Fragment Elements .............................. xvii

Summary of changes .......................... xix

How to send your comments to IBM ....... xxi
If you have a technical problem .......... xxi

Part 1. What is PPFA? ......................... 1

Chapter 1. Introducing Page Printer .... 3
Formatting Aid .................................. 4
Summary of a Form Definition ............. 4
Summary of a Page Definition .............. 5
Formatting Output of Different Data File Types. . 6
Line-Data Files ................................ 6
Traditional Line Data ......................... 7
Record Format Line Data ..................... 7
Mixed-Data Files ................................. 7
MO:DCA-P Data Files ........................... 7
PPFA Concepts .................................. 8
Physical Page .................................. 8
Logical Page .................................... 8
Subpage .......................................... 8
PPFA Basic Terms ............................... 8
Printline ......................................... 8
Layout ........................................... 9
Direction ......................................... 9
Rotation ......................................... 10
Presentation ..................................... 10
N_UP Partitions .................................. 11
Modifications ..................................... 11
Definitions of Command, Subcommand, and Parameter .................................. 12
Commands ......................................... 12
Subcommands ................................... 12
Parameters ....................................... 12
Basic Controls in Traditional Line Data .. 12
Carriage Control Characters (CC) ........ 12
Table-Reference Characters (TRC) ........ 12

Record ID ........................................ 13
Basic Controls in Record Format Line Data .................................. 13
Carriage Control Characters (CC) .......... 13
Table-Reference Characters (TRC) ........ 13
Record ID ........................................ 13
Structured Fields in Line Data .................. 13
Invoke Data Map ................................ 14
Invoke Medium Map ............................. 14
Include Page Segment ........................ 14
Include Page Overlay .......................... 14
Include Object .................................. 14
Presentation Text ............................... 14
No Operation .................................... 14
Normal Duplex and Tumble Duplex ........ 14

Part 2. Examples of Using PPFA ............. 17

Chapter 2. Using Form Definition ....... 19
Commands ......................................... 19
Copy Groups and Subgroups ................ 19
Commands Required to Create a Form Definition .................. 20
Command Nesting Rules ....................... 20
Positioning a Logical Page on a Sheet .... 21
OFFSET Subcommand with Rotated Print Direction .............. 22
Specifying Copies and Electronic Overlays .......... 22
Overlap Names .................................. 23
Printing Constant Forms ..................... 24
Duplex Printing .................................. 25
Duplex Printing in Portrait and Landscape Presentations ........ 27
Specifying Page Presentation on Continuous-Forms Printers ........ 27
When to Use the PRESENT and DIRECTION Subcommands .... 30
When the PRESENT and DIRECTION Subcommands Are Not Required .... 30
The DOWN Direction for Continuous Forms Printers .... 31
Print Quality Control ........................... 32

Chapter 3. Using Page Definition ...... 33
Commands for Traditional Line Data .... 33
Page Formats within Page Definitions .... 33
Page Definition Command Nesting ........ 34
Command Nesting Rules ....................... 34
Defining Logical Page Size ................... 34
Positioning the First Line of Data .......... 35
Changing Logical Page Print Direction .... 37
Printing Line Data on a Print Server Printer .... 39
Processing Fields .............................. 43
POSfITION Subcommand in Processing Fields Example ........ 44
FIELD Command in Processing Fields Example .......... 44
Varying Fonts on a Page ....................... 45
<table>
<thead>
<tr>
<th>Appendix B. More about Direction</th>
<th>473</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix C. Differences in</td>
<td></td>
</tr>
<tr>
<td>Measurements and REPEATs with</td>
<td></td>
</tr>
<tr>
<td>AFP Utilities</td>
<td>475</td>
</tr>
<tr>
<td>Appendix D. More About Bar Code</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>477</td>
</tr>
<tr>
<td>Bar Code Data</td>
<td>477</td>
</tr>
<tr>
<td>MOD Parameter</td>
<td>485</td>
</tr>
<tr>
<td>Check Digit Calculation Method</td>
<td>507</td>
</tr>
<tr>
<td>Bar code Exception Conditions</td>
<td>512</td>
</tr>
<tr>
<td>Data-Check Exceptions</td>
<td>517</td>
</tr>
<tr>
<td>Data Matrix Special-Function Parameters</td>
<td>517</td>
</tr>
<tr>
<td>MaxiCode Special-Function Parameters</td>
<td>522</td>
</tr>
<tr>
<td>PDF417 Special-Function Parameters</td>
<td>526</td>
</tr>
<tr>
<td>QR Code Special-Function Parameters</td>
<td>531</td>
</tr>
<tr>
<td>Appendix E. Set Media Origin (SMO)</td>
<td>539</td>
</tr>
<tr>
<td>Background</td>
<td>539</td>
</tr>
<tr>
<td>FORMDEF PRESENT and DIRECTION Parameters</td>
<td>542</td>
</tr>
<tr>
<td>The SMO Reference Pages</td>
<td>543</td>
</tr>
<tr>
<td>Appendix F. PPFA Keywords</td>
<td>561</td>
</tr>
<tr>
<td>Appendix G. PPFA Media Names</td>
<td>563</td>
</tr>
<tr>
<td>Appendix H. Fill Patterns for</td>
<td></td>
</tr>
<tr>
<td>DRAWGRAPHIC Commands</td>
<td>565</td>
</tr>
<tr>
<td>Appendix I. PPFA Messages and</td>
<td></td>
</tr>
<tr>
<td>Codes</td>
<td>567</td>
</tr>
<tr>
<td>PPFA Messages and Their Meanings.</td>
<td>567</td>
</tr>
<tr>
<td>Appendix J. Accessibility</td>
<td>587</td>
</tr>
<tr>
<td>Accessibility features</td>
<td></td>
</tr>
<tr>
<td>Consult assistive technologies</td>
<td></td>
</tr>
<tr>
<td>Keyboard navigation of the user interface</td>
<td></td>
</tr>
<tr>
<td>Dotted decimal syntax diagrams</td>
<td></td>
</tr>
<tr>
<td>Notices</td>
<td>591</td>
</tr>
<tr>
<td>Policy for unsupported hardware</td>
<td>592</td>
</tr>
<tr>
<td>Minimum supported hardware</td>
<td>593</td>
</tr>
<tr>
<td>Trademarks</td>
<td>593</td>
</tr>
<tr>
<td>Glossary</td>
<td>595</td>
</tr>
<tr>
<td>Index</td>
<td>613</td>
</tr>
</tbody>
</table>
Figures

1. Form Definition and Page Definition Environment ............................................. 4
2. Formatted and Unformatted Print Records ...................................................... 6
3. Example of Record Format Line Data .............................................................. 7
4. Baseline Direction and Inline Direction .......................................................... 10
5. Portrait and Landscape Presentations .............................................................. 11
6. Origin of Logical Page ....................................................................................... 21
7. Origin of a Logical Page on a 5900 Sheet .......................................................... 22
8. The Meaning of OFFSET Parameters within a Landscape Page ....................... 22
9. Form Definition with Two Overlays .................................................................. 23
10. Two Electronic Overlays Incorporated into Two Subgroups .............................. 24
11. Six-Page Formatted Data File ......................................................................... 25
12. Result of Using a Pair of FRONT and BACK Subgroups .................................. 26
13. Form Definition EFGH Using DUPLEX with BOTH ........................................ 27
14. DUPLEX NORMAL: Portrait and Landscape Presentation ............................... 28
15. Result when Either TUMBLE or RNORMAL Is Specified .................................. 29
16. Narrow and Wide Continuous Forms ................................................................ 30
17. The Results of Not Specifying PRESENT LANDSCAPE and DIRECTION DOWN on a Continuous Forms Printer .................................................. 31
18. The Results of Specifying PRESENT LANDSCAPE and DIRECTION DOWN on a Continuous Forms Printer .................................................. 32
19. Logical Page Dimensions .............................................................................. 35
20. LINEONE Coordinates .................................................................................. 36
21. Logical Page Print Directions in Relation to Origin .......................................... 38
22. Line-Data File ................................................................................................. 39
23. Data File Printed on a Line Printer .................................................................... 40
24. Printout Examples Specifying POSITION MARGIN TOP .................................. 41
25. Printout Example Specifying POSITION MARGIN 4.1 ..................................... 41
26. Printout Example Specifying POSITION MARGIN TOP and POSITION MARGIN 4.1 42
27. Field-processing example for the PRINTLINE command ............................... 43
28. Unformatted Print Data File ........................................................................... 43
29. Data Arranged on the Printed Page ................................................................ 44
30. Data File Printed Using a Single Font ............................................................. 45
31. Font Change Using TRCREF Command ........................................................... 46
32. Font Change Using FONT Commands and Subcommands ............................. 47
33. A Printout with More Than One Line Direction .............................................. 48
34. Field Direction .................................................................................................. 49
35. Character Rotation ............................................................................................ 50
36. Example of Assumed Data File and Rotation Specifications ........................... 50
37. AFP Printer Tate Presentation .......................................................................... 51
38. Multiple-Up Page Layout .................................................................................. 52
40. Sample Page Header and Trailer .................................................................... 60
41. Sample Commands and Data With Delimiters ................................................. 61
42. Sample Page Formatting ................................................................................... 62
43. Logical Page Dimensions ............................................................................... 65
44. Logical Page Print Directions in Relation to Origin ........................................... 67
45. Relationship of Margin Definition to Text Orientation ...................................... 69
46. Field-processing example for the LAYOUT command ..................................... 70
47. Unformatted Print Data File ............................................................................ 70
48. Data Arranged on the Printed Page ................................................................ 70
49. A Printout with More Than One Line Direction .............................................. 72
50. Field Direction .................................................................................................. 73
51. Line Data for Single Font Example .................................................................. 74
52. Data File Printed Using a Single Font ............................................................. 74
53. Line Data for Two Font Example ..................................................................... 74
54. Font Change Using FONT Commands and Subcommands ............................ 75
55. Character Rotation ............................................................................................ 76
56. Example of Assumed Data File and Rotation Specifications ........................... 76
57. AFP Printer Tate Presentation .......................................................................... 77
58. Bank Statement (Page 1 of 2) ......................................................................... 78
59. Bank Statement (Page 2 of 2) ......................................................................... 79
60. Statement that Uses the Repeating Box Option .............................................. 86
61. XML Data Elements ......................................................................................... 91
62. XML Data File (Part 1 of 5) ............................................................................. 93
63. XML Data File (Part 2 of 5) ............................................................................. 94
64. XML Data File (Part 3 of 5) ............................................................................. 95
65. XML Data File (Part 4 of 5) ............................................................................. 96
66. XML Data File (Part 5 of 5) ............................................................................. 97
67. XML Data Printed Output (Part 1 of 2) ............................................................ 98
68. XML Data Printed Output (Part 2 of 2) ............................................................ 99
69. Page Definition for XML Output (Part 1 of 7) .................................................. 100
70. Page Definition for XML Output (Part 2 of 7) .................................................. 101
71. Page Definition for XML Output (Part 3 of 7) .................................................. 102
72. Page Definition for XML Output (Part 4 of 7) .................................................. 103
73. Page Definition for XML Output (Part 5 of 7) .................................................. 104
74. Page Definition for XML Output (Part 6 of 7) .................................................. 105
75. Page Definition for XML Output (Part 7 of 7) .................................................. 106
76. Electronic Overlay and Data File for a Sales Report ........................................ 108
77. Sales Report ..................................................................................................... 109
78. Selective Suppression ...................................................................................... 110
79. Input for the Corporate Version of an Individual Sales Report ...................... 111
80. The Corporate Version of the Sales Report with Fixed Text ............................ 113
81. Input for a New Report Produced from the Combined Data Files ................ 114
82. The Sales and the Commission Reports ......................................................... 115
83. Jog Output Example ......................................................................................... 130
84. Duplex Output .................................................................................................. 131

© Copyright IBM Corp. 1985, 2016
<table>
<thead>
<tr>
<th></th>
<th>Narrow Continuous Forms Printer Paper with a Medium Origin of X'00'</th>
<th>Narrow Continuous Forms Printer Paper with a Medium Origin of X'05'</th>
</tr>
</thead>
<tbody>
<tr>
<td>162</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>163</td>
<td>Narrow Continuous Forms Printer Paper with a Medium Origin of X'01'</td>
<td>Cut-Sheet Emulation for Continuous Forms Printer Wide and Narrow Paper Set Media Origin</td>
</tr>
<tr>
<td>164</td>
<td>Narrow Continuous Forms Printer Paper with a Medium Origin of X'04'</td>
<td>...</td>
</tr>
<tr>
<td>165</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>166</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>167</td>
<td>Fill Patterns for DRAWGRAPHIC Commands</td>
<td>...</td>
</tr>
</tbody>
</table>
### Tables

1. Form Definition Tasks .......................... 19
2. Duplex Specifications ......................... 29
3. Page Definition Tasks ........................ 33
4. Record Format Page Definition Tasks .......... 56
5. Form Definitions and Page Definition Tasks 107
6. Conditional Processing Tasks ............... 117
7. Object Types that can be referenced as Secondary Resources .................. 194
8. Character Length for PPFA Names ............ 202
9. XType and XOper values ....................... 222
10. Form Length (LEN) and Form Width (WID) for Cutsheet and Narrow Fanfold Paper 233
11. Form Length (LEN) and Form Width (WID) for Wide Fanfold Paper ............. 233
12. Object Types that can be referenced as Secondary Resources .................. 318
13. Non-OCA Objects supported by IOB ........... 378
14. Object Types that can be referenced as Secondary Resources .................. 378
15. Non-OCA Objects supported by IOB ........... 389
16. Object Types that can be referenced as Secondary Resources .................. 390
17. Non-OCA Objects supported by IOB ........... 437
18. Object Types that can be referenced as Secondary Resources .................. 437
19. The Effect of Additive DIRECTIONs on Formatting and Font Prefixes .......... 474
20. Differences in Measurements and REPEATs with AFP Utilities .................. 475
21. Valid Code Pages and Type Styles ............ 477
22. Valid Characters and Data Lengths ........... 478
23. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128 482
24. Modifier Values by Bar Code Type ............ 486
26. Table Shows How to Convert Data to Hex Values .................. 499
27. Check Digit Calculation Methods For Each Bar Code .......................... 507
28. Supported Sizes for a Data Matrix symbol .... 519
29. Caption ................................... 530
30. Supported Version for a QR Code symbol .... 536
31. Registered Media Types Sorted By Media Name .......................... 563
32. Return Codes ................................ 567
About this publication

This publication describes how to use the Page Printer Formatting Aid (PPFA) to create and compile page definitions and form definitions for printing or viewing files with Advanced Function Presentation products, such as Print Services Facility™.

Notes:
1. All PPFA/370 functions that are described are available on the z/OS® operating system. However, because functions on PSF/VM and PSF/VSE and PPFA/370 functions for VM and VSE are not updated, some of the PPFA/370 functions that are described are not available on the VM and VSE operating systems.
2. Not all of the functions that are provided by PPFA are supported in all print server licensed programs. See the information for the print server licensed program that you are using to determine which functions are supported. For more information about a specific environment, see [Appendix A, “System Dependencies for PPFA,” on page 467](#) for the steps that are required to process page definitions and form definitions.

Who Should Use This Publication?

This publication is for anyone who wants to use PPFA to create form definitions and page definitions (traditional and record format). This publication assumes that you are one of these:

- **A first-time user**
  You are using PPFA for the first time to create form definitions and page definitions. You are familiar with system commands, but you are not familiar with Print Services Facility (PSF) concepts and Page Printer Formatting Aid parameters. You need to read all of the information that is contained in this publication, and then use it as a reference.
  For more information about z/OS, VM, or VSE, see the Application Programming Guide for the operating system that you are using.

- **An intermediate user**
  You are familiar with print server concepts and with Page Printer Formatting Aid parameters and you know the difference between a logical page and a physical page. You already know how to create and use form definitions and page definitions. Use this publication as a reference to learn more about PPFA commands and syntax. Refer to the examples for useful information.

- **An advanced user**
  You understand print server concepts and have used PPFA to create form definitions and page definitions. You understand the use of data stream processing. You are using this publication mostly as a reference. Chapter 5, “Creating Complex Printouts,” on page 107 might be especially helpful.

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFP</td>
<td>Advanced Function Presentation</td>
</tr>
<tr>
<td>BCOCA</td>
<td>Bar Code Object Content Architecture</td>
</tr>
</tbody>
</table>
**CMR**  Color management resource  
**IOCA**  Image Object Content Architecture  
**IPDS**  Intelligent Printer Data Stream  
**LCDS**  Line Conditioned Data Stream  
**MO:DCA**  Mixed Object Document Content Architecture  
**MO:DCA-P**  Mixed Object Document Content Architecture-Presentation  
**PCS**  Profile Connection Space  
**PMF**  Print Management Facility  
**PFFA**  Page Printer Formatting Aid  
**PSF**  Print Services Facility

---

**Related Publications**

You can obtain the publications that are listed in this section from the [AFP Consortium web page](http://www.afpcinc.org) and the [z/OS Internet Library](http://www.ibm.com/systems/z/os/zos/bkserv/).

<table>
<thead>
<tr>
<th>Publication</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Advanced Function Presentation: Programming Guide and Line Data Reference</em></td>
<td>S544-3884</td>
</tr>
<tr>
<td><em>AFP Conversion and Indexing Facility User’s Guide</em></td>
<td>S550-0436</td>
</tr>
<tr>
<td><em>AFP: Printer Information</em></td>
<td>G544-3290</td>
</tr>
<tr>
<td><em>Bar Code Object Content Architecture (BCOCA) Reference</em></td>
<td>AFPC-0005</td>
</tr>
<tr>
<td><em>Color Management Object Content Architecture (CMOCA) Reference</em></td>
<td>AFPC-0006</td>
</tr>
<tr>
<td><em>Font Object Content Architecture Reference</em></td>
<td>S544-3285</td>
</tr>
<tr>
<td><em>Graphic Object Content Architecture for AFP Reference</em></td>
<td>AFPC-0008</td>
</tr>
<tr>
<td><em>IBM AFP Fonts: Font Summary</em></td>
<td>G544-3810</td>
</tr>
<tr>
<td><em>IBM AFP Fonts: Technical Reference for Code Pages</em></td>
<td>S544-3802</td>
</tr>
<tr>
<td><em>IBM Infoprint Fonts: Font Summary</em></td>
<td>G544-5846</td>
</tr>
<tr>
<td><em>Image Object Content Architecture Reference</em></td>
<td>AFPC-0003</td>
</tr>
<tr>
<td><em>Intelligent Printer Data Stream Reference</em></td>
<td>AFPC-0001</td>
</tr>
<tr>
<td><em>Mixed Object Document Content Architecture Reference</em></td>
<td>AFPC-0004</td>
</tr>
<tr>
<td><em>Overlay Generation Language/370 User’s Guide and Reference</em></td>
<td>S544-3702</td>
</tr>
<tr>
<td><em>Presentation Text Object Content Architecture Reference</em></td>
<td>SC31-6803</td>
</tr>
<tr>
<td><em>Print Services Facility/VM: Application Programming Guide</em></td>
<td>S544-3677</td>
</tr>
<tr>
<td><em>Print Services Facility/VM: System Programming Guide</em></td>
<td>S544-3680</td>
</tr>
<tr>
<td><em>Print Services Facility/VSE: Application Programming Guide</em></td>
<td>S544-3666</td>
</tr>
<tr>
<td><em>Print Services Facility/VSE: System Programming Guide</em></td>
<td>S544-3665</td>
</tr>
<tr>
<td><em>PSF for z/OS: AFP Download Plus</em></td>
<td>S550-0433</td>
</tr>
<tr>
<td><em>PSF for z/OS: Customization</em></td>
<td>S550-0427</td>
</tr>
<tr>
<td><em>PSF for z/OS: Diagnosis</em></td>
<td>G550-0428</td>
</tr>
<tr>
<td><em>PSF for z/OS: Download for z/OS</em></td>
<td>S550-0429</td>
</tr>
</tbody>
</table>
Reading Syntax Diagrams

The syntax for PPFA commands is shown by using graphic notation. To read the diagrams, move from left to right and top to bottom, following the main path line.

**Style Rules:**

Syntax diagrams use the following style rules to show how to enter commands and parameters:

- A word in uppercase must be spelled exactly as shown, but can be coded in any case. For example, in coding, FORMDEF, FormDef, or formdef are equivalent.
- A word in all italic, lowercase letters shows a parameter that you can replace. For example, name shows that you replace name with a resource name that is retained in the library.
- A parameter above the line shows the default parameter. For example, SBCS is the default parameter in the syntax diagram for the FONT command:

![Syntax Diagram Example]

**Symbols:**

Syntax diagrams use symbols to help you follow the flow of the information:

- Statements begin with:
  ![Statement Begin]
- And end with:
  ![Statement End]
- Statements longer than one line continue to a second line with:
  ![Second Line]
- Where they resume with:
  ![Resume]

**Required Parameters:**

A parameter that you must include is displayed on the main path line. For example, the syntax diagram for the SEGMENT command shows that you must follow SEGMENT with its required parameter:
If you can choose from two or more required parameters, the parameters are shown with the first choice on the main path line and the other choices on branch lines under it. For example, this partial syntax diagram shows the DIRECTION command:

```
DIRECTION
  ACROSS
  DOWN
  BACK
  UP
```

You must type the command in any of the following ways:

- DIRECTION ACROSS
- DIRECTION DOWN
- DIRECTION BACK
- DIRECTION UP

**Optional Parameters:**

Parameters that you can include with a command are displayed on the branch line below the main path line. For example, this partial syntax diagram shows the COPYGROUP command:

```
COPYGROUP
  name
  CUTSHEET - NO
  CUTSHEET - YES
```

You can type the command in one of these ways:

- COPYGROUP name1 CUTSHEET YES ;
- COPYGROUP name1 CUTSHEET NO ;
- COPYGROUP name1;

Branch lines can include branch lines of their own. For example, this is the partial syntax diagram for the SUBGROUP command with the optional BIN parameter:
Repeating Parameters:

An arrow on a line above a parameter means that you can either repeat the parameter or enter more than one of the listed parameters. An example of this is the partial syntax diagram for the SUPPRESSION subcommand in the SUBGROUP command:

```
SUPPRESSION name
```

The arrow above `name` means you can include one or more field name parameters with the SUPPRESSION command.

Fragment Elements

A syntax diagram can contain a section that either has too many items or groups to fit in the diagram or is used more than once. This section can be presented as a fragment, and given a label that corresponds to the section within the main diagram. An example of this is the syntax diagram for the FORMDEF subcommand with its fragmented OVERLAY subcommand.

Note: This FORMDEF diagram example also displays examples of some of the parameters that are mentioned in other diagrams.
Summary of changes

This publication contains additions and changes to information previously presented in Page Printer Formatting Aid (PPFA) User’s Guide, S544-5284-12. The technical additions and changes are marked with a revision bar (|) in the left margin.

The following information is new or updated:

- The note for the COMMENT subcommand in the FORMDEF command is updated for clarification. See “COMMENT qstring” on page 252.
- Minor text and formatting changes are made to support accessibility and translation.
How to send your comments to IBM

We appreciate your input on this documentation. Please provide us with any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

Use one of the following methods to send your comments:

Important: If your comment regards a technical problem, see instead “If you have a technical problem.”

• Send an email to mhvrdfs@us.ibm.com
• Send an email from the “Contact us” web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html)

Include the following information:
• Your name and address
• Your email address
• Your phone or fax number
• The publication title and order number:
  Page Printer Formatting Aid: User’s Guide
  S544-5284-13
• The topic and page number or URL of the specific information to which your comment relates
• The text of your comment.

When you send comments to IBM®, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one or more of the following actions:

• Visit the IBM Support Portal (support.ibm.com)
• Contact your IBM service representative.
• Call IBM technical support.
Part 1. What is PPFA?

This section contains Chapter 1, “Introducing Page Printer Formatting Aid,” on page 3, which describes:

- Form definitions and page definitions
- Basic data types for printing
- PPFA concepts
- PPFA terms and definitions
- Line data controls and structured fields
- Types of duplex printing
Chapter 1. Introducing Page Printer Formatting Aid

Page Printer Formatting Aid is a program that enables users of Advanced Function Presentation (AFP) products to create their own formatting resources, called form definitions and page definitions. The form definitions and page definitions are stored in libraries¹ as AFP resources. Using AFP resources requires IBM Print Services Facility (PSF) or AFP Conversion and Indexing Facility (ACIF), which merges resources with user data files. This merging creates a data stream for printing or viewing.

Using a form definition or a page definition that is created by PPFA requires you to do three steps:
1. Write a set of PPFA commands that define how to position the data or handle the physical sheets.
2. Run PPFA to build the specified page definition or form definition and store the output as resources in a library.
3. Use your print server to submit the print file, specifying the page definition and form definition that are needed to accomplish the results you want.

Notes:
1. All PPFA/370 functions that are described are available on the z/OS operating system. However, because functions on PSF/VM and PSF/VSE and PPFA/370 functions for VM and VSE are not updated, some of the PPFA/370 functions that are described are not available on the VM and VSE operating systems.
2. Not all functions that are provided by PPFA are supported in all printers and printer server licensed programs. Refer to the information for the printer and printer server licensed program that you are using to determine which functions are supported.

Figure 1 on page 4 shows how form definition and page definition relate to PSF. In the area inside the broken line represents steps 1 and 2. The area outside of the broken line shows how PSF merges resources with the specified print job to form a single print stream and sends it to a page printer.

¹ The term library includes z/OS and VSE libraries, and VM files.
Summary of a Form Definition

A PPFA command stream can contain commands for form definitions. A form definition specifies how the printer controls the processing of the physical sheets of paper. In a form definition, you can specify modifications that distinguish formatting one print job from another when both are derived from the same data. Form definitions are used for all print server print files regardless of data type.

Form definitions can specify the following functions:
- Position of a logical page on a physical page
• Duplex printing
• Inclusion of overlays, which substitute for preprinted forms
• Flash (the use of a forms flash - only on 3800 printers)
• Selection of the number of copies for any page of data
• Suppression (the exclusion of selected fields of data in one printed version of a page of data but not in another)
• Jog (the offset stacking of cut-sheet output or copy marking on continuous-forms output)
• Selection among paper sources in a cut-sheet printer
• Adjustment of the horizontal position of the print area on the sheet (only on 3800 printers)
• Quality (selection among print quality levels)
• Constant (allows front or back printing of a page without variable data)
• Printing one, two, three, or four logical pages on a single side of a page
• Post-processing controls, such as:
  – Selecting functions
    Selecting device-dependent functions that are defined by the post-processing device
• Finishing operations:
  – Center Fold In
  – Corner Staple
  – Edge Staple
  – Saddle Stitch (In and Out)
  – Separation Cut
  – Perforation Cut
  – Fold
  – Z-Fold
  – Punch
  – UP3i Finishing

Summary of a Page Definition

A page definition specifies how you want data to be positioned on the logical page. A page definition can control the following functions:
• Dimensions of the logical page
• Print direction of the logical page
• Print direction of text lines and fields relative to the logical page
• Conditional processing (different formats on different pages, based on content of data)
• Text line spacing (number of lines per inch)
• Location of individual text lines and fields
• Number of text lines per page
• Page segments for inclusion in printed output
• Overlays for inclusion in printed output (positioned anywhere on the page)
• Page-ejection points
• Fonts and font rotation that are used on a page
• Multiple-up printing (placing more than one subpage on one side of a single sheet)
• Colors to be used (on printers that support this function)
One and two dimensional bar codes (on printers that support this function)
External objects for inclusion in printed output (can be positioned anywhere on the page)
Preloading and preripping of external objects and overlays

**Formatting Output of Different Data File Types**

The basic types of data that are printed on the print server printers are:
- Line-data files
- Traditional line data
- Record format line data
- Mixed-data files
- MO:DCA-P data files
- XML data

Line-data files and mixed-data files require a page definition and a form definition. MO:DCA-P data files require only a form definition.

**Line-Data Files**

*Line data* is EBCDIC data that is arranged for printing on line printers. These records can contain line-printer control characters such as carriage control characters (CC or FCFC), table-reference characters (TRC), or only data. To compose pages for the page printer from line data, the print server separates the incoming print records into pages according to specifications in a page definition. A page definition is always required for printing line data with the print server. You can create your own page definition or use a page definition that is provided with the print server. The two types of line data are *traditional* and *record format*.

The line data input to the print server can consist of records that are fully formatted; it can consist of records that contain only the fields of data to be printed; or it can consist of records of both types. You can use the page definition resource to format fields of line data outside of the application program. See [PSF for z/OS: User’s Guide](#) for additional information.

The following example shows two types of line data. The first type is data that is arranged as it prints; the second is data that requires field processing.

**Formatted Print Records**

```
ROBINSON E. CRUSOE
123 ISLAND PLACE
KEY WEST, FL  33040
USA
```

**Unformatted Print Records**

```
Name field ; Street field ; City, State ; Zip ; Country
```

*Figure 2. Formatted and Unformatted Print Records*
The technique of mapping the unformatted data to locations on the output pages is known as field processing or record processing and is available by using page definition controls. Field processing is explained in detail in “Processing Fields” on page 43.

**Traditional Line Data**

Traditional line data is data that is formatted for printing on a line printer. Fully formatted line data can be printed on a line printer without a page definition; however, all line data needs a page definition to be printed on a page printer.

A traditional line data record can contain a 1-byte carriage control character and a 1-byte table reference character followed by the data to be printed. (With a line printer, the maximum number of data bytes in a single input record is 208. With a page printer, the maximum number is 32,768 bytes). See Chapter 3, “Using Page Definition Commands for Traditional Line Data,” on page 33 for additional information about using traditional line data.

**Record Format Line Data**

The record formatting function allows an application to specify a format identifier (record ID) with each set of data fields (data record). The format identifier references a specific layout format in a Page Definition (PAGEDEF). At print time, each layout format (referenced by a record ID in a data record) is retrieved from the PAGEDEF and used to position and format the associated data records or fields on the output page. The PAGEDEF can contain any number of layout formats. The application can use a PAGEDEF layout format to either insert an end of page when a specified last line point is exceeded on the output page or force an end of page. See Chapter 4, “Using Page Definition Commands for Record Format Line Data and XML Data,” on page 55 for information about using record format line data.

**Mixed-Data Files**

Mixed-data files consist of MO:DCA-P data and line data. Such files might specify the beginning and ending of pages and might contain page addresses and data controls for page printing. The line-data portion of such files must be formatted for page printers by page-definition controls.

**MO:DCA-P Data Files**

MO:DCA-P data files are formed into pages before the print server receives them. These files already contain the embedded controls for printing on page printers. They contain such things as page addresses and data controls for page printing functions.
Note: See Mixed Object Document Content Architecture Reference and Advanced Function Presentation Programming Guide and Line Data Reference for more information about MO:DCA-P data. User application programs can also generate MO:DCA-P data.

PPFA Concepts

The concepts of physical page, logical page, and subpage are basic to understanding form definition and page definition controls.

Physical Page

A physical page is the sheet of paper or other medium (a sheet of labels, for instance) that moves through the printer.

Logical Page

A logical page is the area that you define in a PPFA command stream as the space on the physical page where data is printed. The logical page is positioned in relation to the media origin. For more information about the media origin of your printer, see your printer documentation. The positioning of the logical page on the sheet of paper is described in “Positioning a Logical Page on a Sheet” on page 21.

You can use an N_UP command to place one, two, three, or four logical pages on a single sheet. N_UP is in contrast to multiple up, which places subpages on one logical page.

Subpage

A subpage is a part of a logical page on which line data can be placed. Subpages are used only with conditional processing. Multiple-up printing can be done with or without defining subpages. In the page definition, multiple subpages can be placed on the physical page based on changes in the print data. An example is the use of multiple-up printing, which is printing two or four pages on a single side of a sheet. For more information, see “Subpage Description and Processing” on page 121.

PPFA Basic Terms

The following terms have meanings that are special to PPFA:
• Printline
• Layout
• Direction
• Rotation
• Presentation
• N_UP partitions
• Modifications

Printline

Printline is a single line of text, and is the traditional command that is synonymous with the record formatting Layout command. In the formatting of line data, a printline is normally the output that is generated by one record in the print file. However, printlines and print records are not the same.

PRINTLINE commands in the PPFA page definition define the number and position of printlines on a page. Each record in the print file is written to a single printline on a page. Usually, one print record is written to each printline. However,
control information in the print data can specify that two or more print records be written to the same printline, providing overprinting. Controls also can specify that print records skip printlines. For example, a print record might skip the remaining printlines on a page and instead print on the first printline of a new page.

Layout

Layout specifies a single line of text, and is the record formatting command that is synonymous with the traditional Printline command. In the formatting of line data, a layout is normally the output that is generated by one record in the print file. However, layouts and print records are not the same.

LAYOUT commands in the PPFA page definition define the number and position of layouts on a page. Each record in the print file is written to a single layout on a page. Usually, one print record is written to each layout. However, control information in the print data can specify that two or more print records be written to the same layout, providing overprinting. Controls also can specify that print records skip layouts. For example, a print record might skip the remaining layouts on a page and instead print on the first layout of a new page.

Direction

Text can be printed in four print directions. A print direction is a combination of both inline and baseline directions. For each of the directions, characters can be printed in four rotations.

The line direction is the direction in which successive characters are added to a line of text. The four line directions are:

ACROSS
Text characters are placed in a line from left to right across the page.

DOWN
Text characters are placed in a line from top to bottom down the page.

BACK
Text characters are placed in a line from right to left across the page.

UP
Text characters are placed in a line from bottom to top up the page.

The baseline direction is the direction in which successive lines of text are added to a page. The four character rotations, measured clockwise around each inline direction, for each line direction are:

0°
90°
180°
270°

For example, the text in this paragraph is printed ACROSS the page, and its rotation is 0°.

Figure 4 on page 10 shows the four possible directions.
Rotation

Individual characters can be rotated. Character rotation can be 0°, 90°, 180°, or 270° relative to the inline direction of the printline or field.

Note: On the 3800 printers only, character rotation differs between *bounded-box fonts* and *unbounded-box fonts*. Bounded-box fonts rotate the fonts; unbounded-box fonts are rotated by selecting the correct font.

Presentation

Presentation describes the shape of the page as it is viewed by the reader. Figure 5 on page 11 shows an example of how text is presented (positioned) on the page. The two page presentations are *portrait* and *landscape*:

**Portrait**

Designed to be viewed with the short side at the top of the page.

**Landscape**

Designed to be viewed with the long side at the top of the page.
N_UP Partitions

Some printers allow the physical sheet of paper to be divided into equal-sized partitions. For two or three partitions, each sheet is divided along one or two lines that are equally spaced along the longer side of the sheet. The printer positions a logical page of print data in each partition, which enables printing multiple logical pages with different formats and modifications on a single sheet of paper.

The size and arrangement of the partitions on the sheet depends on the number of partitions and the shape and size of the paper. For two or three partitions, each sheet is divided at two or three points that are equally spaced along the longer side of the sheet. For four partitions, each sheet is equally divided both vertically and horizontally. See Chapter 7, “N_UP Printing,” on page 141 for more information.

Modifications

Modifications are sets of form definition controls that apply to one page of a data file. With these controls, you can:

- Define the type of duplex printing to be done.
- Define one, two, three, or four partitions for N_UP.
- Select an overlay.
- Suppress the appearance of a field.
- Select the forms flash option (only for the 3800 printer).
- Specify the number of copies for a set of modifications.
- Specify post-printing processing options.

You can specify different sets of modifications for the same page of data in one form definition. Therefore, in one print job you can specify a series of SUBGROUP commands. For example, a form definition with two SUBGROUP commands has two sets of modifications. The same page of data is printed for each set of modifications, resulting in a slightly different output for each printing.
Definitions of Command, Subcommand, and Parameter

Commands, subcommands, and parameters are terms that are used throughout this publication to refer to the contents of PPFA control statements. Chapter 10, “Form Definition Command Reference,” on page 207 and Chapter 11, “Page Definition Command Reference,” on page 271 describe these commands with all their applicable subcommands.

Commands

*Commands* are the major controls that compose form definitions and page definitions.

Subcommands

*Subcommands* are used to further define commands. The absence of subcommands means that the default values specified with those subcommands are used. Three command terms also appear as subcommand terms: FONT, OVERLAY, and SUPPRESSION. These subcommand terms further define other commands.

Parameters

You can specify *parameters* with subcommands or accept the defaults; valid entries and their defaults are shown in the command reference chapters.

Basic Controls in Traditional Line Data

The following line-printer controls can be included in a line data file and can be used by a page definition to enable AFP functions:

- Carriage control characters
- Table-reference characters
- Record IDs

Carriage Control Characters (CC)

Carriage control characters, which control line skipping, line spacing, and page ejection on line printers, are fields within line-data records. They are compatible with page printers when page definitions format the printed data. In page definitions, you can specify CHANNEL subcommands that correspond to carriage control characters corresponding to channels 1 - 12 in the data. When you do so, the carriage control characters operate just as they do in a line-printer environment.

**Note:** ASCII ANSI, ANSI, and EBCDIC (machine) handle carriage control characters differently. See the SPACE_THEN_PRINT subcommand listed in “Long Form Subcommands” on page 275 for more information.

Table-Reference Characters (TRC)

Table-reference characters (TRCs) control font selection in line-data output. Page definitions can be used to map table-reference characters to AFP fonts for use with page printers.

For more information about Table-reference characters, see the *Advanced Function Presentation: Programming Guide and Line Data Reference.*
Record ID

Record IDs are only used with the record formatting function.

---

Basic Controls in Record Format Line Data

*Record format* line data is a new form of line data that is supported by the print server and formatted by a page definition. With this format, each data record contains a 10-byte record identifier that selects the record descriptor (RCD) in a record format page definition that is used to format the line data. This RCD might contain a carriage control (CC) byte.

- Carriage control characters
- Table-reference characters (not applicable in record format)

Carriage Control Characters (CC)

The CC byte is required when record format data is mixed with MO:DCA-P data, but is ignored. The CC byte is optional for record format line data at all other times; however, if you enter it, you must inform the print server that it is there.

Many functions that are used in the line descriptor (LND) to format traditional line data are used in RCD to format record format line data. Others, such as header and trailer processing, are unique to RCDs.

Traditional line data is similar to record format line data in that neither is formatted into pages. However, traditional line data can be printed on line printers while record format line data cannot. For more information, see Chapter 4, “Using Page Definition Commands for Record Format Line Data and XML Data,” on page 55.

*Note:* ASCII ANSI, ANSI, and EBCDIC (machine) handle carriage control characters differently. See the SPACE_THEN_PRINT subcommand listed in "Long Form Subcommands” on page 275 for more information.

Table-Reference Characters (TRC)

Table-reference characters (TRCs) cannot be used in record formatted line data.

Record ID

Record IDs are only used with the record formatting function. They are in the first 10 characters of each line data record, and control the layout type that is selected for each specified record. These 10 characters are reserved for record IDs and are not included as part of a defined field or conditional area.

---

Structured Fields in Line Data

*Note:* Structured fields are not supported by XML data.

To use the full function of page definitions and form definitions, MO:DCA-P structured fields might be required in the users data. The following MO:DCA-P structured fields can be included in a line-data file to activate AFP functions:

- Invoke Data Map
- Invoke Medium Map
- Include Page Segment
- Include Page Overlay
- Include Object
Include Presentation Text (PTX)

No Operation (NOP)

**Note:** For information about mixed mode, see the Advanced Function Presentation: Programming Guide and Line Data Reference.

**Invoke Data Map**

Add the Invoke Data Map structured field to the line-data file at a point that requires switching from one page format to another. The term *data map* is the name that is used for the term *page format* in PSF publications and PSF terminology.

**Invoke Medium Map**

Add the Invoke Medium Map structured field to the line-data file at a point that requires switching from one copy group to another. The term *medium map* is the name that is used for the term *copy group* in PSF publications and PSF terminology.

**Include Page Segment**

Position the Include Page Segment structured field within the line data for placing the page segment on the page.

**Include Page Overlay**

Position the Include Page Overlay structured field within the line data for placing the overlay anywhere on the page.

**Include Object**

Position the Include Object structured field for placing an object that contains other object types (for example, IOCA or BCoca) for placing the object anywhere on the page.

**Presentation Text**

A presentation text object can be included in line data by using the Presentation Text (PTX) structured field, which is a self-contained object that consists of line spacing, page margin, data position, and font settings. See the AFP Programming Guide and Line Data Reference and the Presentation Text Object Content Architecture Reference for additional information.

**No Operation**

A No Operation (NOP) structured field can be placed in the line data stream. It can be used to insert information, such as a comment, into the data stream.

**Normal Duplex and Tumble Duplex**

Some page printers can print on both sides of a sheet, which is called *duplex* printing. Duplex printing can be done in four ways:

- Normal duplex
- Tumble duplex
- Rotated normal duplex
- Rotated tumble duplex

In normal duplex, both sides have the same orientation, as in most books. In tumble duplex, the back of each page is upside down compared to the front of the
page: the top of one side of the sheet is at the same edge as the bottom of the other side. With these two types of duplex, you can specify top binding or side binding of the printed pages.

Duplex also involves the commands RNORMAL (rotated normal) and RTUMBLE (rotated tumble), which are used with landscape-presentation pages to specify the type of duplex printing. See Figure 14 on page 28 and Figure 15 on page 29 for illustrations of duplex printing.
Part 2. Examples of Using PPFA

This section contains information about:

- Using form definition commands
- Using page definition commands
- Creating complex printouts
- Conditional processing
- N_UP printing
- AFP color management
Chapter 2. Using Form Definition Commands

A form definition is a resource that is used by the print server to specify how the printer controls the processing of the sheets of paper. With form definitions, you can do the tasks that are listed in Table 1.

Table 1. Form Definition Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Location of Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a form definition.</td>
<td>“Commands Required to Create a Form Definition” on page 20</td>
</tr>
<tr>
<td>Position a logical page.</td>
<td>“Positioning a Logical Page on a Sheet” on page 21</td>
</tr>
<tr>
<td>Specify landscape presentation.</td>
<td>“OFFSET Subcommand with Rotated Print Direction” on page 22</td>
</tr>
<tr>
<td>Specify copies and electronic overlays.</td>
<td>“Specifying Copies and Electronic Overlays” on page 22</td>
</tr>
<tr>
<td>Print constant forms.</td>
<td>“Printing Constant Forms” on page 24</td>
</tr>
<tr>
<td>Duplex print in two orientation.</td>
<td>“Duplex Printing” on page 25</td>
</tr>
<tr>
<td>Print portrait and landscape.</td>
<td>“Duplex Printing in Portrait and Landscape Presentations” on page 27</td>
</tr>
<tr>
<td>Specify the page presentation on</td>
<td>“Specifying Page Presentation on Continuous-Forms Printers” on page 29</td>
</tr>
<tr>
<td>continuous-forms printers.</td>
<td></td>
</tr>
</tbody>
</table>

Copy Groups and Subgroups

A single form definition can contain several subsets of page controls, called copy groups. Copy groups define each physical page in the file. When you are printing jobs in duplex, the copy group defines both sides of the physical paper. Copy groups, in turn, can contain up to 127 subgroups, each of which creates a different set of modifications for the same page of data.

A series of copy groups can be used where either the data or the printing requirements call for various page control schemes. Part of the file can be printed from one (bin) paper source and part from another. Part can be printed duplex; part can be printed simplex. Duplex commands can be specified for a printer that does not support this function. This command treats the two adjacent pages as duplexed. Various controls can be contained in one form definition that has several copy groups.

You can control the following options within a copy group:

- Position of the logical page on a sheet of paper
- Duplex printing
- Type of cut-sheet paper to be printed on (by choosing between paper input sources in page printers that have more than one paper source)
- Offset stacking or copy marking of parts of a print job in the output stacker
- Printing one, two, three, or four logical pages on a single side of a sheet
- Vendor-attached devices for post-processing functions to be done on the sheet
- Print-quality level
To access a new copy group within a form definition, you can:

- Add to your data file an Invoke Medium Map structured field immediately before the page of data that requires the new copy group.
- Use a page definition that specifies conditional processing. When you access a new copy group, printing begins on the next physical sheet of paper.

For more information on the Invoke Medium Map structured field, see *Mixed Object Document Content Architecture Reference*.

Subgroups allow the same page of data within a file to be printed more than once, using different sets of modifications each time the page is printed. One example is the printing of an invoice and a packing list from the same records in a data file.

The following modifications to the page of data can be specified in a subgroup:

- Selection of suppressed fields for the page
- Selection of overlays that are used with the page
- Selection of forms flash with the page (only on the 3800 printer)
- Selection of the modification for front, back, or both sides of a sheet
- Selection of the number of copies of the subgroup to print
- Selection of the input bin

**Commands Required to Create a Form Definition**

The following simplified command stream shows the proper nesting of commands and the sequence in which the commands must be entered when you are creating a form definition:

```
SETUNITS ]
FORMDEF
[SUPPRESSION ...]
[COPYGROUP ]
[OVERLAY ...]
[SUBGROUP ...]
[COPYGROUP ]
[OVERLAY ...]
[SUBGROUP ...]
```

**Notes:**

1. If the form definition has only one copy group, the COPYGROUP command can be omitted. The OVERLAY command then follows any SUPPRESSION command.
2. Indentations are used to improve readability.
3. **Complete definitions of commands are in Chapter 10, “Form Definition Command Reference,” on page 207.**

**Command Nesting Rules**

1. SUPPRESSION commands must be specified immediately after FORMDEF commands.
2. SUBGROUP commands are specified under their associated COPYGROUP command or under the FORMDEF command.
3. OVERLAY commands are specified immediately after COPYGROUP commands.
4. The first COPYGROUP command can be omitted in a form definition if the form definition has only one copy group, and if it contains no OVERLAY commands.
5. A SETUNITS command can be placed anywhere in the PPFA command stream and is in effect until another SETUNITS command is encountered.

6. More than one of each command can appear under one form definition.

7. If an OVERLAY occurs outside of a COPYGROUP (immediately after the FORMDEF), PPFA generates a COPYGROUP with the FORMDEF name and issues a warning message. This copy group becomes the first COPYGROUP and might not be the effect that you want.

---

**Positioning a Logical Page on a Sheet**

The example in this section shows how the OFFSET subcommand is used to position the logical page on the physical sheet. A logical page is the area on a sheet of paper where all printing occurs. You establish the logical page origin, the point nearest the media origin, with the OFFSET subcommand. The OFFSET subcommand requires two coordinates and might have four. The first x and y coordinate defines the position on the front of the sheet, and the second x and y coordinate defines the position on the back of the sheet. A sample form definition that specifies the logical page position for a simplex sheet is:

```
FORMDEF ABCD
   OFFSET 1 IN 1 IN ;
```

**Note:** The 1 IN 1 IN is an abbreviation for 1 INCH 1 INCH. PPFA supports a number of different units of measurement formats. See “Units of Measurement” on page 204 for all the different formats.

The example places the logical page origin 1 inch to the right of and 1 inch down from the media origin.

Figure 6 shows the meaning of the x and y coordinates. In writing an OFFSET subcommand, the first parameter specifies x; the second parameter specifies y. If the x and y are repeated for the offset of the back side of the physical page, the same applies. The x defines the horizontal offset; the y defines the vertical offset. In this example, the logical page direction is ACROSS. The arrows within the logical page indicate the inline direction for text on the page. The lines of text are added according to the baseline direction.

---

![Figure 6. Origin of Logical Page](image-url)
Figure 7 shows the meaning of $x$ and $y$ in a logical page specification for a 3900 sheet. The 3900 sheet does not have an unprintable area, but FORMDEFs supplied with the print server have a 1/6 inch offset.

OFFSET Subcommand with Rotated Print Direction

Figure 8 shows that the media origins and logical page origins do not change when the print direction of the page changes, although the way you view the page does change. The arrows within the logical page show the DOWN print direction—producing landscape page presentation.

Be careful to coordinate form definitions and page definitions when you change between portrait and landscape presentations.

Specifying Copies and Electronic Overlays

This example shows how to specify different electronic overlays in different subgroups. The electronic overlays that you specify are created separately by using a program such as IBM Overlay Generation Language/370, and are stored as resources in the overlay library. No positioning controls are needed in the form definition with an overlay; the overlays are merely named. The overlay contains its
own positioning data relative to the physical sheet. A form definition that contains two overlays might look like this example:

```
FORMDEF SLSCOM;
COPYGROUP SLSCOM;
OVERLAY SLSRPT M1001; /*LOCAL NAME AND USER-ACCESS NAME*/
OVERLAY M1002; /*USER-ACCESS NAME ONLY*/
SUBGROUP COPIES 2
  OVERLAY SLSRPT;
SUBGROUP COPIES 3
  OVERLAY M1002;
```

*Figure 9. Form Definition with Two Overlays*

The steps to write this form definition are:

1. Create a copy group.
   a. Write a COPYGROUP command.
   b. Write an OVERLAY command for each overlay.

2. Create two subgroups by writing two SUBGROUP commands. Each subgroup contains an OVERLAY subcommand that names one of the selected overlays.

**Note:** The overlays must be named in each copy group.

### Overlay Names

To identify overlays by name, you must be aware of the three possible names for an overlay: a local name (SLSRPT) and two system names (M1001, O1M1001). The *local name* is used only within the PPFA command stream and is optional. For example, SLSRPT is the local name in the first OVERLAY command in the *Figure 9* command stream.

The *system name* identifies an overlay in the library. It has two forms: the *user-access name* (M1001 in *Figure 9*) and the *library-resource name*. Of these two names, you use only the user-access name. PPFA automatically adds the O1 overlay prefix to the user-access name, which identifies the resource in the library. Therefore, an overlay that is referenced through a form definition that is built with PPFA must begin with the O1 prefix. An example of the result is O1M1001, the library-resource name.

You can make up your own local name for an overlay. However, the local name must be used in the OVERLAY subcommand in the subgroup if it is used in an OVERLAY command for the copy group. If it is not, the subgroup must specify the user-access name, as is done for overlay M1002 in *Figure 9*.

*Figure 9* which specifies copies and electronic overlays, also specifies the number of copies of each subgroup. More than one copy of printed output can be requested by placing in the SUBGROUP command the COPIES subcommand and the number of copies of the subgroup that you want. The example specifies that two copies of the first subgroup and three copies of the second subgroup are to be printed. See *Figure 10 on page 24* which shows the result of printing a job that includes the overlays that are specified in the example.
Printing Constant Forms

This example shows how to specify the constant-forms function using the CONSTANT command. The constant-forms function allows you to print overlays or a forms flash on blank pages without adding blank pages to your print job. Instead, the CONSTANT command generates blank pages on which to print the requested overlays and forms flash. These pages are called constant forms because no variable data from the print file is printed on the pages.

You specify the CONSTANT command for an entire copy group; you identify the overlays and forms flash in the subgroups of the copy groups.

In the following sample, form definition XMPXXX specifies that overlay XMP be printed on the back of each sheet with no variable data from the print job. The data from the print file is printed only on the front side of each sheet.

```
FORMDEF XMPXXX
  REPLACE YES
  DUPLEX NORMAL ;
  COPYGROUP XMPXY
    CONSTANT BACK ;
    OVERLAY XMP;
    SUBGROUP FRONT ;
    SUBGROUP BACK
      OVERLAY XMP;

PAGEDEF XMPXXX
  REPLACE YES ;
  FONT NORMALFONT GT10 ;
  PAGEFORMAT XMPXXX ;
  PRINTLINE CHANNEL 1 REPEAT 20
    POSITION 1 1 ;
```

The steps to write this form definition are:

1. Create a copy group.

Figure 10. Two Electronic Overlays Incorporated into Two Subgroups
a. Specify duplex printing.
b. Specify printing of a constant form as the back side of each sheet.
c. Write an OVERLAY command.

2. Create two subgroups by writing two SUBGROUP commands. The subgroup for the back side specifies the overlay to be printed.

**Note:** If you do not specify an overlay in the subgroup for the back, the back side of each sheet is blank.

### Duplex Printing

Printing on both sides of a sheet (duplex printing) can be done in two ways: by the use of the FRONT and BACK subcommand combination or by the use of the BOTH subcommand. If FRONT and BACK are chosen, the number of copies that are requested for each must be the same.

To demonstrate some of the functions available for duplex printing, assume that you want to print a six-page data file (a simplified version is shown in Figure 11).

---

**Figure 11. Six-Page Formatted Data File**

Assume, too, that the file is already composed and formatted, so only a form definition is needed. The first form definition follows:

```
FORMDEF ABCD
    DUPLEX NORMAL ;
    OVERLAY AB ;
    SUBGROUP FRONT
        OVERLAY AB ;
    SUBGROUP BACK ;
```

In this command stream, form definition ABCD contains two subgroups, one specified with a FRONT subcommand and the other with a BACK subcommand.

By including a pair of FRONT and BACK subcommands within the copy group, you can specify that the front and back of printed sheets are to be controlled by different subgroups. The purpose is to allow modifications (for example, overlays or suppressions) to be separately specified for the front and back of sheets. **Figure 12 on page 26** shows the result of using this control where the front sheets have a header (OVERLAY AB) that the backs do not have.
The rules of the FRONT and BACK subcommands are:

- FRONT and BACK subgroups must be specified in pairs.
- Subgroups specifying FRONT must always immediately precede subgroups that specify BACK.
- FRONT and BACK subgroups must agree in the number of copies.

The BOTH subcommand also can be used with a form definition or a copy group that specifies duplex printing. An example of this type of form definition is:

```
FORMDEF EFGH
  DUPLEX NORMAL ;
  SUBGROUP BOTH
    COPIES 2 ;
```

The form definition EFGH contains only one SUBGROUP command.

**Notes:**

1. The copy group contains the subgroup, but if a form definition contains only one copy group, the copy group need not be specified.
2. With the BOTH subcommand, you specify only one subgroup: both sides of all sheets have the same modifications.
3. The form definition in the example does not put the same data on the front and back of the same sheet. Internally to PPFA, a single BOTH subgroup produces two subgroups. As a result, two pages of data (one for each internal subgroup) are processed before copy number 2 is made. For more information, see "SUBGROUP Command" on page 266.

Figure 13 on page 27 shows a sample print resulting from using the FORMDEF EFGH specifying BOTH to control the printing of the six-page (two copies) data file.
Duplex Printing in Portrait and Landscape Presentations

Duplex printing with PPFA and your print server printers offers several other options. This example shows the combination of portrait and landscape presentations with normal and tumble duplex printing.

**Note:** The terms normal, tumble, portrait, and landscape are used in this example. They are explained in this chapter and in the Glossary.

NORMAL and TUMBLE are parameters of a DUPLEX subcommand. For example, a form definition that specifies DUPLEX NORMAL might be written this way:

```
FORMDEF ABCD ;
 COPYGROUP ABCD
   DUPLEX NORMAL ;
 SUBGROUP BOTH
   COPIES 1 ;
```

Document A in Chapter 10, “Form Definition Command Reference,” on page 207 shows the result of a DUPLEX NORMAL specification in the portrait presentation. Document D shows the result of the same form definition when a landscape presentation is specified. The printout in landscape presentation is really in a
tumble-duplex format, having the tops (of the front side) and the bottoms (of the back side) of the logical pages toward the same edge of the sheet.

Although tumble duplex can be specified in this manner for landscape pages, another parameter, RTUMBLE (rotated tumble), exists to make the form definition look more sensible for use in landscape print jobs. It also produces the results that are shown in [Figure 14] depending on whether the form definition called for portrait or landscape presentation. For landscape, the form definition is written as follows:

```
FORMDEF ABCD
    PRESENT LANDSCAPE ;
    COPYGROUP ABCD
        DUPLEX RTUMBLE ;
    SUBGROUP BOTH
        COPIES 1 ;
```

**Note:** The example is for continuous printers. You must use N_UP for cut-sheet printers. In [Chapter 10, “Form Definition Command Reference,” on page 207](#), see the PRESENT subcommand of COPYGROUP.

![Figure 14. DUPLEX NORMAL: Portrait and Landscape Presentation](image)

The DUPLEX NORMAL and DUPLEX RTUMBLE controls produce the same result on the physical page. RTUMBLE is used to maintain an association between duplex specifications and logical page print direction. The same relationship exists between the RNORMAL and the TUMBLE parameters as exists between the NORMAL and the RTUMBLE parameters; that is, within the two sets the terms are interchangeable.

For example, you can write a form definition with DUPLEX TUMBLE as follows:
To help you remember, use Table 2.

Table 2. Duplex Specifications

<table>
<thead>
<tr>
<th>If the form definition duplex specification is...</th>
<th>And if the page definition direction is...</th>
<th>Then, the duplex printing result is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUPLEX NORMAL</td>
<td>ACROSS or BACK</td>
<td>normal duplex - portrait</td>
</tr>
<tr>
<td>DUPLEX RTUMBLE</td>
<td>DOWN or UP</td>
<td>tumble duplex - landscape</td>
</tr>
<tr>
<td>DUPLEX TUMBLE</td>
<td>ACROSS or BACK</td>
<td>tumble duplex - portrait</td>
</tr>
<tr>
<td>DUPLEX RNORMAL</td>
<td>DOWN or UP</td>
<td>normal duplex - landscape</td>
</tr>
</tbody>
</table>

Note: Other control combinations are not recommended.

Specifying Page Presentation on Continuous-Forms Printers

This example shows how to specify the page presentation (portrait or landscape) on printers that use continuous-forms paper. The page presentation is specified in the form definition with the PRESENT subcommand and the DIRECTION subcommand.
The PRESENT subcommand specifies how your pages are presented when they are printed and has two valid values: PORTRAIT and LANDSCAPE.

The DIRECTION subcommand specifies the inline direction in which your pages are formatted by the page definition (see “FIELD Command” on page 319) or by the program formatting the data. The DIRECTION subcommand has two valid values: ACROSS and DOWN.

The conditions in which you need to use these subcommands and some conditions in which they are not required are described in the following sections. For more information about how these subcommands work with data that is sent to specific printers, see the appropriate printer documentation.

To understand the description that follows, you must be aware of the difference between the two types of continuous forms: narrow and wide. Narrow forms are forms that have perforations on the shorter edge of the paper and tractor holes on the longer edge. Wide forms are forms that have perforations on the longer edge of the paper and tractor holes on the shorter edge. The two types of forms are illustrated in Figure 16.

![Figure 16. Narrow and Wide Continuous Forms](image)

When to Use the PRESENT and DIRECTION Subcommands

You need to use the PRESENT and DIRECTION subcommands if you are building a form definition that is used in these situations:

- With wide forms on a continuous forms printer when the print data is formatted in the DOWN print direction (see “The DOWN Direction for Continuous Forms Printers” on page 31).
- When you do not know which type of form (narrow or wide) is used on a continuous forms printer (see “The DOWN Direction for Continuous Forms Printers” on page 31).

Note: References to a continuous forms printer point of origin also apply to all continuous-forms printers except the 3800.

When the PRESENT and DIRECTION Subcommands Are Not Required

You do not need to use the PRESENT and DIRECTION subcommands if you are building a form definition that is used in these situations:
The DOWN Direction for Continuous Forms Printers

If your data is formatted in the DOWN print direction for landscape page presentation and is to be printed on wide forms on a continuous forms printer, you must specify LANDSCAPE on the PRESENT subcommand to produce readable output.

If PRESENT LANDSCAPE and DIRECTION DOWN are not specified on the FORMDEF command, the data is printed in the landscape presentation; however, the data is upside down, as shown in Figure 17. The data is upside down because the media origin for a continuous forms printer is on the same corner of the form, regardless of whether a narrow or wide form is used (see Figure 17).

![Diagram of Media Origin and Page Orientation](image)

Figure 17. The Results of Not Specifying PRESENT LANDSCAPE and DIRECTION DOWN on a Continuous Forms Printer

If PRESENT LANDSCAPE and DIRECTION DOWN are specified on the FORMDEF command, the data is printed as shown in Figure 18 on page 32. In this example, line data is formatted by using a page definition.

PRESENT LANDSCAPE and DIRECTION DOWN can also be specified for data that is formatted in the DOWN print direction and printed on narrow forms. Although PRESENT LANDSCAPE and DIRECTION DOWN do not need to be specified to produce readable output, specifying them lets you use the same form definition regardless of whether the data is printed on wide forms or narrow forms.

Note: If you are building a form definition that can be used with both wide and narrow forms, the left margin as viewed by the reader becomes the top margin from the printer's perspective (and vice versa). Because many printers have an

- With cut-sheet printers only
- With narrow forms only
- With the 3800 printer only
- With print data that is formatted in the BACK direction by the page definition or the program formatting the data
unprintable area at the margins, you must position the logical page with the
OFFSET subcommand in the form definition so data is placed in the unprintable
area on either wide or narrow forms.

Print Quality Control

If your printer has more than one print-quality selection, you can specify different
levels of print quality. For more information, see the publication for your printer.
Chapter 3. Using Page Definition Commands for Traditional Line Data

A page definition specifies how you want to position data on the logical page.

A page definition is a resource that is used by print servers to define the rules of transforming line data into composed pages and text controls for printing. With page definitions, you can do the tasks that are listed in Table 3.

Table 3. Page Definition Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Location of an Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a page definition.</td>
<td>“Page Definition Command Nesting” on page 34</td>
</tr>
<tr>
<td>Define logical page size.</td>
<td>“Defining Logical Page Size” on page 34</td>
</tr>
<tr>
<td>Position data on a logical page.</td>
<td>“Positioning the First Line of Data” on page 35</td>
</tr>
<tr>
<td>Change the print direction.</td>
<td>“Changing Logical Page Print Direction” on page 37</td>
</tr>
<tr>
<td>Print line data.</td>
<td>“Printing Line Data on a Print Server Printer” on page 39</td>
</tr>
<tr>
<td>Process fields.</td>
<td>“Processing Fields” on page 43</td>
</tr>
<tr>
<td>Change fonts.</td>
<td>“Varying Fonts on a Page” on page 45</td>
</tr>
<tr>
<td>Print in different directions.</td>
<td>“Printing Lines in Two Directions on a Page” on page 48</td>
</tr>
<tr>
<td>Print fields in two directions.</td>
<td>“Printing Fields in Two Directions on the Same Page” on page 48</td>
</tr>
<tr>
<td>Rotate fonts.</td>
<td>“Rotating Fonts” on page 49</td>
</tr>
<tr>
<td>Print kanji.</td>
<td>“Using Traditional Kanji Formatting” on page 51</td>
</tr>
<tr>
<td>Print multiple up.</td>
<td>“Printing Multiple-Up Pages” on page 51</td>
</tr>
</tbody>
</table>

Page Formats within Page Definitions

Just as form definitions can include more than one copy group, page definitions can include several page formats. Page formats use the same subcommands (except REPLACE) as page definitions, and if a subcommand is specified in a page format, it overrides the value that is specified in the page definition for the page format. A single page definition can contain multiple page formats. If pages in a file are to be formatted differently, specify more than one page format in your page definition. Within a page definition, page formats are generated in the order in which they are specified.

Using more than one page format to control different pages requires one of these:

• Adding the Invoke Data Map structured field to the data file each time you want to change page formats
• Using conditional processing.
Page Definition Command Nesting

The following simplified command stream shows the proper nesting of commands and the order in which they must be entered when you create a page definition:

```
[SETUNITS]
PAGEDEF
  [FONT]
  [OBJECT]
  [PAGEFORMAT]
    [TRCREF]
    [OBJECT]
    [SEGMENT]
    [OVERLAY]
    PRINTLINE
      [FIELD]
        [CONDITION]
      [ENDSUBPAGE]
  [SETUNITS]
```

Notes:
1. Brackets enclosing a command mean that the command is optional.
2. A command and its subcommands end with a semicolon.
3. Indentations are used to improve readability.
4. Complete definitions of all commands are included in Chapter 11, “Page Definition Command Reference,” on page 271.

Command Nesting Rules

1. FONT commands must be specified immediately after PAGEDEF commands.
2. A SETUNITS command can be specified anywhere in the PPFA command stream and is in effect until another SETUNITS command is specified.
3. OBJECT commands can appear after the FONT command, and before any PAGEFORMAT command (global objects) or after a specific PAGEFORMAT command. A global object is defined for all page formats in the page definition. Otherwise, the object is defined for the PAGEFORMAT in which it is specified.
4. TRCREF, SEGMENT, and OVERLAY commands must be specified under their associated PAGEFORMAT command.
5. The first PAGEFORMAT command can be omitted in a page definition, if the page definition has only one page format.
6. At least one PRINTLINE command is required.

Defining Logical Page Size

"Positioning a Logical Page on a Sheet" on page 21 shows how to establish the origin point of a logical page, relative to the media origin on a sheet of paper, by using the OFFSET subcommand. The following example shows you how to establish the width and height of the logical page relative to this origin point. This example illustrates how the dimensions of a logical page are determined by form definitions and page definitions.
FORMDEF ABCD
    OFFSET (1)(2) ;
PAGEDEF ABCD
    WIDTH (3)
    HEIGHT (4) ;
    PRINTLINE ;

Note: The parenthetical numbers represent dimensions. Figure 19 shows how these dimensions relate to the logical page.

Normally, all parameters consist of a number and a unit of measurement, for example, 6 IN. (See “Units of Measurement” on page 204 for information on units that are available.) Numbers can be specified with up to three decimal places. The PRINTLINE command is included in the example because at least one is required for all page definitions. See “PRINTLINE Command” on page 423 for more information.

Figure 19. Logical Page Dimensions

The OFFSET subcommand (1) (2) in the sample form definition establishes the corner or origin of the logical page relative to the physical sheet. The WIDTH and HEIGHT subcommands, (3) and (4), specify the dimensions of the logical page relative to the logical page origin.

Note: Be careful not to define a logical page larger than the physical sheet. PPFA does not check the size of the physical sheet.

“Positioning the First Line of Data” shows you two ways to position the first line of data on the page.

Positioning the First Line of Data

“Defining Logical Page Size” on page 34 shows how to define the size of a logical page. The next two examples show you how to position the first line of data inside the logical page by using the LINEONE subcommand. This subcommand position is relative to the logical page origin, as shown in Figure 20 on page 36. The two coordinates, (1) and (2), of the LINEONE parameter define the starting point for the first line of text.
This starting point works with the POSITION, MARGIN, and TOP subcommands (of the PRINTLINE command) to position lines of print on a page.

The defaults for LINEONE are:

- $x = 0$
- $y = 80\%$ of one line space from the top of the logical page. For example:
  - $80\%$ of $1/6$ inch if lines per inch (lpi) = 6
  - $80\%$ of $1/8$ inch if lpi = 8

These defaults leave room for the character ascenders in the first line of text.

**Note:** PPFA subtracts one logical unit (L-unit) from the $y$ value to compensate for the fact that the printer counts L-units beginning with the number 0. Therefore, if you use the LINEONE subcommand to specify the offsets to the first line in L-units (PELS is the measurement command for L-units), you must remember to subtract one L-unit from the $y$ offset value. This subtraction is necessary to prevent descenders on the last printed line from dropping off the bottom of the logical page.

The following examples illustrate two methods for positioning the first line of text:

1. The position of the first line of data defaults by specifying the SETUNITS command before the PAGEDEF command; for example:

   ```
   SETUNITS 1 IN 1 IN
   LINESP 8 LPI;
   FORMDEF ABCD
   OFFSET 0 .5;
   PAGEDEF ABCD
   WIDTH 7.5
   HEIGHT 10
   DIRECTION ACROSS;
   FONT GS12 GS12;
   PRINTLINE REPEAT 60
   FONT GS12
   POSITION 0 TOP;
   ```

   **Note:** The LINESP subcommand (of the SETUNITS command) must precede the PAGEDEF commands.

   If the LINESP subcommand follows the PAGEDEF command, PPFA then uses the default LINESP value to calculate the $y$ offset value, which is used to position the first line of print.
The default for the LINESP subcommand of the SETUNITS command is 6 lpi. If LINEONE is allowed to default, based on the LINESP default, the LINEONE value is 31 L-units:

\[
\text{LINEONE} = \left( \frac{240 \text{ L-units}}{6 \text{ lpi}} \right) \times 80\% - 1 \text{ L-unit} = 31 \text{ L-units.}
\]

This value is the vertical \((y)\) position of the printline because TOP is specified in a later POSITION subcommand. However, this value might cause the data to exceed the bottom boundary of the logical page if the LINESP value is changed later.

2. Another way that you can specify the starting position for the first print line is to specify LINEONE explicitly; for example:

```sql
FORMDEF ABCD
  OFFSET 0 .5;
PAGedef ABCD
  WIDTH 7.5
  HEIGHT 10
  LINEONE 0 PELS 23 PELS
  DIRECTION ACROSS;
SETUNITS 1 IN 1 IN
  LINESP 8 LPI;
FONT GS12 GS12;
PRINTLINE REPEAT 60
  FONT GS12
  POSITION 0 TOP;
```

In this example, the LINESP subcommand that follows the PAGEDEF command does not cause a data placement problem because the LINEONE command determines explicitly where the first line of text is positioned, and no default LINESP value is used:

\[
\text{LINEONE} = \left[ \left( \frac{240 \text{ L-units}}{8 \text{ lpi}} \right) \times 80\% \right] - 1 \text{ L-unit} = 23 \text{ L-units}
\]

If you use the LINEONE command to specify an absolute starting position for the first line, in L-units, you must remember to subtract one L-unit from that value.

---

**Changing Logical Page Print Direction**

Logical pages can have four different print directions: ACROSS, DOWN, BACK, and UP. This example shows that all four directions can be specified in relation to one offset specification:

```sql
FORMDEF ABCD
  OFFSET (1) (2) ;
PAGedef DEFG ;
PAGEFORMAT DEFG1
  WIDTH (3)
  HEIGHT (4)
  DIRECTION ACROSS ;
PRINTLINE ;
PAGEFORMAT DEFG2
  WIDTH (3)
  HEIGHT (4)
  DIRECTION DOWN ;
PRINTLINE ;
PAGEFORMAT DEFG3
  WIDTH (3)
  HEIGHT (4)
  DIRECTION BACK ;
PRINTLINE ;
PAGEFORMAT DEFG4
  WIDTH (3)
  HEIGHT (4)
  DIRECTION UP ;
PRINTLINE ;
```
One page definition is used to simplify the example, yet four logical pages are specified. The PAGEFORMAT commands create subsets of page definitions for each logical page.

**Note:** The page formats in this example require an Invoke Data Map structured field at the place in the data file where you want to change page formats. The PRINTLINE commands are required but are not relevant in the example.

The DIRECTION subcommand with one of its four direction parameters (ACROSS, DOWN, UP, or BACK) specifies the print direction of the logical page.

Figure 21 shows the format of each of the logical pages that are specified in the page definition with the direction specification of each. The pages with the ACROSS and BACK directions are in portrait presentation. The pages with the DOWN and UP directions are in landscape presentation.

![Logical Page Print Directions in Relation to Origin](image)

The media origins and logical page origins do not change with the presentation of the data on the page. The OFFSET subcommand of the form definition need not change. However, the width and height dimensions do change; that is, the WIDTH subcommand always governs the horizontal (inline) dimension as you view the page, and the HEIGHT subcommand always governs the vertical (baseline)
dimension whether the page is in portrait or in landscape presentation. Ensure that these specifications do not cause the logical page to cross the edge of the physical page.

However, if the DOWN direction is specified for use with a continuous forms printer, the PRESENT and DIRECTION subcommands might need to be specified in the form definition. See “Specifying Page Presentation on Continuous-Forms Printers” on page 29 for more information.

Printing Line Data on a Print Server Printer

This example shows how you can print a data file that is developed for a line printer on a page printer without altering the data. The example compares the effects of line printer controls with the corresponding controls in the PPFA commands and subcommands. PRINTLINE, LINESP, POSITION, CHANNEL, and REPEAT are page definition controls related to the lines of text in your printout. Line printer controls examined are the forms control buffer (FCB) and carriage control characters.

As shown in Figure 22, a file that consists of 13 records is to be printed. Several different printouts of this data are formatted in the following examples. In the first two printouts, records 1 - 6 are printed on page 1, records 7 - 9 on page 2, and records 10 - 13 on page 3.

Figure 22. Line-Data File

Figure 23 on page 40 shows the formatting process that is used when the file is printed on a line printer. For many line printers, an FCB is used to format the output in the z/OS, VM, and VSE environment. The sample FCB represented in Figure 23 on page 40 determines that no printed page contains more than eight lines. A page can have exactly eight lines without using carriage control characters in the data. A page can contain any number of lines fewer than eight by placing fewer than eight records between the carriage control characters in the data. In the data file in Figure 22, fewer than eight records are, in all cases, placed between channel 1 carriage control characters. A ninth record, if encountered before a carriage control character, would cause a page eject and a return to the beginning of the FCB. The printout that is shown in Figure 23 on page 40 results from an FCB that formats the data.
A page definition can work the same way. Consider the following example:

```
SETUNITS 1 IN 1 IN
LINESP 6 LPI;
PAGEDEF ABCD
WIDTH 5
HEIGHT 7
LINEONE .5 .5 ;
PRINTLINE CHANNEL 1
  POSITION MARGIN TOP
  REPEAT 8 ;
```

This command stream contains one new command (PRINTLINE) and four new subcommands (LINESP, CHANNEL, POSITION, and REPEAT) related to controlling individual lines.

- The subcommand has the same function as the LPI specifications in the FCB or in a Printer File; it defines the line density iLINESPn lines per inch.
- The PRINTLINE command contains the controls for one or more lines.
- The CHANNEL subcommand has the same function as the channel 1 control character in the FCB, causing a page eject at each channel 1 control character that is encountered in the data records.
- The POSITION subcommand establishes the location of the first line relative to the upper-left corner of the logical page. This example uses the MARGIN and TOP parameters; however, numeric parameters similar to the ones that are used with the OFFSET subcommand can also be used. Those values are also relative to the logical page.
- The REPEAT subcommand is a commonly used control in PPFA text formatting. It is the way that you specify the total number of PRINTLINEs in a logical page.

**Note:** The constraints in specifying a REPEAT value and thus, the number of lines per page, are: the lines per inch specification, the height of the logical page, and the font selection. The REPEAT variable is chosen to equal the maximum number of records to be printed per page. As in the line printer version, if a ninth record were encountered before a channel 1 carriage control character, a page eject would occur and the line would be printed as the first line at the top of the next page.

The result of this page definition is represented in Figure 24.

**Printout**

![Printout Example Specifying POSITION MARGIN TOP](image)

Figure 24. Printout Examples Specifying POSITION MARGIN TOP

Changing line printing specifications for the following example is shown in Figure 25.

**Printout**

![Printout Example Specifying POSITION MARGIN 4.1](image)

Figure 25. Printout Example Specifying POSITION MARGIN 4.1

```plaintext
SETUNITS 1 IN 1 IN
LINESP 6 LPI;
PAGEDEF ABCD
WIDTH 5
```
Observe that the second parameter of POSITION is no longer TOP; instead, it is 4.1, which places the first line of text 4.1 inches down the page rather than at the top (Figure 25 on page 41).

The following example and Figure 26 show a third version of the possible formats for the data that is represented in Figure 23 on page 40.

You also can skip over space by using carriage control characters. This example shows how to use a second PRINTLINE command to create a second starting position on the page (as shown in Figure 26). The second starting position is vertically 4.1 inches down from the top of the page; see the second POSITION subcommand. The two CHANNEL 1 subcommands take turns mapping the records that are governed by the successive channel 1 carriage control characters in the data to their specified positions on the page. In this case, the carriage control 1 characters cause printing to alternate between the TOP position (0.1 inches down the page) and 4.1 inches down the page.
Processing Fields

This section describes the mapping of individual fields to the printed sheets. You can use the technique to print unformatted data according to precise specifications, and these specifications can change without affecting the data file.

The rules for field processing of data files are:

- Each record in your file must correspond to a separate PRINTLINE command because each record is mapped separately. When you are processing identical fields, you can define a single printline and use the REPEAT subcommand.
- Each FIELD command must follow its associated PRINTLINE command, and more than one FIELD command can be specified for a single PRINTLINE command.

For this field-processing example, the data file shown in Figure 28 is used. Figure 29 on page 44 represents an output format that can be used to place data on a form, such as an invoice or an order. The page definition commands to print Figure 29 on page 44 are in Figure 27.

```
PAGedef ABCD
   Width 7 in
   Height 8 in;
   Printline position 1 in 1 in ; /* Processing for R1 */
   Field start 1 length 4 ; /* The printline position is */
   /* The default for the first field */
   Field start 11 length 4
   Position 4 in 0 in ;
   Printline position 3 in 4 in ; /* Processing for R2 */
   Field start 1 length 4 ; /* Default position */
   Field start 6 length 4
   Position 0 in 1 in ;
   Field start 13 length 3
   Position 2 in 3 in ;
   Printline position 1 in 2 in ; /* Processing for R3 */
   Field start 1 length 4 ; /* Default position */
   Field start 11 length 4
   Position 4 in 0 in ;
```

Figure 27. Field-processing example for the PRINTLINE command

```
1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
R1 A A A A B B B B
R2 X X X X Y Y Y Y Z Z Z Z
R3 1 1 1 1 2 2 2 2 2 2 2
```

Data File

Figure 28. Unformatted Print Data File
POSITION Subcommand in Processing Fields Example

The POSITION subcommand of each PRINTLINE command in Figure 27 on page 43 specifies the printline position relative to the logical page origin. The POSITION subcommands below FIELD commands specify a field position relative to the governing printline position. Following POSITION subcommands come the horizontal (x) then the vertical (y) offsets from the reference point. They are parallel in structure to the OFFSET subcommand of the form definition.

For example, the final POSITION subcommand places the final field 1 + 4 inches to the right of the left edge of the logical page, combining the x value of 1 in the PRINTLINE command, and the x value of 4 in the nested FIELD command. The 0 in the FIELD command specifies no change to the y value in the PRINTLINE command. Thus, the position of the final field is 5 IN (x), 2 IN (y).

Note: The first FIELD command within each PRINTLINE has no position specification because the PRINTLINE POSITION value is the default for the first FIELD command nested under it.

Alternative controls for the x and y values of a POSITION subcommand are available. See the description of the POSITION subcommand in “FIELD Command” on page 319 and “PRINTLINE Command” on page 423.

FIELD Command in Processing Fields Example

In the FIELD command in Figure 27 on page 43, the START and LENGTH parameters specify the location of the field in the record to be processed. START indicates the starting byte position, and LENGTH specifies the number of bytes in the field.

Because a field can be located independently within the data and on the printed page, more than one page definition or page format can be created for the same data file, each specifying different mapping of the data to the output pages.
Varying Fonts on a Page

This example illustrates a simple font variation within a printout. The task is to print a line-data file that has the first line of each page in bold-faced type and the rest in standard type. This task requires controls for two fonts in the page definition.

The commands to select a single font for the page, as shown in Figure 30, are as follows:

The FONT command contains two names: the local (STANDARD) name and the user-access (M101) name for the selected font.

| PAGEDEF ABCD ;   |
| FONT STANDARD M101 ; |
| PRINTLINE ; |

**Note:** Fonts cannot be an FGID. Also, all page definitions require a PRINTLINE command.

The next command stream changes the font by incorporating a TRCREF command. Assume the data file to be formatted incorporates table reference characters (TRCs) as shown in Figure 31 on page 46.

The commands to define local fonts are as follows:

| FONT BOLDFACE M102 ; |
| PAGEFORMAT ABCD ;   |
| TRCREF 0 ;           |

Figure 30. Data File Printed Using a Single Font

Figure 31. Data File Printed Using Multiple Fonts
The TRCs in the data cause the font switch to be made. The TRCREF command equates a TRC in the data file with the local name of a font that is specified in the FONT command. The FONT command also contains the user-access name for the font. See Table 8 on page 202 for information on local names and user-access names. Because of the relationship among the user-access name, the local name, and the TRC number that is established in the page definition, the TRCs in the data can cause a font switch automatically.

You can specify fonts within a PRINTLINE command when the data file contains no TRCs. For example, assume the data file represented in the sample print in Figure 32 on page 47 is to be formatted by this page definition:

```
PAGEDEF ABCD;
  FONT M101;
  FONT BOLDFACE M102;
  PRINTLINE CHANNEL 1
    POSITION MARGIN TOP
    FONT BOLDFACE;
  PRINTLINE POSITION MARGIN NEXT /*STANDARD-TYPE LINE */
    FONT M101
    REPEAT 7;
```

---

**Figure 31. Font Change Using TRCREF Command**

![Diagram showing font change using TRCREF command](image-url)
This command stream, based on a data file without TRCs, works on the principle that each line of output whose font you want to change from the font in the previous line must be controlled by a separate PRINTLINE command. The FONT subcommand of the PRINTLINE command names the font that is desired for that line. In this example, two PRINTLINE commands are used because one font change and two fonts are intended for the output. The user-access font names appear in the two FONT commands immediately below the PAGEDEF command and optionally, a local name. M101 and M102 in the example are user-access names; BOLDFACE is a local name. Use the local name in the FONT subcommand of PRINTLINE if it is included in the corresponding FONT command, as is done for the first PRINTLINE command.

Changing fonts field by field is similar to changing them in PRINTLINEs. You map each field individually with a FIELD command; include a FONT subcommand in the FIELD command. If a font change is wanted for a field, as with the FONT subcommand of a PRINTLINE command, the font must be previously named in a FONT command.

Two possible defaults apply in case you do not specify a font within a field. If the governing PRINTLINE has a FONT subcommand, it contains the font default for the field. If the governing PRINTLINE has no font specification, the print server assigns a font according to its default rules.
Printing Lines in Two Directions on a Page

Lines can be printed in any of four directions, depending on the type of printer that is used.

The four parameters for line direction are ACROSS, DOWN, BACK, and UP. The PPFA commands that are used to format a line-data file with lines that are printed in more than one direction (as shown in Figure 33) are stated in the following page definition:

```
PAGedef ATOG
  DIRECTION ACROSS;
  PRINTLINE POSITION 1 IN 1 IN /*LINES A-E*/ REPEAT 5;
  PRINTLINE POSITION .5 IN 6 IN /*LINE F*/
          DIRECTION UP;
  PRINTLINE POSITION 1 IN 6 IN /*LINE G*/
```

Figure 33. A Printout with More Than One Line Direction

In this page definition, the logical page direction ACROSS is specified. ACROSS is the default, but its inclusion clarifies that no direction control is needed for lines A–E. The default direction of a prineline is the direction specification of the logical page of which it is part. The PRINTLINE command for the record F has a DIRECTION subcommand because the direction specification changes from the specification of the previous line. Record G is to be printed in the ACROSS direction again. However, a direction is not specified because the ACROSS direction is the default for all lines in this page definition.

Note: If you are building the page definition for use with the 3800 printer, and if the input data contains table reference characters, you can use the DIRECTION subcommand of the TRCREF command to specify a font that prints UP on the page, as in line F. For more information, see “TRCREF Command (Traditional)” on page 445.

Printing Fields in Two Directions on the Same Page

This example is similar to “Printing Lines in Two Directions on a Page,” except that you learn how to control direction field by field. This method creates a field-processing page definition and places direction controls in the FIELD commands. This command stream contains a portion of the page definition controls, showing only the PRINTLINE commands:

```
PRINTLINE POSITION MARGIN TOP;
  FIELD START 1 LENGTH 4;
PRINTLINE POSITION 2 IN 4 IN;
  FIELD START 7 LENGTH 4
          DIRECTION UP;
```
As expected in field processing, FIELD commands are nested within PRINTLINE commands. Figure 34 shows a simplified portion of an unformatted file and two pages of the printout that is formatted by the page definition, part of which is shown in the command stream. Two PRINTLINEs are specified because, as Figure 34 shows, the data file contains two input record formats (1 and 3 are alike; 2 and 4 are alike) and because the fields are mapped to two different positions in the output. The assumption of this sample is that the data file is much longer than the portion shown. However, if the records in the file alternate in format as the first four do, the two PRINTLINEs of this page definition format as many records as are presented, two to a page, on pages 1 - n.

If more than two mappings are required by the print job, more than two PRINTLINE commands are required in the page definition.

Fonts rotate relative to the inline direction of lines (or fields).

This example focuses on a single letter A from FONTA. With PPFA, a single font that is specified in a page definition can produce letters in any of four rotations. This rotation is accomplished by a FONT command that specifies rotation. In this example, if you want to vary the rotation of a font twice within a page, you use two FONT commands, one for each rotation. You also use two PRINTLINE commands to map the data to the printout by using the two rotations of the font. In a field processing application, FIELD commands can be used in the same way. These PRINTLINE commands name the rotated font in a FONT subcommand.

Figure 35 on page 50 breaks down the elements that are required for the FONT commands and subcommands. Distinct local names and rotation specifications for each font are placed in a FONT command. These specifications identify a font as rotated within a page definition. The rotation of a character is relative to the inline direction of a printline or field. The characters and rotations that are shown here assume an inline direction of ACROSS. See "PPFA Basic Terms" on page 8.
You can use up to 16 possible combinations of logical page direction and font rotation for page printers other than the 3800.

The FONT subcommands within PRINTLINE or FIELD commands that name the rotated font in that page definition use only the local name. The following command stream shows the proper specification and nesting of FONT commands and subcommands for rotation.

```
PAGEDEF ABCD ;
  FONT FontA M103 ; /*NO ROTATION, LOCAL AND */
  /USER-ACCESS NAMES.*/
  FONT FontArtD180 M103 /*ROTATED FONT, LOCAL, USER-ACCESS*/
  /* NAMES PLUS ROTATION SUBCOMMAND */
  /* AND PARAMETER. */
  PRINTLINE FONT FontA REPEAT 3 ; /*LOCAL NAME */
  PRINTLINE FONT FontArtD180 REPEAT 2 ; /*LOCAL NAME */
```

![Figure 35. Character Rotation](image)

You can use up to 16 possible combinations of logical page direction and font rotation for page printers other than the 3800.

The FONT subcommands within PRINTLINE or FIELD commands that name the rotated font in that page definition use only the local name. The following command stream shows the proper specification and nesting of FONT commands and subcommands for rotation.

```
PAGEDEF ABCD ;
  FONT FontA M103 ; /*NO ROTATION, LOCAL AND */
  /USER-ACCESS NAMES. */
  FONT FontArtD180 M103 /*ROTATED FONT, LOCAL, USER-ACCESS*/
  /* NAMES PLUS ROTATION SUBCOMMAND */
  /* AND PARAMETER. */
  PRINTLINE FONT FontA REPEAT 3 ; /*LOCAL NAME */
  PRINTLINE FONT FontArtD180 REPEAT 2 ; /*LOCAL NAME */
```

![Figure 36. Example of Assumed Data File and Rotation Specifications](image)
FONTA, identified in the first FONT command, requires no rotation parameter because it is printed in the default position (or 0° rotation) for font M103. For the rotated font, the second FONT command identifies FONTARTD180 (the local name) as M103 rotated 180°.

### Using Traditional Kanji Formatting

Traditional kanji print presentation, called *tate*, is possible with a printer by using a combination of font rotation and logical page direction. A logical page in the DOWN direction and a 270° font rotation provide the right combination to present kanji in tate format on the printer.

```plaintext
FORMDEF TATE
  OFFSET 1 IN 1 IN ;
PAGEDEF TATE
  HEIGHT 5 IN
  WIDTH 6 IN
  DIRECTION DOWN ;
  FONT KANJI TND M104
  ROTATION 270 ;
  PRINTLINE FONT KANJI TND
              REPEAT 3 ;
```

![Figure 37. AFP Printer Tate Presentation](image)

Figure 37 shows the result of formatting with the page definition in the example. The characters are added to lines down the page. Lines are added right to left.

### Printing Multiple-Up Pages

*Multiple up* is a printer’s term for printing two or more pages of data on one side of a sheet, which is possible with your print server printers and PPFA formatting. The steps that are used in this example are:

1. Change the print direction of the logical page to one of the landscape presentations.
2. Conceptually divide the sheet of paper into parts, one for each multiple-up page (subpage).
3. Create a PRINTLINE position at the top of each multiple-up page.

This example assumes the existence of a line-data file with carriage control 1 characters after records 4, 7, and 11. Each carriage control 1 character begins a new page. Because four pages really exist on the sheet, a skip-to-channel 1 must be used four times. The fifth channel 1 character causes a page eject and the beginning of a new physical sheet. The PPFA commands that follow are for one
version of a multiple-up page. This set of commands creates a page layout like the one shown in Figure 38 (the physical sheet is not shown).

```
FORMDEF MULTUP
  OFFSET 1 IN .5 IN ;
SETUNITS LINESP 4 LPI ;
PAGEDEF MULTUP1
  WIDTH 10 IN
  HEIGHT 8 IN
  DIRECTION DOWN /* FOR LANDSCAPE PRESENTATION */
  PRINTLINE CHANNEL 1 /* PAGE 1 */
    POSITION 1 IN 1.5 IN
    REPEAT 6 ;
ENDSUBPAGE ;
PRINTLINE CHANNEL 1 /* PAGE 2 */
  POSITION 1 IN 5.5 IN
  REPEAT 6 ;
ENDSUBPAGE ;
PRINTLINE CHANNEL 1 /* PAGE 3 */
  POSITION 6 IN 1.5 IN
  REPEAT 6 ;
ENDSUBPAGE ;
PRINTLINE CHANNEL 1 /* PAGE 4 */
  POSITION 6 IN 5.5 IN
  REPEAT 6 ;
```

The DOWN PRINTLINE direction creates a page with a landscape presentation typical of multiple-up printing. Individual PRINTLINEs are specified for the initial lines of the four pages. Ensure that the lines of each page fit in the space that is designated by the use of a small font.

**Note:** In this example, no font is specified for the page definition; therefore, the default font for the page printer is used. If you want a different font, write a FONT command that names it.

The next set of commands alters the sequence of pages.
FORMDEF MULTUP
OFFSET 1 IN .5 IN ;
SETUNITS LINESP 4 LPI ;
PAGEDEF MULTUP2
WIDTH 10 IN
HEIGHT 8 IN
DIRECTION DOWN ;
PRINTLINE CHANNEL 1 /* PAGE 1 */
  POSITION 1 IN 1.5 IN
  REPEAT 4 ;
  ENDSUBPAGE ;
PRINTLINE CHANNEL 1 /* PAGE 2 */
  POSITION 6 IN 1.5 IN
  REPEAT 4 ;
  ENDSUBPAGE ;
PRINTLINE CHANNEL 1 /* PAGE 3 */
  POSITION 1 IN 5.5 IN
  REPEAT 4 ;
  ENDSUBPAGE ;
PRINTLINE CHANNEL 1 /* PAGE 4 */
  POSITION 6 IN 5.5 IN
  REPEAT 4 ;

Here, the upper-right and lower-left pages are reversed by reversing the position controls for the second and third printlines.

Figure 39 shows the changed printout that results from the page definition command changes. When you set up your basic page definition, changes such as the ones described here become easy.

![Figure 39. Multiple-Up Page Layout after Page Definition Modification](image)

**Note:** The ENDSUBPAGE command can be used to mark the boundaries between subpages. Without it, the page definition is no different from any other sequence of PRINTLINEs with POSITION commands. Boundaries do not have to be marked unless conditional processing is being done. The examples that are given here print identically with and without ENDSUBPAGE commands. (See “Subpage Description and Processing” on page 121 for more information.)
Chapter 4. Using Page Definition Commands for Record Format Line Data and XML Data

This chapter describes the record formatting function and the XML page definition formatting function.

Record Formatting Function

The record formatting function allows an application to specify a format identifier (Record ID) with each set of output data fields (Data Record). The format identifier references a specific layout format in a page definition (PAGEDEF). At print time, each layout format (referenced by a Record ID in a Data Record) is retrieved from the PAGEDEF and used to position and format the associated Data Records/fields on the output page.

The purpose of the record formatting capabilities is to move more of the output formatting function into the PAGEDEF and allow for greater flexibility in creating and changing output pages without changing the base application. Rather than the application generating page headers, page trailers, and group headers for each page (and thus fixing the page endings), the page headers, page trailers, and group headers can be generated by a PAGEDEF layout, which allows the page endings to change as font sizes or data layouts change.

To visualize how the record formatting function can be used, review the first six pages of “Record Formatting Examples” on page 77. These examples show the output of an application before and after it is formatted with PAGEDEF by using the record formatting functions.

These functions are provided by several new PPFA commands (LAYOUT, DEFINE COLOR, DRAWGRAPHIC, and ENDGRAPHIC), and modifications to the PAGEDEF, PAGEFORMAT, FONT, CONDITION, and FIELD commands. This chapter provides an explanation of the record formatting functions with examples of their use. For details on the syntax of these commands, see Chapter 11, “Page Definition Command Reference,” on page 271.

Some of the functions that can be accomplished in a layout format with the record formatting commands include:

- Selecting different formatting for different types of Data Records/fields based on the Record ID. The output formatting can change mid-page independent of where the output occurs on a page.
- Defining page headers and trailers to be automatically printed on subsequent pages. The headers and trailers can incorporate data from the associated Data Record.
- Numbering the output pages.
- Inserting page ejects can be automatic when text reaches the bottom margin.
- Creating group headings to be printed at the beginning of a group of data. For example, you can create group headings (including column headings) to be repeated each time a different account type is formatted on a banking statement. An active group heading is automatically repeated on subsequent pages until the data group ends.
- Forcing page ejects to occur in the output.
• Creating boxes with or without black and white or color shading. A set of boxes for a table can be started in a group header and automatically ended and restarted on subsequent pages until the table completes.
• Creating graphical objects, such as circles, ellipses, lines, and graphs, in color or black and white output.
• Formatting database records created with field delimiters (rather than fixed-length fields).
• Aligning field output to the left or right.

Record Format Page Definition

A record format page definition specifies how you want data to be positioned on the logical page.

A record format page definition is a resource that is used by the print server that defines the rules of transforming line data into composed pages and text controls for printing. With record format page definitions, you can do the tasks that are listed in Table 4.

Table 4. Record Format Page Definition Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Location of an Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a page definition.</td>
<td>“Page Definition Command Nesting” on page 57</td>
</tr>
<tr>
<td>Assign a record ID.</td>
<td>“Record ID Data Format” on page 57</td>
</tr>
<tr>
<td>Use the LAYOUT command.</td>
<td>“LAYOUT Command” on page 58</td>
</tr>
<tr>
<td>Use the FIELD command.</td>
<td>“FIELD Command” on page 60</td>
</tr>
<tr>
<td>Define logical page size.</td>
<td>“Defining Logical Page Size” on page 64</td>
</tr>
<tr>
<td>Position data on a logical page.</td>
<td>“Positioning the Data” on page 65</td>
</tr>
<tr>
<td>Change the print direction.</td>
<td>“Changing Logical Page Print Direction” on page 65</td>
</tr>
<tr>
<td>Process fields.</td>
<td>“Processing Fields” on page 69</td>
</tr>
<tr>
<td>Print in different directions.</td>
<td>“Printing Lines in Two Directions on a Page” on page 72</td>
</tr>
<tr>
<td>Print fields in two directions.</td>
<td>“Printing Fields in Two Directions on the Same Page” on page 72</td>
</tr>
<tr>
<td>Change fonts.</td>
<td>“Varying Fonts on a Page” on page 73</td>
</tr>
<tr>
<td>Rotate fonts.</td>
<td>“Rotating Fonts” on page 75</td>
</tr>
<tr>
<td>Print kanji.</td>
<td>“Using Traditional Kanji Formatting” on page 77</td>
</tr>
<tr>
<td>Use record formatting and commands.</td>
<td>“Record Formatting Examples” on page 77</td>
</tr>
</tbody>
</table>

Page Formats within Page Definitions

Just as form definitions can include more than one copy group, page definitions can include several page formats. Page formats use basically the same subcommands as page definitions, and if a subcommand is specified in a page format, it overrides the value that is specified in the page definition for the page format. A single page definition can contain multiple page formats. If pages in a file are to be formatted differently, specify more than one page format in your page definition. Within a page definition, page formats are generated in the order in which they are specified.
Using more than one page format to control different pages requires one of these:

- Adding the Invoke Data Map structured field to the data file each time you want to change page formats.
- Using conditional processing.

See Advanced Function Presentation: Programming Guide and Line Data Reference for more information about the Invoke Data Map structured field.

**Page Definition Command Nesting**

The following simplified command stream shows the proper nesting of commands and the order in which they must be entered when you create a page definition:

```
[SETUNITS]
PAGEDEF
FONT
[OBJECT]
[DEFINE COLOR]
[PAGEFORMAT]
 [SEGMENT]
[OVERLAY]
[LAYOUT]
 [CONDITION]
 [FIELD]
 [DRAWGRAPHIC]
 [ENDGRAPHIC]
[PAGEFORMAT]
 [SEGMENT]
[OVERLAY]
[LAYOUT]
 [CONDITION]
 [FIELD]
 [DRAWGRAPHIC]
 [ENDGRAPHIC]
```

**Notes:**

1. Brackets enclosing a command mean that the command is optional.
2. Indentations are used to improve readability.
3. Complete definitions of all commands are included in Chapter 11, “Page Definition Command Reference,” on page 271.

**Rules:**

1. Record format LAYOUT commands and traditional PRINTLINE commands cannot be used within the same PAGEDEF. At least one LAYOUT command is required per page format for a record formatting page definition.
2. A SETUNITS command can be placed before any other PPFA command. The values that are set are in effect until the next SETUNITS command.
3. SEGMENT and OVERLAY commands must be specified under their associated PAGEFORMAT command.
4. The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.
5. One file can contain multiple sets of page definitions.

**Record ID Data Format**

In order to allow different formats for different groups (or tables) of data, each of which have an unpredictable number of entries, a Record ID is assigned to each
output record to identify the type of record and control layout formatting. An
application can group data fields that are to be formatted together as an entity into
Data Records with a specific Record ID. For example, in a bank statement, the data
fields for a check transaction might be grouped with a Record ID identifying that
record as a check transaction. The PAGDEFF would then define a special layout
format for a check transaction with a matching Record ID (see “Record Formatting
Examples” on page 77 for detailed examples).

Record formatting in PPFA is achieved by identifying each input record in the data
file with a 10-byte ID, similar to an expanded carriage control (CC) (see “Basic
Controls in Record Format Line Data” on page 13 for additional information). Each
record in the data file must contain a Record ID if record formatting is used. The
Record ID must be the first 10 bytes in every print record in the data file.

Even though the Record ID is specified as a character string, the Record ID is
treated as a hexadecimal string, not a character string. This means that there is no
translation from ASCII to EBCDIC or vice versa when the Record ID is processed.
The Record ID in the input data must match exactly the string that is specified for
the LAYOUT Record ID in the page definition in order for correct processing to
occur.

When a record is read from the data file at print time, the print server uses the
10-byte Record ID to determine which LAYOUT command in the page definition
must be used to format the record.

TRCs (Table Reference Characters) cannot be used with record format data. If you
have TRCs in the data and tell the print server that TRCs are present at print time,
the print server uses the TRC byte as the first byte of the Record ID, and the
Record ID is not recognized as such.

Data files can contain both carriage controls and Record IDs. If your data file is
mixed mode (line data plus MO:DCA structured fields), then you must have a CC
byte in the data. The CC byte is not counted as part of the 10-byte Record ID. If
your file is plain line data, then CCS are allowed but not required. (See “Basic
Controls in Record Format Line Data” on page 13 for additional information.)

**LAYOUT Command**

When record formatting, the LAYOUT command is used instead of traditional
PRINTLINE commands in the page definition. You cannot mix record format
LAYOUT and traditional PRINTLINE commands in the page definition. With
LAYOUT (see “LAYOUT Command (Record Format)” on page 365), you can
identify four types of Data Records:

- Body Records
- Page Headers
- Page Trailers
- Group Headers

Each of the record types is discussed in the following sections. No matter which
types of record you are formatting, you can control the positioning, font, color, and
direction for the print record.

The POSITION keyword on the LAYOUT command is used to set the initial print
position for subsequent text and graphics that are placed with the FIELD and
DRAWGRAPHIC commands.
• The horizontal position can be specified as LEFTMARGIN, at the same position as
  the previous layout, or at an absolute or relative location that is given in inches,
millimeters, centimeters, points, or pels (see “PAGEDEF Command” on page 404).
• The vertical position can be specified as TOPMARGIN, at the same position as the
  previous layout, at the next vertical position (with the current LINESP value), or
at an absolute or relative location that is given in inches, millimeters,
centimeters, points, or pels (see “PAGEDEF Command” on page 404).

Body Records
The BODY layout type is used for most of the data in the user’s input file. That is,
any record that is not used for special processing as a page header, page trailer, or
 group header, contains data to be formatted and placed on the page.

Body records are positioned initially with the LAYOUT command. The default x
(horizontal) position for each body record is to be at the same horizontal position
as the previous LAYOUT. If it is the first LAYOUT on a logical page, the default
horizontal position is 0.

The default y (vertical) position is to place the layout record down one line (as
defined in the LINESP subcommand of the last SETUNITS command) from the
previous field. If it is the first LAYOUT on a logical page, the default vertical
position is one line down from the top margin of the logical page. See “PAGEDEF
Command” on page 404 for details about margins.

You can specify the rotation of data with the DIRECTION keyword on LAYOUT.
All of the fields that are defined for this record layout use the same direction
unless it is overridden on the FIELD command. On relative LAYOUTs and their
fields, the rotation must be ACROSS so that they have the same net rotation as the
page format.

You can also specify fonts and color to be used for the text that is formatted with
this layout record. Additionally, double-byte fonts can be requested if you have
double-byte characters in your data. The color of the text and graphic borders is
specified with the COLOR keyword. See “DEFINE COLOR Command” on page 280 and
“FONT Command” on page 360 for details.

Page segments, overlays, and objects can be included with keywords on the
LAYOUT command. This processing is the same as the traditional PRINTLINE
command.

Body records can also be identified as belonging to a group. When the GROUP
keyword is used on the body LAYOUT, the group header that is in effect at the
time is repeated on subsequent pages while the input records use Record ID’s that
select body LAYOUT and use the GROUP keyword. The group is ended as soon as
a Record ID in the input selects a LAYOUT that does not use the GROUP
keyword.

Page Headers and Trailers
Page headers and trailers are printed automatically on each new page. Default
headers and trailers can be created, which are automatically invoked on each new
page without requiring or allowing any input data. No input record data is
allowed in a default header or trailer because they are triggered automatically by
page ejects and are not associated with any records in the input data file. See
“LAYOUT Command (Record Format)” on page 365 for more details.
Rather than using the defaults, you can create headers and trailers that are invoked by a Data Record containing the header or trailer Record ID. These headers and trailers can use input record data in their layout; however, it is not required.

The following example creates a page header and trailer. See “PAGEDEF Command” on page 404 for more details.

```
LAYOUT C'statmid'
   SEGMENT ibmlink 1.15 in 1.35 in
   PAGEHEADER NEWPAGE
   POSITION SAME ABSOLUTE NEXT;

LAYOUT C'pgenum' PAGETRAILER
   POSITION SAME ABSOLUTE 10.7 in;
```

Figure 40. Sample Page Header and Trailer

**Group Headers**

A Group Header layout consists of text, graphics, and other data that is to be printed at the beginning of a group of user records. For example, if you are creating a banking statement, you might define a Group Header for checking and one for savings.

The group header is defined with a special LAYOUT GRPHEADER command, and stays in effect until a BODY layout is encountered that specifies NOGROUP. See “LAYOUT Command (Record Format)” on page 365 for more details on the GRPHEADER subcommand.

If a logical page eject occurs before the group is ended, the header is printed after the top margin on each new page until the group ends.

**FIELD Command**

The FIELD command is used to identify a field in a Data Record to be formatted and placed on the page. FIELD must follow the LAYOUT command, and parameters that are not specified on FIELD are inherited from the previous LAYOUT. This section describes the new keywords on FIELD that are used with record formatting.

Page numbering can be accomplished by specifying FIELD with the PAGENUM parameter. Most often, you specify FIELD PAGENUM with other formatting information such as position and alignment, which causes the current page number to print at the specified position. The current page number is calculated based on the specification of the PAGECOUNT parameter on the previous PAGEDEF or PAGEFORMAT command. You can override the page number to a specific value with the RESET parameter on the FIELD command. For details, see “Page Numbering” on page 62.

You can retrieve the value of the Record ID for printing with the RECID keyword on FIELD. RECID also has START and LENGTH subparameters to allow only portions of the Record ID to be printed. Normally, you use the RECID parameter only for debugging your application by tracing which Record IDs are being processed, although it can be used for anything that makes sense for your application.

You can also specify the POSITION, COLOR, DIRECTION, and ALIGN keywords with the PAGENUM or RECID parameters on FIELD. The BARCODE and
SUPPRESSION keywords are not allowed with PAGENUM or RECID, but can be used with other text fields from the Data Record.

ALIGN is a keyword that is allowed with the START/LENGTH or TEXT forms of the FIELD command, but only if you are doing record formatting. You use ALIGN to specify whether the field text is LEFT or RIGHT aligned at the given horizontal position.

If your Data Records are stored in a database, the fields might be separated with field delimiters instead of just being positional within the record. The DELIMITER keyword on the preceding LAYOUT command is used to specify the 1-byte or 2-byte value that is used to separate fields in the Data Records.

If your data uses field delimiters, you can also specify the FLDNUM parameter on the FIELD command to indicate the number of the field within the record to be extracted, rather than the START position. Fields are numbered from left to right beginning with 1. You can also use the starting position (START) and LENGTH keywords with the FLDNUM to indicate that only part of the field is to be formatted. An example of a typical command is:

```
COMMANDS

LAYOUT 'abc' DELIMITER '*';
FIELD FLDNUM 1 START 2 LENGTH 8 ALIGN RIGHT
  POSITION 5.6 \n CURRENT
  FONT varb; /* Variable text - Amount */
FIELD FLDNUM 2 ALIGN LEFT
  POSITION 1.1 in .9 in
  FONT varb; /* variable - customer name */

DATA
  abc     *Here is some data*more data*

FIELDS used
  1st field 'ere is s'
  2nd field 'more data'
```

Figure 41. Sample Commands and Data With Delimiters.

**Controlling Page Formatting**

You use the parameters on the PAGEDEF and PAGEFORMAT commands to specify the margins of the page. The TOPMARGIN and BOTMARGIN keywords are used to reserve space at the top and bottom of the page. The page headers and trailers are normally placed into this reserved space.

**Note:** No other text or objects need to be written into the margins - only page header and trailer data.

The bottom margin is also used for two other purposes:
- A BODY or GRPHEADER Data Record, which causes the baseline position to move into the bottom margin area, causes a logical page eject.
- Any graphic that is started with the DRAWGRAPHIC command, but not explicitly ended, automatically ends at print time before it extends into the bottom margin area.
You can force a new logical page in the output with the NEWPAGE keyword on a
LAYOUT command (see “LAYOUT Command (Record Format)” on page 365).
When an input record is encountered whose Record ID matches that LAYOUT
name, a page eject is completed before the record data is processed. If it is a
header or trailer layout, the page eject is done before the header or trailer becomes
active.

The ENDS(PE)ACE keyword can also be used to control where page ejects are done.
If ENDS(PE)ACE is coded on a LAYOUT, and a Data Record with the matching
Record ID is encountered, a page eject is done before the data is processed - if the
remaining space on the page (before the bottom margin) is less than the
ENDSPACE value.

The ENDS(PE)ACE keyword can be used to ensure that a Table Heading (Group
Heading) does not print at the end of a page without allowing space for more Data
Records (body records), or to ensure that a table entry does not print at the bottom
of a page without allowing space for a totals record.

The following example shows the use of page margins and the NEWPAGE and
ENDSPACE keywords:

```
PAGEFORMAT chub1 TOPMARGIN 2 in BOTMARGIN 2 in;
/*******************************************************************************/
/** statmid BODY **/ 
/*******************************************************************************/
LAYOUT C\'statmid\'
PAGEHEADER NEWPAGE ENDS(PE)ACE .5 in
POSITION .6 in ABSOLUTE .55 in;
FIELD
TEXT C\'Big Brother Bank\'
ALIGN LEFT
FONT comp ; /* default to LAYOUT positioning*/
```

Figure 42. Sample Page Formatting

Page Numbering
Page numbers can be placed with the PAGENUM keywords on the FIELD
command. You use PAGENUM to specify whether the page number prints, and
whether you want it reset to a specific value rather than using the current value
(page count).

The page number prints as an integer (for example, 1, 2, 3, ...) and has a valid
range of 1 - 4 billion (four unsigned bytes of data). If the specified or defaulted
font that is used for printing the page number is other than an EBCDIC font, you
must specify it using the TYPE subcommand on the FONT command.

The page number prints by using the font that is specified on the FIELD
command. You can also select a POSITION, COLOR, and DIRECTION for the page
number by using existing FIELD keywords.

The ALIGN parameter on FIELD can also be used to specify whether you want the
page number LEFT or RIGHT aligned at the given position.

The PAGECOUNT keyword is allowed with the PAGEDEF and PAGEFORMAT
commands so that you can specify how page numbering is to be handled when
you are switching between page formats. Page numbering can be stopped, reset,
resumed for a certain point or continued from a certain point. For a detailed
description on how to specify these options, see “PAGEDEF Command” on page
404.
**Graphical Objects**

When you are creating output with record formatting, you can use the DRAWGRAPHIC commands to create boxes, lines, circles, and ellipses relative to the data printed with the LAYOUT command. DRAWGRAPHIC can be used with DEFINE COLOR to shade an object with a percentage of black or other colors; however, DRAWGRAPHIC is not allowed if you are formatting with the traditional PRINTLINE.

**Conditional Processing Considerations**

Conditional processing works much the same in record formatting as when you are using the traditional PRINTLINE processing. The only difference is the ability to process based on a field that is defined by delimiters instead of just a fixed start position and length.

**Logical Page Eject Processing**

A logical page eject can be caused by these:

- Any Record ID that references a layout format with a specification of New Page.
- A relative baseline overflow, which is a Body or Group Header layout format that, when processed against the current input record, causes an overflow of the current print position into the bottom margin. If processing of the input record causes a relative baseline overflow, the page eject is processed before any part of the input record is printed.
- A Data Map change or Medium Map change, or, in Mixed-Mode, a Begin Document or Begin Page structured field.

Page Header, Page Trailer, and Group Header Data Records that are used with page ejects are activated in the following manner:

- If a Data Record specifies the Record ID of a PAGEDEF Page Header layout format, that Data Record is not printed on receipt but is saved as the active page header record (for that PAGEFORMAT). It is saved during the job or until a subsequent Data Record specifies a Page Header (for that PAGEFORMAT).
- If a Data Record specifies the Record ID of a PAGEDEF Page Trailer layout format, that Data Record is not printed on receipt but is saved as the active page trailer record (for that PAGEFORMAT). It is saved during the job or until a subsequent Data Record specifies a Page Trailer (for that PAGEFORMAT).
- If a Data Record specifies the Record ID of a PAGEDEF Group Header layout format, that Data Record is not printed on receipt but is saved as the active group header record. The PAGEDEF Group Header is printed when the next Data Record specifies a Body layout with a GROUP specification and on subsequent page ejects. The Group Header and its associated Data Record are kept active until a subsequent Data Record specifies a Body layout with a NOGROUP specification.

When a logical page eject occurs, the following actions are taken in the following order:

- For the current page:
  1. If it is the start of a line data document (no previous page ejects, group header records or body records are processed with this PAGEDEF), current page items 1 - 3 are skipped.
  2. If an active page header record was in effect before this layout format, that record is presented on the current page by using the matching layout. Otherwise, if the active PAGEFORMAT contains a default Page Header layout, that layout is used to present a page header.
3. If an active page trailer record was in effect before this layout format, that record is presented on the current page by using the matching layout. Otherwise, if the active PAGEFORMAT contains a default Page Trailer layout, that layout is used to present a page trailer.

- For the new page:
  1. The current print position is moved to the top of the new page and offset from the top of the new page by the top margin. If the PAGEFORMAT is changed, the new Data Map’s Margin Definition and layouts are used.
  2. If an active group header record exists for this PAGEFORMAT, that record is presented on the new page by using the matching Record layout. The group header is not printed and causes no action until a Body layout with Group Indicator is processed for the page. If the layout specifies relative positioning, the baseline position of the layout is offset from the top of the page by the top margin plus one line.
  3. If the page eject was caused by a Body layout, the input record that causes the page eject is presented on the new page with the layout that is referenced by the record. If the layout specifies relative positioning and is preceded on the page by a group header, the baseline position is relative to the last printed line of the group header. If the layout specifies relative positioning and is not preceded on the page by a group header, the baseline position of the layout is offset from the top of the page by the top margin plus one line.

Note: The actual locations of ‘top of page’ and ‘top margin’ are affected by the text orientation. See “Using Margins in Record Formatting” on page 67 for more information.

**Defining Color Models**

Record formatting provides you with the ability to redefine a color with your own name, and then use that name anytime this color is needed. It works in much the same way as a FONT command where you define the FONT with an internal name and then use that name when you place text on the page.

**Defining Logical Page Size**

“Positioning a Logical Page on a Sheet” on page 21 shows how to establish the origin point of a logical page, relative to the media origin on a sheet of paper, by using the OFFSET subcommand. The following example shows you how to establish the width and height of the logical page relative to this origin point. This example illustrates how the dimensions of a logical page are determined by form definitions and page definitions.

```
SETUNITS 1 IN 1 IN
LINESP 8 LPI;
FORMDEF ABCD OFFSET 0 .5;
PAGEDEF ABCD
  WIDTH 7.5
  HEIGHT 10
  DIRECTION ACROSS;
FONT GS12 GS12;
LAYOUT 'abc'
  FONT GS12
  POSITION 0 TOP;
```

Normally, all parameters consist of a number and a unit of measurement, for example, 6 IN. (See “Units of Measurement” on page 204 for information on units that are available.) Numbers can be specified with up to three decimal places. The LAYOUT command is included because at least one is required for all page
definitions. See “LAYOUT Command (Record Format)” on page 365 for more information.

![Logical Page Dimensions](image)

The OFFSET subcommand (0) (.5) in the sample form definition establishes the corner or origin of the logical page relative to the physical sheet. The WIDTH and HEIGHT subcommands, (7.5) and (10), specify the dimensions of the logical page relative to the logical page origin.

**Note:** Be careful not to define a logical page larger than the physical sheet. PPFA does not check the size of the physical sheet.

### Positioning the Data

“Defining Logical Page Size” on page 64 shows you how to define the size of a logical page. “Changing Logical Page Print Direction” and “Using Margins in Record Formatting” on page 67 show you how to position data inside the logical page.

### Changing Logical Page Print Direction

Logical pages can have four different print directions: ACROSS, DOWN, BACK, and UP. This example shows that all four directions can be specified in relation to one offset specification:

```plaintext
FORMDEF ABCD
  OFFSET (1) (2) ;
PAGEDEF DEFG ;
FONT GS12 GS12;
PAGEFORMAT DEFG1
  WIDTH (3) 
  HEIGHT (4)
  DIRECTION ACROSS ;
LAYOUT 'abc' ;
PAGEFORMAT DEFG2
  WIDTH (3) 
  HEIGHT (4)
  DIRECTION DOWN ;
LAYOUT 'def' ;
PAGEFORMAT DEFG3
  WIDTH (3) 
  HEIGHT (4)
  DIRECTION BACK ;
LAYOUT 'ghi' ;
PAGEFORMAT DEFG4
```

Figure 43. Logical Page Dimensions
Note: The parenthetical numbers represent dimensions. Figure 43 on page 65 shows how these dimensions relate to the logical page.

One page definition is used to simplify the example, yet four logical pages are specified. The PAGEFORMAT commands create subsets of page definitions for each logical page.

Note: The page formats in this example require an Invoke Data Map structured field at the place in the data file where you want to change page formats. The LAYOUT commands are required but are not relevant in the example.

The DIRECTION subcommand with one of its four direction parameters ACROSS, DOWN, UP, or BACK specifies the print direction of the logical page. Figure 44 on page 67 shows the format of each of the logical pages that are specified in the page definition with the direction specification of each. The pages with the ACROSS and BACK directions are in portrait presentation. The pages with the DOWN and UP directions are in landscape presentation.
The media origins and logical page origins do not change with the presentation of the data on the page. The OFFSET subcommand of the form definition need not change. However, the width and height dimensions do change; that is, the WIDTH subcommand always governs the horizontal (inline) dimension as you view the page, and the HEIGHT subcommand always governs the vertical (baseline) dimension whether the page is in portrait or in landscape presentation. Ensure that these specifications do not cause the logical page to cross the edge of the physical page.

However, if the DOWN direction is specified for use with a continuous forms printer, the PRESENT and DIRECTION subcommands might need to be specified in the form definition. See “Specifying Page Presentation on Continuous-Forms Printers” on page 29 for more information.

### Using Margins in Record Formatting

Margins follow the inline direction of the page. For example, if the text orientation is ACROSS, the upper-left diagram in Figure 44 on page 69 shows the left, top, right, and bottom margins. When specified, these margins define a bounding box for the PAGEFORMAT as indicated by the dotted lines.

If the text orientation is changed, the same bounding box applies to the new orientation, but the names of the margins change in the new orientation. For
example, if the new text orientation is DOWN, as shown in the upper-right
diagram of this same figure, the top margin in the new orientation is now defined
on the long side of the page.

**Left Margin**
Specifies the offset of the left margin along the i axis from the left edge of
the page. The left edge of the page is the zero position on the i axis.

**Top Margin**
Specifies the offset of the top margin along the b axis from the top edge of
the page. The top edge of the page is the zero position on the b axis.

**Right Margin**
Specifies the offset of the right margin along the i axis from the right edge
of the page.

**Bottom Margin**
Specifies the offset of the bottom margin along the b axis from the bottom
dge of the page.
Processing Fields

This section describes the mapping of individual fields to the printed sheets. You can use this technique to print unformatted data according to precise specifications, and these specifications can change without affecting the data file.

The rule for field processing of data files is: Each FIELD command must follow its associated LAYOUT command, and more than one FIELD command can be specified for a single LAYOUT command.

Figure 45. Relationship of Margin Definition to Text Orientation
For this field-processing example, the data file shown in Figure 47 is used. Figure 48 represents an output format that might be used to place data on a form, such as an invoice or an order. The page definition commands to print Figure 48 are in Figure 46.

```
PAGEDEF ABCD
  WIDTH 7 IN
  HEIGHT 8 IN ;
  FONT GS12 GS12;
  LAYOUT 'abc' POSITION 1 IN ABSOLUTE 1 IN ; /*PROCESSING FOR R1*/
    FIELD START 1 LENGTH 4 ; /*THE LAYOUT POSITION IS*/
    /*THE DEFAULT FOR THE FIRST FIELD*/
    FIELD START 11 LENGTH 4
    POSITION 4 IN 0 IN ;
  LAYOUT 'def' POSITION 3 IN ABSOLUTE 4 IN ; /*PROCESSING FOR R2*/
    FIELD START 1 LENGTH 4 ; /*DEFAULT POSITION*/
    FIELD START 6 LENGTH 4
    POSITION 0 IN 1 IN ;
    FIELD START 13 LENGTH 3
    POSITION 2 IN 3 IN ;
  LAYOUT 'ghi' POSITION 1 IN ABSOLUTE 2 IN ; /*PROCESSING FOR R3*/
    FIELD START 1 LENGTH 4 ; /*DEFAULT POSITION*/
    FIELD START 11 LENGTH 4
    POSITION 4 IN 0 IN ;
```

Figure 46. Field-processing example for the LAYOUT command

```
R1
A A A A | B B B B
R2
X X X X | Y Y Y Y | Z Z Z Z
R3
1 1 1 1 | 2 2 2 2
```

Data File

Figure 47. Unformatted Print Data File. The data area of this example does not show the Record ID.

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| R1| R2| R3|
| AAAA| BBBB| 1 1 1 1| 2 2 2 2|
| XXXX| YYYY| Z ZZ|
```

Figure 48. Data Arranged on the Printed Page
**POSITION Subcommand in Processing Fields Example**

The POSITION subcommand of each LAYOUT command specifies the layout position relative to either the logical page origin or the previous LAYOUT position. The POSITION subcommands below FIELD commands specify a field position relative to the governing LAYOUT position.

This subcommand is for use in positioning text, objects, and graphics. If RELATIVE is specified or POSITION is not specified, the baseline of the Position is relative to the previous LAYOUT position.

1. For PAGEHEADER LAYOUT the baseline position can be anywhere on a logical page.
2. For PAGETRAILER, GROUPHEADER, and BODY LAYOUT the baseline position can be anywhere on a logical page and can be specified as RELATIVE.

Following POSITION subcommands come the horizontal \((x)\) then the vertical \((y)\) offsets from the reference point:

\(x\) Specifies the horizontal offset from the left side of the logical page.
\(y\) Specifies the vertical offset from the top side of the logical page.

They are parallel in structure to the OFFSET subcommand of the form definition.

For example, the final POSITION subcommand on the previous example places the final field \(1 + 4\) inches to the right of the left edge of the logical page, combining the \(x\) value of \(1\) in the LAYOUT command, and the \(x\) value of \(4\) in the nested FIELD command. The \(0\) in the FIELD command specifies no change to the \(y\) value in the LAYOUT command. Thus, the position of the final field is \(5\) IN \((x)\), \(2\) IN \((y)\).

**Note:** The first FIELD command within each LAYOUT has no position specification because the LAYOUT POSITION value is the default for the first FIELD command nested under it.

Alternative controls for the \(x\) and \(y\) values of a POSITION subcommand are available. See the description of the POSITION subcommand in FIELD command (Record Format).

**FIELD Command in Processing Fields Example**

In the FIELD command, the START and LENGTH parameters specify the location of the field in the record to be processed. START indicates the starting byte position, and LENGTH specifies the number of bytes in the field.

```plaintext
setunits linesp 6 lpi;
PAGEDEF rel9 replace yes
direction across width 8.5 in height 11.0 in;
FONT GS12 GS12;
LAYOUT 'abc' position 0 IN 1.0 IN;

/* The fields will be placed at +120 pels, +24 pels (next) */
/* and +48 pels (.20 IN) from lines previously placed on page */

setunits linesp 10 lpi;
LAYOUT 'def' position 0 relative next;
FIELD START 1 LENGTH 3 position 0 IN .5 IN;
FIELD START 4 LENGTH 3 position 0 IN next;
FIELD START 7 LENGTH 3 position current .20 IN;
```
Printing Lines in Two Directions on a Page

Lines can be printed in any of four directions, depending on the type of printer that is used. See your printer's documentation for the print directions that are supported by your printer.

The four parameters for line direction are ACROSS, DOWN, BACK, and UP. The PPFA commands that are used to format a line-data file with lines that are printed in more than one direction (as shown in Figure 49) are stated in the following page definition:

```
PAGEDEF ATOG
  DIRECTION ACROSS ;
  FONT GS12 GS12;
  LAYOUT 'abc' POSITION 1 IN ABSOLUTE 1 IN ; /*LINES A-E */
  LAYOUT 'def' POSITION .5 IN ABSOLUTE 6 IN /*LINE F */
    DIRECTION UP ;
  LAYOUT 'ghi' POSITION 1 IN ABSOLUTE 6 IN ; /*LINE G */
```

```
Figure 49. A Printout with More Than One Line Direction. The data area of this example does not show the Record ID.
```

In the page definition, the logical page direction ACROSS is specified. This value is the default, but its inclusion clarifies that no direction control is needed for lines A - E. The default direction of a layout is the direction specification of the logical page of which it is part. The LAYOUT command for the record F has a DIRECTION subcommand because the direction specification changes from the specification of the previous line. Record G is to be printed in the ACROSS direction again. However, a direction is not specified because the ACROSS direction is the default for all lines in the page definition.

Printing Fields in Two Directions on the Same Page

This example is similar to Printing Lines in Two Directions on a Page, except that you learn how to control direction field by field. This method creates a field-processing page definition and places direction controls in the FIELD commands. This command stream contains a portion of the page definition controls, showing only the LAYOUT commands:

```
LAYOUT 'abc' POSITION LEFTMARGIN TOPMARGIN NEWPAGE;
  FIELD START 1 LENGTH 4 ;
LAYOUT 'def' POSITION 2 IN ABSOLUTE 4 IN ;
  FIELD START 7 LENGTH 4
    DIRECTION UP ;
```

```
As expected in field processing, FIELD commands are nested within LAYOUT commands. Figure 50 on page 73 shows a simplified portion of an unformatted file and two pages of the printout that is formatted by the page definition, part of which is shown in the command stream. Two layouts are specified because the data file contains two input record formats (1 and 3 are alike; 2 and 4 are alike)
and because the fields are mapped to two different positions in the output. The assumption of this sample is that the data file is much longer than the portion shown. However, if the records in the file alternate in format as the first four do, the two LAYOUTs of the page definition format as many records as are presented, two to a page, on pages 1 - n.

If more than two mappings are required by the print job, more than two LAYOUT commands are required in the page definition.

**Varying Fonts on a Page**

This example illustrates a simple font variation within a printout. The task is to print a line-data file with the first line of each page in bold-faced type and the rest in standard type. This task requires controls for two fonts in the page definition.

The commands to select a single font for the page, as shown in Figure 52 on page 74 are as follows:

The FONT command contains two names: the local (STANDARD) name and the user-access (M101) name for the selected font.

```
PAGDEFF ABCD;
  FONT STANDARD M101;
  FONT BOLDFACE M102;
  LAYOUT 'abc' FONT BOLDFACE NEWPAGE;
  LAYOUT 'def' FONT STANDARD NEWPAGE;
  LAYOUT 'ghi' FONT STANDARD;
```

**Note:** Fonts cannot be a Font Typeface Global Identifier (FGID). Also, all page definitions require a LAYOUT command.

The following example shows line data that uses a single font:
This command stream works on the principle that each line of output whose font you want to change from the font in the previous line must be controlled by a separate LAYOUT command. The FONT subcommand of the LAYOUT command names the font that is wanted for that line. In this example, two LAYOUT commands are used because one font change and two fonts are intended for the output. The user-access font names appear in the two FONT commands immediately below the PAGEDEF command and optionally, a local name. M101 and M102 in the example are user-access names; BOLDFACE is a local name. Use the local name in the FONT subcommand of LAYOUT if it is included in the corresponding FONT command, as is done for the first LAYOUT command.

```
def    Record 1
ghi    Record 2
ghi    Record 3
ghi    Record 4
ghi    Record 5
ghi    Record 6
def    Record 7
ghi    Record 8
ghi    Record 9
def    Record 10
ghi    Record 11
ghi    Record 12
ghi    Record 13
```

Figure 51. Line Data for Single Font Example

```
Record 1
Record 2
Record 3
Record 4
Record 5
Record 6
Record 7
Record 8
Record 9
Record 10
Record 11
Record 12
Record 13
```

Figure 52. Data File Printed Using a Single Font

```
def    Record 1
ghi    Record 2
ghi    Record 3
ghi    Record 4
ghi    Record 5
ghi    Record 6
def    Record 7
ghi    Record 8
ghi    Record 9
def    Record 10
ghi    Record 11
ghi    Record 12
ghi    Record 13
abc    Record 1
ghi    Record 2
ghi    Record 3
ghi    Record 4
ghi    Record 5
ghi    Record 6
ghi    Record 7
ghi    Record 8
ghi    Record 9
abc    Record 10
ghi    Record 11
ghi    Record 12
ghi    Record 13
```

Figure 53. Line Data for Two Font Example
Changing fonts field by field is similar to changing them in layouts. You map each field individually with a FIELD command; include a FONT subcommand in the FIELD command. If a font change is wanted for a field, as with the FONT subcommand of a LAYOUT command, the font must be previously named in a FONT command.

**Rotating Fonts**

Fonts rotate relative to the inline direction of lines (or fields).

This example focuses on a single letter A from FONTA. With PPFA, a single font that is specified in a page definition can produce letters in any of four rotations. This formatting is accomplished by a FONT command that specifies rotation. If you want to vary the rotation of a font twice within a page, you use two FONT commands, one for each rotation. You also use two LAYOUT commands to map the data to the printout by using the two rotations of the font. In a field processing application, FIELD commands can be used in the same way. These LAYOUT commands name the rotated font in a FONT subcommand.

[Figure 55 on page 76](#) breaks down the elements that are required for the FONT commands and subcommands. Distinct local names and rotation specifications for each font are placed in a FONT command. These specifications identify a font as rotated within a page definition. The rotation of a character is relative to the inline direction of a field or LAYOUT. The characters and rotations that are shown here assume an inline direction of ACROSS.
You can use up to 16 possible combinations of logical page direction and font rotation for page printers other than the 3800.

The FONT subcommands within LAYOUT or FIELD commands that name the rotated font in that page definition use only the local name. The following command stream shows the proper specification and nesting of FONT commands and subcommands for rotation.

```markdown
PAGEDEF ABCD ;
  FONT FONTA M103 ; /*NO ROTATION, LOCAL AND USER-ACCESS NAMES. */
  FONT FONTARTD180 M103 /*ROTATED FONT, LOCAL, USER-ACCESS*/
    ROTATION 180 ; /*NAMES PLUS ROTATION SUBCOMMAND*/
  LAYOUT 'abc' FONT FONTA ; /*LOCAL NAME*/
  LAYOUT 'def' FONT FONTARTD180 ; /*LOCAL NAME*/
```

**Figure 55. Character Rotation**

![Figure 55. Character Rotation](image)

**Figure 56. Example of Assumed Data File and Rotation Specifications**

FONTA, identified in the first FONT command, requires no rotation parameter because it is printed in the default position (or 0° rotation) for font M103. For the rotated font, the second FONT command identifies FONTARTD180 (the local name) as M103 rotated 180°.
Using Traditional Kanji Formatting

Traditional kanji print presentation, called *tate*, is possible with your print server printers by using a combination of font rotation and logical page direction. A logical page in the DOWN direction and a 270° font rotation provide the right combination to present kanji in tate format on a print server printer.

FORMDEF TATE
OFFSET 1 IN 1 IN ;
PAGEDEF TATE
HEIGHT 5 IN
WIDTH 6 IN
DIRECTION DOWN ;
FONT KANJIRTD M104
ROTATION 270 ;
LAYOUT 'tate' FONT KANJIRTD;

Figure 57 shows the result of formatting with the page definition in the example. The characters are added to lines down the page. Lines are added right to left.

Record Formatting Examples

To allow different formats for different groups (or tables) of data, each of which have an unpredictable number of entries, a Record ID is assigned to each output record to identify the type of record and control layout formatting. An application can group data fields that are to be formatted together as an entity into Data Records with a specific Record ID. For example, in a bank statement, the data fields for a check transaction might be grouped with a Record ID identifying that record as a check transaction. The PAGEDEF would then define a special layout format for a check transaction with a matching Record ID.

The same thing might be done for a deposit transaction, customer account information, deposit totals, and check totals. If the customer account information is going to be used in a page header on each page, the PAGEDEF can define a special layout format for a customer information record that automatically generates a page header for each page.

This section shows two complete examples that use the record formatting process. Each is divided into three parts - the output you want (after PAGEDEF processing), the application output (before PAGEDEF processing), and the PPFA commands.
Example 1 - Bank Statement

Figure 58 and Figure 59 on page 79 show the printed output for a bank statement after PAGedef processing (the second figure is resized). “Application Output (before PAGedef Processing)” on page 79 and “PPFA Commands” on page 81 show the user data and the PPFA commands that create the printed output.

![Bank Statement](image)

**Big Brother Bank**

“*We watch over you*”

P.O. Box 1573

Beantown, MA 02116

Justin Case

123 Redlight Lane

TwistNshout, MA 02345

**Account Number:** 026-257311

**Statement Begin Date:** JAN 02, 2002

**Statement End Date:** FEB 01, 2002

---

**Super Checking Account Activity**

<table>
<thead>
<tr>
<th>Description</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPOSIT</td>
<td>01/05/02</td>
<td>26.90</td>
</tr>
<tr>
<td>AUTO DEPOSIT</td>
<td>01/15/02</td>
<td>954.27</td>
</tr>
<tr>
<td>AUTO DEPOSIT</td>
<td>01/30/02</td>
<td>954.27</td>
</tr>
<tr>
<td>INTEREST</td>
<td>01/31/02</td>
<td>11.49</td>
</tr>
</tbody>
</table>

**Total Credits** 1946.93

---

**Checks**

<table>
<thead>
<tr>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>352</td>
<td>01/04/02</td>
<td>$321.50</td>
<td>353</td>
<td>01/05/02</td>
<td>$100.00</td>
</tr>
<tr>
<td>354</td>
<td>01/10/02</td>
<td>$122.30</td>
<td>355</td>
<td>01/11/02</td>
<td>$59.95</td>
</tr>
<tr>
<td>356</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>357</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>358</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>359</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>360</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>361</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>362</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>363</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>364</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>365</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>366</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>367</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>368</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>369</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>370</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>371</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>372</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>373</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>374</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>375</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>376</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>377</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>378</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>379</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>380</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>381</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>382</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>383</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>384</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>385</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>386</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>387</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>388</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>389</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>390</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>391</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>392</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>393</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>394</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>395</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
</tbody>
</table>

---

Figure 58. Bank Statement (Page 1 of 2)
Big Brother Bank  
“We watch over you”  
P.O. Box 1573  
Beantown, MA  02116  

Account Number: 026-257311  
Statement Begin Date: JAN 02, 2002  
Statement End Date: FEB 01, 2002  

Justin Case  
123 Redlight Lane  
TwistNshout, MA 02345  

<table>
<thead>
<tr>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>396</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>397</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>398</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>399</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>400</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>401</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>402</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>403</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>404</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>405</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>406</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>407</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>408</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>409</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>410</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>411</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>412</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>413</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>414</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>415</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>416</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>417</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
<tr>
<td>418</td>
<td>01/15/02</td>
<td>$852.33</td>
<td>419</td>
<td>01/30/02</td>
<td>$500.35</td>
</tr>
</tbody>
</table>

Total Checks 1956.43

<table>
<thead>
<tr>
<th>Date</th>
<th>Balance</th>
<th>Date</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/04/02</td>
<td>$2269.74</td>
<td>01/05/02</td>
<td>$2196.64</td>
</tr>
<tr>
<td>01/10/02</td>
<td>$2074.34</td>
<td>01/11/02</td>
<td>$2014.39</td>
</tr>
<tr>
<td>01/15/02</td>
<td>$2016.33</td>
<td>01/30/02</td>
<td>$2570.25</td>
</tr>
</tbody>
</table>

Final Balance $2581.74

Interest Rate as of 01/04 ** * 5.321%

Figure 59. Bank Statement (Page 2 of 2).

Application Output (before PAGDEDEF Processing): Each layout record contains all information for a specified layout. Because of lack of space, only the first 80 bytes are shown here. The first 10 characters must contain the layout ID.
1234567890 026-257311 Justin Case 123 Redlight Lane Twistnshout MA 02345

statmid 026-257311 Justin Case 123 Redlight Lane Twistnshout MA 02345
statsum $2591.24 $1946.93 $1956.43 $0.00 $2581.72
pgenum

crheader
ckdatal DEPOSIT 01/06/02 $26.90
crdata AUTO DEPOSIT 01/15/02 $954.27
crdata AUTO DEPOSIT 01/30/02 $954.27
crdata INTEREST 01/31/02 $11.49

crheader
ckdatal 352 01/04/02 $321.50
ckdatar 353 01/05/02 $100.00
ckdatar 354 01/10/02 $122.30
ckdatar 355 01/11/02 $59.95
ckdatar 356 01/15/02 $852.33
ckdatar 357 01/30/02 $500.35
ckdatar 358 01/15/02 $852.33
ckdatar 359 01/30/02 $500.35
ckdatar 360 01/15/02 $852.33
ckdatar 361 01/30/02 $500.35
ckdatar 362 01/15/02 $852.33
ckdatar 363 01/30/02 $500.35
ckdatar 364 01/15/02 $852.33
ckdatar 365 01/30/02 $500.35
ckdatar 366 01/15/02 $852.33
ckdatar 367 01/30/02 $500.35
ckdatar 368 01/15/02 $852.33
ckdatar 369 01/30/02 $500.35
ckdatar 370 01/15/02 $852.33
ckdatar 371 01/30/02 $500.35
ckdatar 372 01/15/02 $852.33
ckdatar 373 01/30/02 $500.35
ckdatar 374 01/15/02 $852.33
ckdatar 375 01/30/02 $500.35
ckdatar 376 01/15/02 $852.33
ckdatar 377 01/30/02 $500.35
ckdatar 378 01/15/02 $852.33
ckdatar 379 01/30/02 $500.35
ckdatar 380 01/15/02 $852.33
ckdatar 381 01/30/02 $500.35
ckdatar 382 01/15/02 $852.33
ckdatar 383 01/30/02 $500.35
ckdatar 384 01/15/02 $852.33
ckdatar 385 01/30/02 $500.35
ckdatar 386 01/15/02 $852.33
ckdatar 387 01/30/02 $500.35
ckdatar 388 01/15/02 $852.33
ckdatar 389 01/30/02 $500.35
ckdatar 390 01/15/02 $852.33
ckdatar 391 01/30/02 $500.35
ckdatar 392 01/15/02 $852.33
ckdatar 393 01/30/02 $500.35
ckdatar 394 01/15/02 $852.33
ckdatar 395 01/30/02 $500.35
ckdatar 396 01/15/02 $852.33
ckdatar 397 01/30/02 $500.35
ckdatar 398 01/15/02 $852.33
ckdatar 399 01/30/02 $500.35
ckdatar 400 01/15/02 $852.33
ckdatar 401 01/30/02 $500.35
ckdatar 402 01/15/02 $852.33
ckdatar 403 01/30/02 $500.35
ckdatar 404 01/15/02 $852.33
ckdatar 405 01/30/02 $500.35
ckdatar 406 01/15/02 $852.33
PPFA Commands:

PAGEDEF justin replace yes
  WIDTH 8.5 in
  HEIGHT 11.0 in;
  FONT comp a075nc ; /*Big Brother Bank font */
  FONT Ital a175dc ; /*Italic theme */
  FONT addr a075dc ; /*Big Brother address */
  FONT varb gt10 ; /*Variable data */
  FONT super a075dc ; /*Super Checking Account */
  FONT head a055ac ; /*Headings */
  FONT bhead a075ac ; /*Bold Headings */

PAGEFORMAT chub1 TOPMARGIN 2 in BOTMARGIN 2 in;
/** statmid BODY **/
/**************************************************************************/
LAYOUT C'statmid' PAGEHEADER NEWPAGE
  POSITION .6 in ABSOLUTE .55 in;
  FIELD TEXT C'Big Brother Bank' ALIGN LEFT
  FONT comp ; /* default to LAYOUT positioning*/
  FIELD TEXT C'We watch over you' ALIGN LEFT
  POSITION 0 NEXT
  FONT Ital ; /*default to next line */
  FIELD TEXT C'P.O. Box 1573' ALIGN LEFT
  POSITION 0 NEXT
  FONT addr ; /*default to next line */
  FIELD TEXT C'Beantown, MA 02116' ALIGN LEFT
  POSITION 0 NEXT
  FONT addr ; /*default to next line */
  FIELD TEXT C'Account Number:' ALIGN LEFT
  POSITION 4.3 in .2 in
  FONT head ; /*New area on right */
  FIELD TEXT C'Statement Begin Date:' ALIGN LEFT
  POSITION 4.3 in NEXT
  FONT head ; /*New area on right */
  FIELD TEXT C'Statement End Date:' ALIGN LEFT
  POSITION 4.3 in NEXT
  FONT head ; /*New area on right */
  FIELD START 1 LENGTH 10 ALIGN RIGHT
  POSITION 7.5 in .2 in
  FONT varb ; /*variable - account number*/
  FIELD START 75 LENGTH 12
POSITION 7.5 in NEXT
ALIGN RIGHT /* data is missing from example */
FONT varb ; /*variable - begin date */
FIELD START 88 LENGTH 12
POSITION 7.5 in NEXT
ALIGN RIGHT /* data is missing from example */
FONT varb ; /*variable - end date */
FIELD START 11 LENGTH 19 ALIGN LEFT
POSITION 1.1 in .9 in
FONT varb ; /*variable - customer name */
FIELD START 30 LENGTH 19 ALIGN LEFT
POSITION 1.1 in NEXT
FONT varb ; /*variable - customer address */
FIELD START 49 LENGTH 22 ALIGN LEFT
POSITION 1.1 in NEXT
FONT varb ; /*variable - customer city, st. */

/*********************/
/** statsum BODY **/
/*********************/
LAYOUT C'statsum' BODY
POSITION .6 in .5 in;
FIELD TEXT C'Super Checking Account Activity'
FONT super ; /* Static text - Super Checking */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
POSITION 0 .15 in
COPY down 2 spaced 1 mm;
FIELD TEXT C'Beginning Balance'
POSITION .3 in .4 in
FONT head ; /* Static text - first header */
FIELD TEXT C'Credits'
POSITION 2.4 in CURRENT
FONT head ; /* Static text - first header */
FIELD TEXT C'Debits'
POSITION 3.6 in CURRENT
FONT head ; /* Static text - first header */
FIELD TEXT C'Service Charge'
POSITION 4.8 in CURRENT
FONT head ; /* Static text - first header */
FIELD TEXT C'Ending Balance'
POSITION 6.3 in CURRENT
FONT head ; /* Static text - first header */
FIELD START 1 LENGTH 8
POSITION .6 in .6 in
FONT varb ; /* Variable text - Beg balance */
FIELD START 10 LENGTH 8
POSITION 2.2 in CURRENT
FONT varb ; /* Variable text - Credits */
FIELD START 20 LENGTH 8
POSITION 3.4 in CURRENT
FONT varb ; /* Variable text - Debits */
FIELD START 30 LENGTH 5
POSITION 5.0 in CURRENT
FONT varb ; /* Variable text - Service Chrg */
FIELD START 40 LENGTH 8
POSITION 6.5 in CURRENT
FONT varb ; /* Variable text - End Balance */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
POSITION 0 .7 in;

/*********************/
/** crheader GROUPHEADER **/
/*********************/
LAYOUT C'crheader' GRPHEADER XSPACE .2 in
POSITION SAME .9 in;
FIELD TEXT C'Credits'
FONT bhead ; /* Static text - Credits */
FIELD TEXT 'Description'
   POSITION 1.3 in CURRENT
FONT head ; /* Stat text - Deposit Descr. */
FIELD TEXT 'Date'
   POSITION 3.2 in CURRENT
FONT head ; /* Static text - Date */
FIELD TEXT 'Amount'
   POSITION 5.0 in CURRENT
FONT head ; /* Stat text - Amount of deposit*/
DRAWGRAPHIC LINE ACROSS 6.2 IN LINEWT BOLD
   POSITION 1.3 in next;

******************************************************************************
/** crdata BODY ***************************************************************/
******************************************************************************
LAYOUT C'crdata' BODY GROUP;
FIELD START 1 LENGTH 13
   POSITION 1.3 in CURRENT
FONT varb ; /* Variable text - Description */
FIELD START 14 LENGTH 8
   POSITION 3 in CURRENT
FONT varb ; /* Variable text - Date */
FIELD START 24 LENGTH 8 ALIGN RIGHT
   POSITION 5.6 in CURRENT
FONT varb ; /* Variable text - Amount */
******************************************************************************
/** crttotal BODY ***************************************************************/
******************************************************************************
LAYOUT C'crtotal' BODY GROUP;
FIELD TEXT C'Total Credits'
   POSITION 1.5 in .2 in
FONT bhead ; /* Stat text - Total credits */
FIELD START 24 LENGTH & ALIGN RIGHT
   POSITION 7.3 in CURRENT
FONT varb ; /* Variable text - Amount */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
   POSITION 0 next;
******************************************************************************
/** ckheader GROUPHEADER ***********************************************************/
******************************************************************************
LAYOUT C'ckheader' GRPHEADER XSPACE .2 in
   POSITION SAME .6 in;
FIELD TEXT C'Checks'
   FONT bhead ; /* Static text - Checks */
FIELD TEXT C'Check No.'
   POSITION 1.4 in CURRENT
FONT head ; /* Stat text - Check number */
FIELD TEXT C'Date'
   POSITION 2.5 in CURRENT
FONT head ; /* Stat text - Date of check */
FIELD TEXT C'Amount'
   POSITION 3.5 in CURRENT
FONT head ; /* Stat text - Amount of check*/
FIELD TEXT C'Check No.'
   POSITION 4.6 in CURRENT
FONT head ; /* Stat text - Check number */
FIELD TEXT C'Date'
   POSITION 5.6 in CURRENT
FONT head ; /* Stat text - Date of check */
FIELD TEXT C'Amount'
   POSITION 6.8 in CURRENT
FONT head ; /* Stat text - Amount of check*/
DRAWGRAPHIC LINE ACROSS 6.2 IN LINEWT BOLD
   POSITION 1.3 in next;
DRAWGRAPHIC LINE DOWN LINETYPE shortdash
POSITION 4.5 in CPOS;

/**********************************************************/
** ckdata1 BODY left side */
/**********************************************************/
LAYOUT C'ckdata1' BODY GROUP
POSITION SAME NEXT;
FIELD START 2 LENGTH 3
POSITION 1.4 in CURRENT
FONT varb ; /* Variable text - Check number */
FIELD START 14 LENGTH 8
POSITION 2.4 in CURRENT
FONT varb ; /* Variable text - Date */
FIELD START 24 LENGTH 8 ALIGN RIGHT
POSITION 4.4 in CURRENT
FONT varb ; /* Variable text - Amount */

/**********************************************************/
** ckdatar BODY right side */
/**********************************************************/
LAYOUT C'ckdatar' BODY GROUP
POSITION SAME SAME;
FIELD START 2 LENGTH 3
POSITION 4.6 in CURRENT
FONT varb ; /* Variable text - Check number */
FIELD START 14 LENGTH 8
POSITION 5.6 in CURRENT
FONT varb ; /* Variable text - Date */
FIELD START 24 LENGTH 8 ALIGN RIGHT
POSITION 7.5 in CURRENT
FONT varb ; /* Variable text - Amount */

/**********************************************************/
** cktotal BODY */
/**********************************************************/
LAYOUT C'ckttotal' BODY GROUP;
ENDGRAPHIC LPOS; /*ends dashed line between checks*/
FIELD TEXT C'Total Checks'
POSITION 1.5 in .2 in
FONT bhead ; /* Stat text - Total checks */
FIELD START 24 LENGTH 8 ALIGN RIGHT
POSITION 7.3 in CURRENT
FONT varb ; /* Variable text - Amount */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
POSITION 0 next;

/**********************************************************/
** balhead GROUPHEADER */
/**********************************************************/
LAYOUT C'balhead' GRPHEADER XSPACE .2 in
POSITION SAME .6 in;
FIELD TEXT C'Daily'
FONT bhead ; /* Static text - Daily Balance */
FIELD TEXT C'Date'
POSITION 1.3 in CURRENT
FONT head ;/* Stat text - Date of balance */
FIELD TEXT C'Balance'
POSITION 2.8 in CURRENT
FONT head ;/* Static text - Balance */
FIELD TEXT C'Balance'
POSITION 4.3 in CURRENT
FONT head ; /* Stat text - Date of balance */
FIELD TEXT C'Balance'
POSITION 5.8 in CURRENT
FONT head ; /* Static text - Balance */
FIELD TEXT C'Balances'
POSITION 0 NEXT

84  Page Printer Formatting Aid: User's Guide
Chapter 4. Using Page Definition Commands for Record Format Line Data and XML Data
Example 2 - Statement that Uses Repeated and Unended Boxes

Figure 60 shows the printed output for a statement that uses the repeated box option, a single circle and some unended boxes during PAGEDEF processing.

"Application Output (before PAGEDEF Processing)" and "PPFA Commands" show the user data and the PPFA commands that create the printed output.

<table>
<thead>
<tr>
<th>Date:</th>
<th>Check no.</th>
<th>Payable to:</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/04/02</td>
<td>352</td>
<td>Walmart</td>
<td>$321.50</td>
</tr>
<tr>
<td>01/05/02</td>
<td>353</td>
<td>Sears</td>
<td>$100.00</td>
</tr>
<tr>
<td>01/10/02</td>
<td>354</td>
<td>Boulder Bookstore</td>
<td>$122.30</td>
</tr>
<tr>
<td>01/11/02</td>
<td>355</td>
<td>Kathy's Pretty Things</td>
<td>$59.95</td>
</tr>
<tr>
<td>01/15/02</td>
<td>356</td>
<td>Pirie Racing Enterprises</td>
<td>$852.33</td>
</tr>
<tr>
<td>01/30/02</td>
<td>357</td>
<td>Rockley's Music Center</td>
<td>$500.35</td>
</tr>
</tbody>
</table>

Figure 60. Statement that Uses the Repeating Box Option

Application Output (before PAGEDEF Processing):

11111112222222233333333344444444455555555556666666667
1234567890123456789012345678901234567890123456789012345678901

statmid  Justin Case  123 Redlight Lane  Twistnshout  MA 02345
ckheader
ckdata  352  01/04/02  $321.50  Walmart
ckdata  353  01/05/02  $100.00  Sears
ckdata  354  01/10/02  $122.30  Boulder Bookstore
ckdata  355  01/11/02  $59.95  Kathy's Pretty Things
ckdata  356  01/15/02  $852.33  Pirie Racing Enterprises
ckdata  357  01/30/02  $500.35  Rockley's Music Center
ckend

PPFA Commands:

PAGEDEF rept1 replace yes;
  FONT addr a075dc;       /*customer address */
  FONT varb gt10;       /*Variable data */
  FONT bhead a075ac;       /*Bold Headings */
  SETUNITS LINESP .25 in;    /* Line spacing */

  PAGEFORMAT rept1 BOTMARGIN 2 in;
  SEGMENT ibmlog;    /*IBM logo */
4. Using Page Definition Commands for Record Format Line Data and XML Data

/* statmid PAGEHEADER */
LAYOUT C'statmid'
  SEGMENT ibmlog 1.15 in 1.35 in
  PAGEHEADER NEWPAGE
  POSITION SAME ABSOLUTE NEXT;
  DRAWGRAPHIC CIRCLE RADIUS .5 in /* 1 inch circle */
  POSITION 1.5 in 1.5 in;
  DRAWGRAPHIC BOX BOXSIZE 2.6 IN .25 IN ROUNDED LARGE
  LINETWT 0 /* invisible border */
  POSITION 4 IN 1 IN
  COPY DOWN 2 DOT02;
  FIELD START 2 LENGTH 19 ALIGN LEFT
  POSITION 4.2 in 1.2 in
  FONT addr; /*variable - customer name */
  FIELD START 21 LENGTH 19 ALIGN LEFT
  POSITION 4.2 in NEXT
  FONT addr; /*variable - customer address */
  FIELD START 40 LENGTH 22 ALIGN LEFT
  POSITION 4.2 in NEXT
  FONT addr; /*variable - customer city, st. */
/** cheader GROUPHEADER */
LAYOUT C'cheader' GROUPHEADER XSPACE .25 in
  POSITION 1 in ABSOLUTE 2.5 in; /* set position */
  DRAWGRAPHIC BOX BOXSIZE .95 IN .3 IN
  POSITION 0 0;
  DRAWGRAPHIC BOX BOXSIZE .95 IN /* box started for data */
  POSITION 0 .3 in; /* no vertical size */
  FIELD TEXT C'Date'
  POSITION .3 in .2 in
  FONT bhead; /* Stat text - Date of check */
  DRAWGRAPHIC BOX BOXSIZE .8 IN .3 IN
  POSITION .95 IN 0;
  DRAWGRAPHIC BOX BOXSIZE .8 IN /* box started for data */
  POSITION .95 in .3 in; /* no vertical size */
  FIELD TEXT C'Check No.'
  POSITION 1 in .2 in
  FONT bhead; /* Stat text - Check number */
  DRAWGRAPHIC BOX BOXSIZE 3 IN .3 IN
  POSITION 1.75 IN 0;
  DRAWGRAPHIC BOX BOXSIZE 3 IN /* box started for data */
  POSITION 1.75 in .3 in; /* no vertical size */
  FIELD TEXT C'Payable to:
  POSITION 2.9 in .2 in
  FONT bhead; /* Static text - Payable to: */
  DRAWGRAPHIC BOX BOXSIZE .95 IN .3 IN
  POSITION 4.75 IN 0 in;
  DRAWGRAPHIC BOX BOXSIZE .95 in /* box started for data */
  POSITION 4.75 in .3 in; /* no vertical size */
  FIELD TEXT C'Amount'
  POSITION 5 in .2 in
  FONT bhead; /* Stat text - Amount of check */
/** ckdata BODY w/ un-ended boxes */
LAYOUT C'ckdata' BODY GROUP;
  FIELD START 2 LENGTH 3 ALIGN LEFT
  POSITION 1.2 in CURRENT
  FONT varb; /* Variable text - Check number */
  FIELD START 14 LENGTH 8 ALIGN LEFT
  POSITION .1 in CURRENT
  FONT varb; /* Variable text - Date */
  FIELD START 35 LENGTH 25 ALIGN LEFT
XML Page Definition Formatting Function

The XML page definition formatting function allows an application to specify formatting instructions for XML data by specifying an XLAYOUT command with specific formatting instructions for the data. The XLAYOUT command addresses an XML data item by specifying a QTAG (qualified tag) for that data. A QTAG is a series of XML start tags that fully identify the XML data item. For example, in Figure 61 on page 91 for your customer's first name, the QTAG would be Customer, name, and first. To define a local name of first for easy reference, you can use the following DEFINE command:

```
DEFINE first QTAG 'Customer','name','first';
```

You can then reference it with the following XLAYOUT command by using the defined local name of first:

```
XLAYOUT first POSITION ...;
```

Before it prints the data, PSF scans the XML data item and matches it to an XLAYOUT command in the page definition by using its QTAG. The matching XLAYOUT command in the page definition is used to position and format the associated XML data item and its attributes on the printed page.

The XML page definition function has the following new PPFA concepts:

**Relative Inline Positioning:**
Relative inline positioning places data relative to the current position. If you position a text field and then place the text, the end of the text becomes the new current position. Graphics, bar codes, objects, segments, and overlays do not change the current position after they are originally positioned. For example, if you position a line with a DRAWGRAPHIC LINE command, the new current position is the starting point of that line. The length of the graphic line does not change the current position.

Be aware of several restrictions when you are using relative inline positioning:

1. XLAYOUT commands with relative positioning cannot contain any of these:
   - FIELD commands with inline positioning relative to the XLAYOUT (LPOS)
   - FIELD ATTR (attribute) with inline positioning relative to the XLAYOUT (LPOS)
   - FIELD commands with bar codes
   - DRAWGRAPHIC commands
   - OBJECT subcommands
   - SEGMENT subcommands

---

The XML page definition formatting function allows an application to specify formatting instructions for XML data by specifying an XLAYOUT command with specific formatting instructions for the data. The XLAYOUT command addresses an XML data item by specifying a QTAG (qualified tag) for that data. A QTAG is a series of XML start tags that fully identify the XML data item. For example, in Figure 61 on page 91 for your customer's first name, the QTAG would be Customer, name, and first. To define a local name of first for easy reference, you can use the following DEFINE command:

```
DEFINE first QTAG 'Customer','name','first';
```

You can then reference it with the following XLAYOUT command by using the defined local name of first:

```
XLAYOUT first POSITION ...;
```

Before it prints the data, PSF scans the XML data item and matches it to an XLAYOUT command in the page definition by using its QTAG. The matching XLAYOUT command in the page definition is used to position and format the associated XML data item and its attributes on the printed page.

The XML page definition function has the following new PPFA concepts:

**Relative Inline Positioning:**
Relative inline positioning places data relative to the current position. If you position a text field and then place the text, the end of the text becomes the new current position. Graphics, bar codes, objects, segments, and overlays do not change the current position after they are originally positioned. For example, if you position a line with a DRAWGRAPHIC LINE command, the new current position is the starting point of that line. The length of the graphic line does not change the current position.

Be aware of several restrictions when you are using relative inline positioning:

1. XLAYOUT commands with relative positioning cannot contain any of these:
   - FIELD commands with inline positioning relative to the XLAYOUT (LPOS)
   - FIELD ATTR (attribute) with inline positioning relative to the XLAYOUT (LPOS)
   - FIELD commands with bar codes
   - DRAWGRAPHIC commands
   - OBJECT subcommands
   - SEGMENT subcommands
2. You can use the SAME parameter for inline positioning on the XLAYOUT command only when the previously used XLAYOUT command used absolute inline positioning.

Absolute Inline Positioning:
Allows absolute inline positioning on a FIELD command for specific placement of elements.

Attributes are Special FIELDS:
The attribute is identified by name and the data that is printed is from the attribute value or a portion of the attribute value and not from the element content.

Notes:
1. If a FIELD is used for presenting any piece of data on the XLAYOUT command, FIELD commands must be used for all pieces of data that is presented on the XLAYOUT command. Because an attribute is a special field, if you want to print both an attribute value and the element data, you need to code the attribute field for the attribute value and a regular field for the element data.
2. PSF suppresses leading and trailing blanks (X'40' for EBCDIC or X'20' for ASCII) in the data. Multiple embedded blanks are reduced to one blank.

POSITION Subcommand
The POSITION subcommand on each XLAYOUT command specifies the starting layout position in the printout. The POSITION subcommand on the FIELD command specifies a field position relative to the governing XLAYOUT position.

Relative Baseline Position
See the Advanced Function Presentation: Programming Guide and Line Data Reference for more detailed information on RCD and XMD processing.

The baseline position corresponds to the $y$-pos position in PPFA and the inline position corresponds to the $x$-pos position.

If the baseline position is relative, the offset is measured as follows:
- For Page Header, the offset is relative to the top of the page.
- For Page Trailer, it is relative to the last Record processed; if none exists, it is relative to the top margin.
- For Group Header and Body, the offset is relative to the last Group Header or Body processed; if none exists, it is relative to the top margin.
- For Field, it is relative to the last Field or Body that was processed for print (whether or not data is printed).

The inline position is not changed in the case where no data exists to print.

For example:

```
XLAYOUT text1
  POSITION 6 mm 8 mm;
  FIELD ATTR 'attr1' FONT FONT03;
  FIELD ATTR 'attr2' FONT FONT03 POSITION 20 mm 5 mm;
  FIELD ATTR 'attr3' FONT FONT03;
```
attr1 = abcdefg
attr2 = W
attr3 = hjkmnp

The output is:

```
    abcdefg
      Whjkmnp
```

In the output, W is positioned at 6+20 mm across and 8+5 mm down and hjkmnp is positioned next to W. Because no POSITION is specified for FIELD for attr3, its starting point is the current position after attr2 is printed.

If attr2 has no value:
attr1 = abcdefg
attr2 =
attr3 = hjkmnp

The output is:

```
    abcdefg
      hjkmnp
```

In the output, hjkmnp is positioned next to abcdefg but underneath, down 5 mm. Because no POSITION is specified for FIELD for attr3, its starting point is the current position after attr2 is printed. When no data exists for attr2, the x position does not change from where it ended with attr1, but the y position moves down 5 mm.

**XML Data Element Example**

An application can group XML data elements to be formatted together as an entity by grouping those elements hierarchically under a collection XML data element. The data order normally does not matter in formatting the data elements unless the elements are to be placed relative to each other in the inline direction. Any elements to be placed inline relative to each other must be ordered in inline presentation order. Use the XLAYOUT/FIELD commands to place the data on the presentation device. Figure 61 on page 91 is an example of a bank customer with the name and address fields that are placed together:
<Customer>
    <name>
        <title>Dr.</title>
        <first>Kelly</first>
        <last>Green</last>
    </name>
    <address>
        <strno>1911</strno>
        <street>Colt Lane</street>
        <city>Longmont</city>
        <state>CO</state>
        <zip>80501</zip>
    </address>
</Customer>

Figure 61. XML Data Elements

The example in Figure 61 results in the following printed output:

Dr. Kelly Green
1911 Colt Lane
Longmont, CO 80501

The page definition that is used to create the output is as follows:

PAGEDEF xmp101 UDTYPE ebcDIC REPLACE yes;
/*----------------------------------------*/
/* Font definitions: */
/*----------------------------------------*/
FONT E21HOC TYPE EBCDIC;

/*----------------------------------------*/
/* Use QTAG definitions to define short alias names */
/* that make coding the XLAYOUTs easier. Do the */
/* messy work here, allowing us to code on the XLAYOUT: */
/* XLAYOUT zip ... */
/* instead of: */
/* XLAYOUT QTAG 'Customer', 'address', 'zip' ... */
/*----------------------------------------*/
Define cust QTAG 'Customer' ;
Define title QTAG 'Customer', 'name', 'title';
Define first QTAG 'Customer', 'name', 'first';
Define last QTAG 'Customer', 'name', 'last';
Define strno QTAG 'Customer', 'address', 'strno';
Define street QTAG 'Customer', 'address', 'street';
Define city QTAG 'Customer', 'address', 'city';
Define state QTAG 'Customer', 'address', 'state';
Define zip QTAG 'Customer', 'address', 'zip';

/*----------------------------------------*/
/* Print first line "Dr. Kelly Green" */
/* NOTE:-The "collector" Customer starts a new page */
/* -RELATIVE 0 is not the same as SAME */
/* -RELATIVE 0.167 is equivalent to a 6 CPI space */
/* along with FIELD TEXT, giving us 2 ways to */
/* leave a space. */
/* -Watch out for the POSITION defaults on XLAYOUT */
/* and FIELDS */
/*----------------------------------------*/
XLAYOUT cust NEWPAGE;
XLAYOUT title POSITION ABSOLUTE 1 in ABSOLUTE 1 in;
XLAYOUT first POSITION RELATIVE 0 in RELATIVE 0;
FIELD TEXT'';
FIELD START 1 LENGTH *=;
XLAYOUT last POSITION RELATIVE 0.167 in RELATIVE 0;
In Figure 61 on page 91, the XML data items Dr., Kelly, and Green are printed relative to each other by using relative inline positioning. This positioning can be done only because the data appears in the following order: the title, Dr. is first; the first name, Kelly is next; and the last name, Green is last. However, if you wanted to use this data, and change the order of the names to print the last name followed by the first name, you must position the names by using absolute inline positioning because the data cannot be reordered with relative inline positioning.

XML Data Format Example

XML allows the same data to be used for multiple presentation media. In Figure 62 on page 93, Figure 66 on page 97 XML data file is shown formatted for printing with PPFA's XML support.
<?xml version="1.0"?>
<bankstatement>
  <customer>
    <acctno>026-257311</acctno>
    <name>Justin Case</name>
    <street>123 Redlight Lane</street>
    <cityst Twistnhout, MA 02345</cityst>
  </customer>
  <begindate>JAN 02, 2002</begindate>
  <enddate>FEB 01, 2002</enddate>
</bankstatement>

<!-- Page number generator -->
<supercheckingactivity type="superchk">
  <balance>
    <begin>2591.24</begin>
    <credit>1946.93</credit>
    <debit>1956.43</debit>
    <svchg>0.00</svchg>
    <end>0.00</end>
  </balance>
  <credits>
    <transaction>
      <type>DEPOSIT</type>
      <date>01/05/2002</date>
      <amt>26.90</amt>
    </transaction>
    <transaction>
      <type>AUTO DEPOSIT</type>
      <date>01/05/2002</date>
      <amt>954.27</amt>
    </transaction>
    <transaction>
      <type>AUTO DEPOSIT</type>
      <date>01/30/2002</date>
      <amt>954.27</amt>
    </transaction>
  </credits>
</supercheckingactivity>

Figure 62. XML Data File (Part 1 of 5)
<transaction>
  <type>INTEREST</type>
  <date>01/31/2002</date>
  <amt> 11.49</amt>
</transaction>

<credits>  
</credits>

<checks>
  <transaction>
    <chkno>352</chkno>
    <date>01/04/2002</date>
    <amt> 321.50</amt>
  </transaction>
  <transaction>
    <chkno>353</chkno>
    <date>01/05/2002</date>
    <amt> 100.00</amt>
  </transaction>
  <transaction>
    <chkno>354</chkno>
    <date>01/10/2002</date>
    <amt> 122.30</amt>
  </transaction>
  <transaction>
    <chkno>355</chkno>
    <date>01/11/2002</date>
    <amt>  59.95</amt>
  </transaction>
  <transaction>
    <chkno>356</chkno>
    <date>01/15/2002</date>
    <amt> 852.33</amt>
  </transaction>
  <transaction>
    <chkno>357</chkno>
    <date>01/30/2002</date>
    <amt> 500.35</amt>
  </transaction>
</checks>

<balances>
  <baldata>
    <date>01/04/2002</date>
    <bal>2269.74</bal>
  </baldata>
  <baldata>
    <date>01/05/2002</date>
    <bal>2196.64</bal>
  </baldata>
  <baldata>
    <date>01/10/2002</date>
    <bal>2074.34</bal>
  </baldata>
  <baldata>
    <date>01/11/2002</date>
    <bal>2014.39</bal>
  </baldata>
</balances>

Figure 63. XML Data File (Part 2 of 5)
<bankstatement>
  <customer>
    <acctno>887-278342</acctno>
    <name>Anna Merkin</name>
    <street>123 Chantilly Lane</street>
    <cityst>Long Neck Goose, VA 21177</cityst>
  </customer>
  <begindate>JAN 02, 2002</begindate>
  <enddate>FEB 01, 2002</enddate>
  <!--<!-- Page number generator -->
  <pagenumber>
    <!-- New account type = Super Checking Account -->
    <supercheckingactivity="suprchk">
      <balance>
        <begin>3722.23</begin>
        <credit>2084.58</credit>
        <debit>1908.94</debit>
        <svchg>0.00</svchg>
        <end>3897.87</end>
      </balance>
      <credits>
        <transaction>
          <type>DEPOSIT</type>
          <date>01/11/2002</date>
          <amt> 17.37</amt>
        </transaction>
        <transaction>
          <type>AUTO DEPOSIT</type>
          <date>01/15/2002</date>
          <amt>1029.81</amt>
        </transaction>
        <transaction>
          <type>AUTO DEPOSIT</type>
          <date>01/30/2002</date>
          <amt>1029.81</amt>
        </transaction>
      </credits>
    </supercheckingactivity>
  </pagenumber>
</bankstatement>

Figure 64. XML Data File (Part 3 of 5)
<transaction>
  <type>INTEREST</type>
  <date>01/31/2002</date>
  <amt>7.59</amt>
</transaction>
<total>2084.58</total>
</credits>
<checks>
  <transaction>
    <chkno>759</chkno>
    <date>01/03/2002</date>
    <amt>144.00</amt>
  </transaction>
  <transaction>
    <chkno>760</chkno>
    <date>01/04/2002</date>
    <amt>93.11</amt>
  </transaction>
  <transaction>
    <chkno>761</chkno>
    <date>01/09/2002</date>
    <amt>322.72</amt>
  </transaction>
  <transaction>
    <chkno>762</chkno>
    <date>01/11/2002</date>
    <amt>102.43</amt>
  </transaction>
  <transaction>
    <chkno>763</chkno>
    <date>01/17/2002</date>
    <amt>794.46</amt>
  </transaction>
  <transaction>
    <chkno>764</chkno>
    <date>01/29/2002</date>
    <amt>452.22</amt>
  </transaction>
</checks>

Figure 65. XML Data File (Part 4 of 5)
Figure 66. XML Data File (Part 5 of 5)

<balances>
  <baldata>
    <date>01/04/2002</date>
    <bal>3722.23</bal>
  </baldata>
  <baldata>
    <date>01/05/2002</date>
    <bal>3629.12</bal>
  </baldata>
  <baldata>
    <date>01/10/2002</date>
    <bal>3306.40</bal>
  </baldata>
  <baldata>
    <date>01/11/2002</date>
    <bal>3221.34</bal>
  </baldata>
  <baldata>
    <date>01/15/2002</date>
    <bal>4251.15</bal>
  </baldata>
  <baldata>
    <date>01/30/2002</date>
    <bal>3897.87</bal>
  </baldata>
  <total>3897.87</total>
</balances>

Figure 67 on page 98 - Figure 68 on page 99 shows the resulting printed output from the XML data in Figure 62 on page 93 - Figure 66.
Big Brother Bank
“We watch over you”
P.O. Box 1573
Beantown, MA 02116

Justin Case
123 Redlight Lane
TwistNshout, MA 02345

Super Checking Account Activity

<table>
<thead>
<tr>
<th>Beginning Balance</th>
<th>Credits</th>
<th>Debits</th>
<th>Service Charge</th>
<th>Ending Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>2591.24</td>
<td>1946.93</td>
<td>1956.43</td>
<td>0.00</td>
<td>2581.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Credits</th>
<th>Description</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPOSIT</td>
<td></td>
<td>01/05/02</td>
<td>26.90</td>
</tr>
<tr>
<td>AUTO DEPOSIT</td>
<td></td>
<td>01/15/02</td>
<td>954.27</td>
</tr>
<tr>
<td>AUTO DEPOSIT</td>
<td></td>
<td>01/30/02</td>
<td>954.27</td>
</tr>
<tr>
<td>INTEREST</td>
<td></td>
<td>01/31/02</td>
<td>11.49</td>
</tr>
<tr>
<td>Total Credits</td>
<td></td>
<td></td>
<td>1946.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checks</th>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>352</td>
<td>01/04/02</td>
<td>321.50</td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>01/05/02</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>354</td>
<td>01/10/02</td>
<td>122.30</td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>01/11/02</td>
<td>59.95</td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>01/15/02</td>
<td>852.33</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>01/30/02</td>
<td>500.35</td>
<td></td>
</tr>
<tr>
<td>Total Checks</td>
<td></td>
<td></td>
<td>1956.43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Daily Balances</th>
<th>Date</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01/04/02</td>
<td>2269.74</td>
</tr>
<tr>
<td></td>
<td>01/05/02</td>
<td>2196.64</td>
</tr>
<tr>
<td></td>
<td>01/10/02</td>
<td>2074.34</td>
</tr>
<tr>
<td></td>
<td>01/11/02</td>
<td>2014.39</td>
</tr>
<tr>
<td></td>
<td>01/15/02</td>
<td>852.33</td>
</tr>
<tr>
<td></td>
<td>01/30/02</td>
<td>500.35</td>
</tr>
<tr>
<td>Final Balance</td>
<td></td>
<td>2581.74</td>
</tr>
</tbody>
</table>

Interest Rate as of 01/04 * * 5.321%

Page 1
## Super Checking Account Activity

<table>
<thead>
<tr>
<th>Beginning Balance</th>
<th>Credits</th>
<th>Debits</th>
<th>Service Charge</th>
<th>Ending Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>3722.23</td>
<td>2084.58</td>
<td>1908.94</td>
<td>0.00</td>
<td>3897.87</td>
</tr>
</tbody>
</table>

### Credits

<table>
<thead>
<tr>
<th>Description</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPOSIT</td>
<td>01/11/02</td>
<td>17.37</td>
</tr>
<tr>
<td>AUTO DEPOSIT</td>
<td>01/15/02</td>
<td>1029.81</td>
</tr>
<tr>
<td>AUTO DEPOSIT</td>
<td>01/30/02</td>
<td>1029.81</td>
</tr>
<tr>
<td>INTEREST</td>
<td>01/31/02</td>
<td>7.59</td>
</tr>
</tbody>
</table>

**Total Credits**

<table>
<thead>
<tr>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>2084.58</td>
</tr>
</tbody>
</table>

### Checks

<table>
<thead>
<tr>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>759</td>
<td>01/03/02</td>
<td>144.00</td>
</tr>
<tr>
<td>760</td>
<td>01/04/02</td>
<td>93.11</td>
</tr>
<tr>
<td>761</td>
<td>01/09/02</td>
<td>322.72</td>
</tr>
<tr>
<td>762</td>
<td>01/11/02</td>
<td>102.43</td>
</tr>
<tr>
<td>763</td>
<td>01/17/02</td>
<td>794.46</td>
</tr>
<tr>
<td>764</td>
<td>01/29/02</td>
<td>452.22</td>
</tr>
</tbody>
</table>

**Total Checks**

<table>
<thead>
<tr>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1908.94</td>
</tr>
</tbody>
</table>

### Daily Balances

<table>
<thead>
<tr>
<th>Date</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/04/02</td>
<td>3722.23</td>
</tr>
<tr>
<td>01/05/02</td>
<td>3629.12</td>
</tr>
<tr>
<td>01/10/02</td>
<td>3306.40</td>
</tr>
<tr>
<td>01/11/02</td>
<td>3221.34</td>
</tr>
<tr>
<td>01/15/02</td>
<td>4251.15</td>
</tr>
<tr>
<td>01/30/02</td>
<td>3897.87</td>
</tr>
</tbody>
</table>

**Final Balance**

<table>
<thead>
<tr>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>3897.87</td>
</tr>
</tbody>
</table>

*Interest Rate as of 01/04 * * * 5.321%
PAGEDEF bbbank replace yes
WIDTH 8.5 in
HEIGHT 11.0 in
UDTYPE EBCDIC;
FONT comp a075nc TYPE EBCDIC; /*Big Brother Bank font */
FONT italic a175dc TYPE EBCDIC; /*Italic theme */
FONT addr a075dc TYPE EBCDIC; /*Big Brother address */
FONT varb gt10 TYPE EBCDIC; /*Variable data */
FONT super a075dc TYPE EBCDIC; /*Super Checking Account */
FONT head a055ac TYPE EBCDIC; /*Headings */
FONT bhead a075ac TYPE EBCDIC; /*Bold Headings */
/*****************************/
/** QTAG declarations **/
/*****************************/
/***** statmid declarations ----------------------*/
DEFINE statmid QTAG C'document',
C'bankstatement',C'customer';
DEFINE acctno QTAG C'document',
C'bankstatement',C'customer',C'acctno';
DEFINE name QTAG C'document',
C'bankstatement',C'customer',C'name';
DEFINE street QTAG C'document',
C'bankstatement',C'customer',C'street';
DEFINE cityst QTAG C'document',
C'bankstatement',C'customer',C'cityst';
DEFINE begindate QTAG C'document',
C'bankstatement',C'begindate';
DEFINE enddate QTAG C'document',
C'bankstatement',C'enddate';
/***** statsum declarations ----------------------*/
DEFINE statsum QTAG C'document',
C'bankstatement',C'supercheckingactivity' C'balance' ;
DEFINE statsumf1 QTAG C'document',
C'bankstatement',C'supercheckingactivity' C'balance', C'begin';
DEFINE statsumf2 QTAG C'document',
C'bankstatement',C'supercheckingactivity' C'balance', C'credit';
DEFINE statsumf3 QTAG C'document',
C'bankstatement',C'supercheckingactivity' C'balance', C'debit';
DEFINE statsumf4 QTAG C'document',
C'bankstatement',C'supercheckingactivity' C'balance', C'svchg';
DEFINE statsumf5 QTAG C'document',
C'bankstatement',C'supercheckingactivity' C'balance', C'end';
/***** crdata declarations ----------------------*/
DEFINE crheader QTAG C'document',
C'bankstatement',C'supercheckingactivity',
C'credits';
DEFINE crdata1 QTAG C'document',
C'bankstatement',C'supercheckingactivity',
C'credits', C'transaction', C'type';
DEFINE crdata2 QTAG C'document',
C'bankstatement',C'supercheckingactivity',
C'credits', C'transaction', C'date';

Figure 69. Page Definition for XML Output (Part 1 of 7)
DEFINE crdata3 QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'credits',C'transaction',C'amt' ;
DEFINE crtotal QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'credits',C'total' ;

/**** ckdata declarations ------------------------*/
DEFINE ckheader QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'checks' ;
DEFINE ckdata1 QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'checks',C'transaction',C'chkno' ;
DEFINE ckdata2 QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'checks',C'transaction',C'date' ;
DEFINE ckdata3 QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'checks',C'transaction',C'amt' ;
DEFINE cktotal QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'checks', C'total' ;

/**** baldata declarations ----------------------*/
DEFINE balhead QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'balaes' ;
DEFINE baldata1 QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'balances', C'baldata', C'date' ;
DEFINE baldata2 QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'balances', C'baldata', C'bal' ;
DEFINE baltotal QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'balances', C'total' ;

/**** misc. declarations -----------------------*/
DEFINE statrail QTAG C'document',
    C'bankstatement',C'supercheckingactivity',
    C'stmtrailer' ;
DEFINE pgenum QTAG C'document',
    C'bankstatement',C'pagenumber' ;
/***** end of QTAG declarations --------------*/
/***** end of PAGEFORMAT declarations */
PAGEFORMAT xhub1 TOPMARGIN 2 in BOTMARGIN 1 in;
/*****************************/

Figure 70. Page Definition for XML Output (Part 2 of 7)
Figure 71. Page Definition for XML Output (Part 3 of 7)
FIELD TEXT C'Debits'
    POSITION 3.6 in CURRENT
    FONT head ; /* Static text - first header */
FIELD TEXT C'Service Charge'
    POSITION 4.8 in CURRENT
    FONT head ; /* Static text - first header */
FIELD TEXT C'Ending Balance'
    POSITION 6.3 in CURRENT
    FONT head ; /* Static text - first header */
XLAYOUT statsumf1 BODY
    POSITION SAME .6 in;
FIELD START 1 LENGTH 8
    POSITION .6 in CURRENT
    FONT varb ; /* Variable text - Beg balance */
XLAYOUT statsumf2 BODY
    POSITION SAME SAME;
FIELD START 1 LENGTH 8
    POSITION 2.2 in CURRENT
    FONT varb ; /* Variable text - Credits */
XLAYOUT statsumf3 BODY
    POSITION SAME SAME;
FIELD START 1 LENGTH 8
    POSITION 3.4 in CURRENT
    FONT varb ; /* Variable text - Debits */
XLAYOUT statsumf4 BODY
    POSITION SAME SAME;
FIELD START 1 LENGTH 5
    POSITION 5.0 in CURRENT
    FONT varb ; /* Variable text - Service Chrg */
XLAYOUT statsumf5 BODY
    POSITION SAME SAME;
FIELD START 1 LENGTH 8
    POSITION 6.5 in CURRENT
    FONT varb ; /* Variable text - End Balance */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
    POSITION 0 .1 in;
/************************************************/
/** crheader GROUPHEADER  **/
/****************************************************************************/
XLAYOUT crheader GRPHEADER XSPACE .2 in
    POSITION SAME .3 in;
FIELD TEXT C'Credits'
    FONT bhead ; /* Static text - Credits */
FIELD TEXT C'Description'
    POSITION 1.3 in CURRENT
    FONT head ; /* Stat text - Deposit Descr. */
FIELD TEXT C'Date'
    POSITION 3.2 in CURRENT
    FONT head ; /* Static text - Date */
FIELD TEXT C'Amount'
    POSITION 5.0 in CURRENT
    FONT head ; /* Stat text - Amount of deposit*/
DRAWGRAPHIC LINE ACROSS 6.2 IN LINEWT BOLD
    POSITION 1.3 in next;
/****************************************************************************/
/** crdata BODY  **/
/****************************************************************************/
XLAYOUT crdata1 BODY GROUP;
FIELD START 1 LENGTH 13
    POSITION 1.3 in CURRENT
    FONT varb ; /* Variable text - Description */

Figure 72. Page Definition for XML Output (Part 4 of 7)
Figure 73. Page Definition for XML Output (Part 5 of 7)
FIELD START 1 LENGTH 8 ALIGN RIGHT
  POSITION 7.3 in CURRENT
  FONT varb ; /* Variable text - Amount */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
  POSITION 0 next;
/*****************************************************************************/
/** balhead GROUPHEADER **/
/******************************************************************************/
XLOOP balhead GRPHEADER XSPACE .2 in
  POSITION SAME .6 in;
FIELD TEXT C'Daily'
  FONT bhead ; /* Static text - Daily Balance */
FIELD TEXT C'Date'
  POSITION 1.3 in CURRENT
  FONT head ; /* Stat text - Date of balance */
FIELD TEXT C'Balance'
  POSITION 3.15 in CURRENT
  FONT head ; /* Static text - Balance */
FIELD TEXT C'Balances'
  POSITION 0 NEXT
  FONT bhead ; /* Static text - Daily Balance */
DRAWGRAPHIC LINE ACROSS 6.2 IN LINEWT BOLD
  POSITION 1.3 in CPOS;
/******************************************************************************/
/*****************************************************************************/
/** baldata BODY **/
/******************************************************************************/
XLOOP baldata1 BODY GROUP
  POSITION SAME NEXT;
FIELD START 01 LENGTH 8
  POSITION 1.3 in CURRENT
  FONT varb ; /* Variable text - Date */
XLOOP baldata2 BODY GROUP position same same;
FIELD START 01 LENGTH 8 ALIGN RIGHT
  POSITION 3.8 in CURRENT
  FONT varb ; /* Variable text - Amount */
/******************************************************************************/
/*****************************************************************************/
/** baltotal BODY **/
/******************************************************************************/
XLOOP baltotal BODY GROUP;
FIELD TEXT C'Final Balance'
  POSITION 1.5 in .2 in
  FONT bhead ; /* Stat text - Final balance */
FIELD START 1 LENGTH 8 ALIGN RIGHT
  POSITION 7.3 in CURRENT
  FONT varb ; /* Variable text - Amount */
/******************************************************************************/
/*****************************************************************************/
/** statrail BODY **/
*******************************************************************************/
XLOOP statrail1 BODY
  POSITION SAME .4 in;
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
  POSITION 0 CPOS;
FIELD TEXT C'Interest Rate'
  POSITION 2.0 in NEXT
  FONT bhead ; /* Static text - Interest rate */
FIELD TEXT C'As of 01/04 * * * 5.321'%
  POSITION CURRENT CURRENT
  FONT varb ; /* Static text */
DRAWGRAPHIC LINE ACROSS 7.5 IN LINEWT BOLD
  POSITION 0 NEXT
  copy down 2 spaced 1 mm;

Figure 74. Page Definition for XML Output (Part 6 of 7)
XINCLUDE "xml/generic/pagenum.xquery"

/*
 * pgenum PAGE NUMBER
 */
	XLAYOUT pgenum PAGETRAILER
		POSITION SAME ABSOLUTE 10.7 in;
		FIELD TEXT C 'Page ' POSITION 6.5 in CURRENT
		FONT varb; /* placement of page number */
		FIELD PAGENUM PRINT /* request page numbering */
		FONT varb /* placement of page number */
		POSITION CURRENT CURRENT;

Figure 75. Page Definition for XML Output (Part 7 of 7)
Chapter 5. Creating Complex Printouts

You are now ready to learn about some formatting tasks that might apply to more complex printouts. You have learned in other chapters about the basic form definition and page definition elements. This chapter describes how these elements are combined to create complete print jobs.

The advanced techniques that are covered in this section are illustrated in the following examples:

Table 5. Form Definitions and Page Definition Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Example location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combine field processing with an overlay.</td>
<td>“Combining Field Processing and an Electronic Overlay”</td>
</tr>
<tr>
<td>Suppress data.</td>
<td>“UsingSuppressions to Vary Data Presentation” on page 109</td>
</tr>
<tr>
<td>Include fixed text.</td>
<td>“Incorporating Fixed Text into a Page Definition” on page 110</td>
</tr>
<tr>
<td>Combine two reports.</td>
<td>“Combining Two Reports into One Printout” on page 113</td>
</tr>
</tbody>
</table>

The examples in this chapter build on a single sales application, showing different sales reports being formatted by form definitions and page definitions.

Combining Field Processing and an Electronic Overlay

This example involves printing a monthly individual sales report for a specified distribution. The following items are needed to generate the sales report:

- A pre-designed electronic overlay for the sales report
- An unformatted print data file with periodic sales statistics

An example of the overlay and data file is shown in Figure 76 on page 108
The code example that follows contains a form definition and a page definition. The page definition maps the file to the overlay.

In Figure 76, the 0,0 point is the upper-left corner of the overlay, which means that the logical page origin must coincide with the overlay origin in this example. POSITION subcommands are relative to the logical page origin. The overlay origin point that positions the overlay is specified in the Overlay Generation Language/370 that creates the overlay, but can be modified in the page definition. In mapping to an overlay, you must check the input to the overlay creation program so you can coordinate its origin with the logical page origin. You can reposition the overlay through the PRINTLINE command.

```
01 FORMDEF SLSRPT OFFSET 0 0 ;
02   OVERLAY SLSRPT ;
03   SUBGROUP OVERLAY SLSRPT ;
04
05 PAGEDEF SLSRPT ;
06   PRINTLINE POSITION 2 IN 1.3 IN ; /* RECORD 1 */
07   FIELD START 1 LENGTH 23 ;
08   PRINTLINE POSITION 2 IN 1.70 IN ; /* RECORD 2 */
09   FIELD START 1 LENGTH 9 ; /* DEFAULT POSITION */
10   FIELD START 10 LENGTH 5
11       POSITION 4.3 IN * ; /* THE ASTERISK MEANS */
12       /* CURRENT LINE */
13   PRINTLINE POSITION 1.5 IN 6 IN ; /* RECORD 3 */
14   FIELD START 1 LENGTH 4 ;
15 SETUNITS LINESP 4 LPI ;
16 PRINTLINE REPEAT 4 /* RECORDS 4-7 */
17   FIELD START 1.5 IN 3.6 IN ;
18   FIELD START 1 LENGTH 7 ; /* DEFAULT POSITION */
19   FIELD START 10 LENGTH 3
```

Figure 76. Electronic Overlay and Data File for a Sales Report

The overlay is 6 in. x 4 in. and the data file is 12 in. x 8 in. The overlay and data file are shown in Figure 76.
A time-saving device is to use the REPEAT subcommand (line 16), which maps a single printline with its field subsets to records 4 - 7 with all model names and sales statistics. The length values in the repeated fields are 7, 3, 3, and 3, which are sufficient to accommodate the largest model name, unit value, $(000), and percentage fields mapped by this FIELD command.

Figure 77 shows the report that is formatted by the resources that are generated in the command stream of this example.

![Figure 77. Sales Report](image)

Using Suppressions to Vary Data Presentation

You use PPFA and your print server printers to produce variations of the same report in a single job. The essential function for this capability is called suppression. Suppression involves the coordinated specification of elements in both the page definition and the form definition. You create a suppression in the page definition and turn it on or off in a subgroup within a form definition.

This example shows how to alter the controls in the previous example (“Combining Field Processing and an Electronic Overlay” on page 107) to generate a second report along with the one already created.

First, change the page definition by adding a SUPPRESSION subcommand to the third field in the repeated PRINTLINE (the PRINTLINE that mapped the models and sales figures in “Combining Field Processing and an Electronic Overlay” on page 107). The suppression is, in effect, created by the SUPPRESSION subcommand in the FIELD command. The following example shows the addition at line 23.

```
18 FIELD START 1 LENGTH 7 ;
19 FIELD START 10 LENGTH 3
20 POSITION 1.5 IN * ;
21 FIELD START 16 LENGTH 3
22 POSITION 2.5 IN *
23 SUPPRESSION SALES ; /*ADDED LINE */
24 FIELD START 21 LENGTH 3
25 POSITION 3.5 IN * ;
```
The SUPPRESSION subcommand creates the potential for selective suppression of the data in the $(000)$ field of the report.

Then, rewrite the form definition, creating two subgroups within the copy group. Next, write a SUPPRESSION command immediately after the FORMDEF command. Finally, place a SUPPRESSION subcommand in the subgroup in which you want to suppress the data. The subcommand names the suppression. The resulting form definition command stream is as follows:

```
FORMDEF SECRPT ;
  SUPPRESSION SALES ; /*NAMING THE SUPPRESSION */
  COPYGROUP SECRPT ;
  OVERLAY SLSRPT ; /*NAMING THE OVERLAY */
  SUBGROUP COPIES 1
    OVERLAY SLSRPT ;
  SUBGROUP COPIES 1
    OVERLAY SLSRPT
    SUPPRESSION SALES ; /*TURNING ON THE SUPPRESSION */
```

The result is shown in Figure 78. The second subgroup creates the second output page of the same data with a second set of modifications; in this case, modifications means a suppression that is not in the first subgroup.

```
<table>
<thead>
<tr>
<th>SALESMAN</th>
<th>John Smith</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERRITORY</td>
<td>Texas</td>
</tr>
<tr>
<td>NO.</td>
<td>07714</td>
</tr>
<tr>
<td>MONTH</td>
<td>Nov.</td>
</tr>
<tr>
<td>MODEL</td>
<td>UNITS</td>
</tr>
<tr>
<td>Sierra</td>
<td>12</td>
</tr>
<tr>
<td>Otero</td>
<td>16</td>
</tr>
<tr>
<td>Agua</td>
<td>60</td>
</tr>
<tr>
<td>Allegre</td>
<td>71</td>
</tr>
<tr>
<td>MODEL</td>
<td>UNITS</td>
</tr>
<tr>
<td>Sierra</td>
<td>12</td>
</tr>
<tr>
<td>Otero</td>
<td>16</td>
</tr>
<tr>
<td>Agua</td>
<td>60</td>
</tr>
<tr>
<td>Allegre</td>
<td>71</td>
</tr>
</tbody>
</table>
```

Figure 78. Selective Suppression

Review the steps in this example. To suppress a field, identify the field as suppressible in the page definition under the FIELD command in question. Then, create a subgroup, activating this suppression with a SUPPRESSION subcommand in the form definition.

The first subgroup produces an output identical to the report in “Combining Field Processing and an Electronic Overlay” on page 107. It contains no suppression.

Note: This example can be printed in simplex only.

### Incorporating Fixed Text into a Page Definition

Fixed text can be incorporated into an electronic overlay by using programs, such as Overlay Generation Language/370. By having another place (the page definition) to incorporate fixed text, you can format documents more efficiently.

In “Combining Field Processing and an Electronic Overlay” on page 107, a territory sales report for salesman John Smith is created. Here, the territory sales report is incorporated into a larger format that goes to ACME’s corporate
headquarters in Chicago. Therefore, the identification for the region needs to appear on the report form. An overlay is used as a header for the composite report, which means that two overlays appear in the command stream: one carries over from "Combining Field Processing and an Electronic Overlay" on page 107 and the other is the header.

So, as shown in Figure 79, three fixed inputs generate the final report: overlay SLSRPT, overlay HDR, and the fixed regional identification text. (It is the second item that is worked into the page definition in this example.)

<table>
<thead>
<tr>
<th>SALESMAN</th>
<th>NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERRITORY</td>
<td></td>
</tr>
<tr>
<td>MONTH</td>
<td></td>
</tr>
<tr>
<td>MODEL</td>
<td>UNITS</td>
</tr>
</tbody>
</table>

Overlay SLSRPT

<table>
<thead>
<tr>
<th>INDIVIDUAL SALES REPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACME CORP. - CHICAGO</td>
</tr>
</tbody>
</table>

Regional Mgrs. Submit
First Monday in Each Month

Overlay HDR

Southwest Region
Jim Jones - Manager

Fixed Text

Figure 79. Input for the Corporate Version of an Individual Sales Report

The data file used to generate this report is the same as the one shown in Figure 76 on page 108.

FORMDEF CORP ;
OVERLAY SLSRPT ;
OVERLAY HDR ;
SUBGROUP OVERLAY SLSRPT HDR ;
PAGEDEF CORP
  WIDTH 6 IN
  HEIGHT 7 IN ;
  PRINTLINE POSITION 1.9 IN 2.5 IN ; /*RECORD 1 */
  FIELD TEXT C 'SOUTHWEST REGION' ; /*DEFAULT FIELD TEXT */
  /*POSITION */
  FIELD POSITION -.2 IN NEXT /*NOTE NEGATIVE VALUE */
  TEXT 1 C 'JIM JONES - MANAGER' ;
  FIELD START 1 LENGTH 23
  POSITION .1 IN .8 IN ;
  PRINTLINE POSITION 2 IN 3.7 IN ; /*RECORD 2 */
The same basic commands from “Combining Field Processing and an Electronic Overlay” on page 107 are used in the command stream, although the positions of fields are changed to accommodate the new layout.

New FIELD commands with TEXT subcommands are inserted in the first PRINTLINE command to produce the regional text, which is positioned at the bottom of the header form. The 1 is a duplication parameter that indicates how many times the fixed text is to be repeated. The C can precede single-byte characters, such as those that are used to write English or German. Both 1 and C are the default values for a TEXT subcommand. The text that you want inserted appears between single quotation marks. Observe how the POSITION subcommands change to accommodate both fixed text and record-1 text.

**Note:** Each PRINTLINE command in your PPFA command stream must have a corresponding record in the input data file. If you specify a fixed-text data field and an input data field under the same PRINTLINE command, they are both associated with the same input data file record. However, if all the FIELD commands under a PRINTLINE command specify fixed text, the corresponding input record is discarded. In that case, you must insert a blank record into the input data file to preserve the correct relationship between records and PRINTLINE commands.

Figure 80 on page 113 shows how the finished output looks.
Combining Two Reports into One Printout

This example combines two data files and two page layouts into one printout, also building on "Combining Field Processing and an Electronic Overlay" on page 107. Figure 81 on page 114 shows the new data and a new overlay.
This command stream is needed to generate both pages of the preceding report:

```
FORMDEF SLSCOM ;
COPYGROUP SLSRPT ;
OVERLAY SLSRPT ;
SUBGROUP OVERLAY SLSRPT ;
COPYGROUP COMRPT ;
OVERLAY COMRPT ;
SUBGROUP OVERLAY COMRPT ;
PAGDEF SLSCOM ;
FONT M104 ;
FONT M105 ;
PAGEFORMAT SLSRPT ; /*SALES REPORT*/
PRINTLINE FONT M104
   POSITION 2 IN .5 IN ;
   FIELD START 1 LENGTH 23 ;
PRINTLINE POSITION 2 IN .75 IN ;
   FIELD START 1 LENGTH 9 ; /*DEFAULT FIELD POSITION*/
   FIELD START 10 LENGTH 5
   POSITION 2.3 IN * ;
PRINTLINE POSITION 1.5 IN 1 IN ;
   FIELD START 1 LENGTH 4 ;
PRINTLINE REPEAT 4
   POSITION .3 IN 1.8 IN ;
   FIELD START 1 LENGTH 7 ; /*DEFAULT FIELD POSITION */
   FIELD START 11 LENGTH 3
   POSITION 1.5 IN * ;
   FIELD START 16 LENGTH 3
   POSITION 3 IN * ;
```

*Figure 81. Input for a New Report Produced from the Combined Data Files*
Although requiring a complex series of commands, the following commission report is handled much like any other field processing problem: the data must be carefully mapped into the overlay exactly where it is wanted. If you change copy groups and page formats, both the Invoke Medium Map structured field and the Invoke Data Map structured field must be inserted into the data file where you want the changes. Here they occur together.

Figure 82 shows both the commission report and the sales report. With page printers and with careful data positioning, such reports look like they were individually prepared with no differences in the presentation of the fixed data.

<table>
<thead>
<tr>
<th>SALESMAN</th>
<th>John Smith</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERRITORY</td>
<td>Texas</td>
</tr>
<tr>
<td>NO.</td>
<td>07714</td>
</tr>
<tr>
<td>MONTH</td>
<td>Nov.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MODEL</th>
<th>UNITS</th>
<th>$(000)</th>
<th>% of TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sierra</td>
<td>12</td>
<td>59</td>
<td>6</td>
</tr>
<tr>
<td>Otero</td>
<td>16</td>
<td>70</td>
<td>10</td>
</tr>
<tr>
<td>Aqua</td>
<td>60</td>
<td>104</td>
<td>15</td>
</tr>
<tr>
<td>Allegre</td>
<td>71</td>
<td>265</td>
<td>40</td>
</tr>
</tbody>
</table>

Salesman’s Commission Report
Acme Corp.
To: John Smith

As your sales for Nov. 1995 were $498,000.00, and your commission rate is 2%, your monthly commission is $9960.00. Should there be any question, please contact Al Jankowski in accounting.

Sincerely,
Acme Corp.

Figure 82. The Sales and the Commission Reports
Chapter 6. Conditional Processing

You use conditional processing to test fields within an input line data record (for example, a customer number). Based on the results of the test, you can specify the action to be taken, such as to change copy groups or page formats. This chapter provides:

- An explanation of how conditional processing works
- A detailed list of rules, restrictions, and considerations
- Examples showing how conditional processing can be used to do some commonly requested functions

General Description

With conditional processing, you can:

- Test the input data by using the CONDITION command.
- Choose the copy group and page format to be used when you are printing the data.
- Change to a different copy group or page format after the data is read. You can specify that the new copy group or page format is used:
  - Before the current subpage is printed.
  - Before the current line is printed.
  - After the current line is printed.
  - After the current subpage is printed.

Table 6 shows the tasks that you can do with conditional processing.

<table>
<thead>
<tr>
<th>Task</th>
<th>Location of the Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack offset from previous jobs.</td>
<td>“Jog Output Example” on page 130</td>
</tr>
<tr>
<td>Use different print directions for front and back sides of a sheet.</td>
<td>“Duplex Output with Different Front and Back Print Directions” on page 130</td>
</tr>
<tr>
<td>Do record reprocessing.</td>
<td>“Record Reprocessing Example” on page 131</td>
</tr>
<tr>
<td>Select different paper sources.</td>
<td>“Selecting Paper from an Alternate Bin Example” on page 132</td>
</tr>
<tr>
<td>Use multiple CONDITION commands.</td>
<td>“Multiple CONDITION Commands” on page 133</td>
</tr>
<tr>
<td>Use repeat PRINTLINE commands.</td>
<td>“Field Processing When PRINTLINEs Are Repeated” on page 136</td>
</tr>
</tbody>
</table>

Using Conditional Processing versus Normal Line Data Processing

Normal line-data processing consists of:

- Setting up the physical page environment by defining a copy group
- Setting up the logical page environment by defining a page format

Input records correspond to PRINTLINE commands that determine such things as where the input records are to be printed, which font to use and what print direction to use. Only one copy group and page format can be used for processing each input record.
You can use conditional processing as a preprocessor to test the input data before you decide which copy group and page format to use. Furthermore, you can change these specifications based on changes in the input data. Except for record reprocessing (see “Record Reprocessing Description and Processing” on page 121), when the copy group and page-format specifications are made, conditional processing operates the same as normal line-data processing.

**Note:** The copy group and page format can also be changed by placing Advanced Function Presentation data stream (AFP data stream) Invoke Medium Map (IMM) and Invoke Data Map (IDM) structured fields in the input data. Use of these structured fields within the input print file causes results that differ from what is described in this section. See *Mixed Object Document Content Architecture Reference* for information about these structured fields.

### Using Conditional Processing to Set Up the Environment

Setting up the environment consists of selecting a copy group and a page format.

#### Selecting a Copy Group

Conditional processing can be used to *select* a copy group; it does not *process* the copy group.

As described in Chapter 2, “Using Form Definition Commands,” on page 19, a form definition contains the controls that govern the physical page on which the print file is to be printed. A form definition can contain one or more copy groups as shown in the following diagram.

<table>
<thead>
<tr>
<th>PPFA Commands</th>
<th>Resulting Form Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMDEF FDEFX</td>
<td>F1FDEFX</td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>COPYGROUP CGA</td>
<td>CGA</td>
</tr>
<tr>
<td>.</td>
<td>CGB</td>
</tr>
<tr>
<td>.</td>
<td>CGC</td>
</tr>
<tr>
<td>OVERLAY ...</td>
<td></td>
</tr>
<tr>
<td>SUBGROUP ...</td>
<td></td>
</tr>
<tr>
<td>COPYGROUP CGB</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>OVERLAY ...</td>
<td></td>
</tr>
<tr>
<td>SUBGROUP ...</td>
<td></td>
</tr>
<tr>
<td>COPYGROUP CGC</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>OVERLAY ...</td>
<td></td>
</tr>
<tr>
<td>SUBGROUP ...</td>
<td></td>
</tr>
</tbody>
</table>

The first copy group within a form definition is always active when print file processing begins. To select a different copy group, use the CONDITION command.

**Note:** By using the BEFORE SUBPAGE and BEFORE LINE parameters with conditional processing, you can change to a different active copy group before any lines are formatted.
Using the previous diagram as a reference, assume that copy group CGB is active. The copy-group selections that can be made from a CONDITION command are:

- **condname**
  Starts the named copy group.

- **CURRENT**
  Restarts copy group CGB.

- **=**
  Restarts copy group CGB (alternative for CURRENT).

- **NEXT**
  Starts copy group CGC.

- **FIRST**
  Starts copy group CGA.

- **NULL**
  Does not change the current copy group processing.

- **/**
  Does not change the current copy group processing (alternative for NULL).

See “Using the CONDITION Command to Select a Copy Group and a Page Format” on page 128 for more information about each option.

### Selecting a Page Format
Conditional processing can be used to select an active page format. Selecting the page format does not change the basic rules for processing a page format:

- PRINTLINE commands are processed sequentially unless skip-to-channel or spacing commands are used.
- When the end of the page format is reached, processing returns to the first PRINTLINE command in the same page format. Processing does not continue with the next page format (if any) in the page definition.

However, conditional processing does involve some additional considerations:

#### Subpages
A page format consists of one or more subpages. A subpage is defined by a group of PRINTLINE commands followed by an ENDSUBPAGE command. If an ENDSUBPAGE command is not defined, then the entire page format is one subpage. See “Subpage Description and Processing” on page 121 for more information.

#### Record reprocessing
Record reprocessing is used when input records are processed according to one set of copy-group and page-format specifications, and then new specifications are invoked for the same input records. See “Record Reprocessing Description and Processing” on page 121 for more information.

As described in Chapter 3, “Using Page Definition Commands for Traditional Line Data,” on page 33, a page definition is a set of controls for formatting line-data files for printing on a logical page. A page definition can contain one or more page formats as shown in the following diagram.
The first page format in the page definition is always active when print file processing begins. To invoke a new page format, use the CONDITION command.

**Note:** By using the BEFORE SUBPAGE and BEFORE LINE parameters, it is possible to change to a different active page format before any lines are formatted.

Using the previous diagram as a reference, assume that page format PFMTB is active. The page-format selections that can be made from a CONDITION command are:

- `condname` Starts the named page format.
- `CURRENT` Restarts page format PFMTB.
- `=` Restarts page format PFMTB (alternative for CURRENT).
- `NEXT` Starts page format PFMTC.
- `FIRST` Starts page format PFMTA.
- `NULL` Does not change the current page format processing.
- `/` Does not change the current page format processing (alternative for NULL).

See "Using the CONDITION Command to Select a Copy Group and a Page Format" on page 128 for more information about each option.
Subpage Description and Processing

A page format consists of one or more subpages. A subpage is defined by a group of PRINTLINE commands followed by an ENDSUBPAGE command. If an ENDSUBPAGE command is not defined, then the entire page format is one subpage. The following considerations apply to subpages:

- Subpages are necessary only with conditional processing. Multiple-up printing can be done with or without defining them as subpages, but to change the page format or copy group at the level of one of the multiple-up pages, the multiple-up pages must be defined as subpages. In the following diagram, pages 1 - 4 can be defined as four separate subpages within one page format, or all defined within one subpage. However, to present the data on page 3 (for example) in a format different from the format that is used for pages 1 and 2, the four pages must be defined as subpages.

![Diagram showing page format with subpages](image)

- A subpage is processed sequentially starting from the beginning of the page format. Moving from one subpage to the next subpage is done by processing all the PRINTLINE commands for a specified subpage, or by skipping (with the CHANNEL subcommand) or spacing to a PRINTLINE command in a different subpage.

Note: Conditional processing cannot be used to select a subpage except by default. When a page format is started (or the current one is restarted), processing begins with the first PRINTLINE command of the page format. The effect is to select the first subpage in the page format.

Record Reprocessing Description and Processing

Record reprocessing is used when input records are processed according to one set of copy group and page format specifications, and then new specifications are invoked for the same input records. If the new specifications are to be applied with either the BEFORE SUBPAGE or the BEFORE LINE parameter, then the input records must be processed again with the new specifications instead of the original ones.
Note: Input records are not printed twice; record reprocessing just changes the specifications that are used when formatting the records.

The process is shown in the following diagram.

<table>
<thead>
<tr>
<th>PPFA Commands</th>
<th>Input Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGEFORMAT PFMTA ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE POSITION 1 IN 1 IN DIRECTION ACROSS REPEAT 5 ;</td>
<td></td>
</tr>
<tr>
<td>CONDITION cond1 START 2 LENGTH 1 WHEN EQ 'B' BEFORE SUBPAGE NULL PAGEFORMAT PFMTB ;</td>
<td>A B</td>
</tr>
<tr>
<td>PAGEFORMAT PFMTB ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE POSITION 7 IN 1 IN DIRECTION DOWN REPEAT 5 ;</td>
<td>A</td>
</tr>
<tr>
<td>CONDITION cond2 START 4 LENGTH 1 WHEN EQ 'Y' BEFORE SUBPAGE NULL PAGEFORMAT PFMTA ;</td>
<td>A</td>
</tr>
</tbody>
</table>

Assume that page format PFMTA is active. Under normal processing, the first input record would print in the ACROSS direction, starting at a horizontal offset of 1 inch and a vertical offset of 1 inch. However, the third record satisfies the CONDITION statement and causes a new page format (PFMTB) to be started. Since CONDITION cond1 specifies BEFORE SUBPAGE, the first two records must be reprocessed by using page format PFMTB. As a result, all of the records are printed in a DOWN direction, starting at a horizontal offset of 7 inches and a vertical offset of 1 inch.

If allowed to operate without restrictions, record reprocessing might force PSF into an infinite loop. For example:
As in the previous example, page format PFMTA is initially active, and input record 3 results in the selection of page format PFMTB. However, page format PFMTB has a condition that checks position four for the character 'Y', which is satisfied by input record 5. Therefore, if no restrictions exist, page format PFMTA would again be selected, the input data would be reprocessed (starting with input record 1), leading to an infinite loop.

To prevent this situation, after a BEFORE condition is satisfied, all other BEFORE conditions are ignored until data is formatted. See "Record Reprocessing" on page 124 for detailed information on this restriction.

### Conditional Processing Rules, Restrictions, and Considerations

This section describes the rules, restrictions, and considerations for conditional processing in these situations:

- Multiple conditions
- Record reprocessing
- Interaction between a CONDITION command and a REPEAT subcommand
- Interaction between the CONDITION command and the CHANNEL subcommand
- When CHANGE is always false at start of a page format
- CC and TRC fields relationship to the START subcommand
- Using the CONDITION command to select a copy group and a page format
- Variable length records and the CONDITION command
- Truncation of blanks and the CONDITION command
Multiple Conditions

Conditional processing supports:
- Multiple PRINTLINE commands in each subpage
- Multiple CONDITION commands on one PRINTLINE command
- Multiple WHEN statements on one CONDITION command

Rule
For all these situations, the rule is the same; the first true condition is the one processed, and any following true conditions are ignored.

Conditional Processing Considerations
Conditions are evaluated when they are encountered. For example, if a true condition is not detected when an OTHERWISE statement is encountered, the OTHERWISE statement always results in a true condition. (An exception to this consideration is explained in “Interaction Between the CONDITION Command and the CHANNEL Subcommand” on page 125.)

See “Multiple CONDITION Commands” on page 133 for an example of multiple CONDITION commands.

Record Reprocessing

Conditional Processing Restrictions
To prevent an infinite program loop, be aware that the following restrictions apply:
1. When the conditional action is to take place before the current subpage:
   a. Actions, which are specified as taking place before the current subpage, are shut off until the current subpage end.
   b. Actions, which are specified as taking place before the current line, are shut off for one line (the first line that is processed in the subpage).
2. When the conditional action is to take place before the current line, actions, which are specified as taking place before the current subpage or before the current line, are shut off for one line.

Considerations
- If a before subpage condition is true and causes a switch to a new page format, all before subpage conditions in the new page format are ignored.
- If a before line condition is true and causes a switch to a new page format, all before subpage and before line conditions in the new page format are ignored until one line is processed.

The consequence of this is that, after a true condition, at least one line must be processed before the next before condition is considered. This consequence can be confusing because a condition that would otherwise yield a true result can be ignored.

See “Record Reprocessing Example” on page 131 for an example of record reprocessing.

Interaction Between a CONDITION Command and a REPEAT Subcommand

See “Interaction Between the CONDITION Command and the CHANNEL Subcommand” on page 125 for what can appear to be an exception to the following rules.
Rule for a CONDITION Command and a REPEAT Subcommand

The REPEAT subcommand is used with the PRINTLINE command to specify the number of printlines (typically greater than one) that are to be constructed with the same specifications (such as font and direction). The CONDITION command is used to invoke conditional processing based on the data in a particular line. When the REPEAT and CONDITION commands are both specified for the same PRINTLINE command, every line that is described by the PRINTLINE command is checked for the specified condition until either the condition is satisfied or no more lines are described by the PRINTLINE command.

Note: This interaction is different from how the CHANNEL and POSITION subcommands interact with the PRINTLINE command. The CHANNEL and POSITION subcommands apply only to the first line described by the PRINTLINE command.

Rule for a CONDITION Command With an OTHERWISE Subcommand

The REPEAT subcommand is used with the PRINTLINE command to specify the number of printlines (typically greater than one) that are to be constructed with the same specifications (such as font and direction). The CONDITION command is used to invoke conditional processing based on the data in a particular line. The CONDITION command includes one or more WHEN subcommands and can include an OTHERWISE subcommand. If an OTHERWISE is coded, and none of the preceding WHEN conditions are true, the OTHERWISE condition is always true. If an OTHERWISE command is not coded, it is treated as a null.

Considerations

For the situation where REPEAT and CONDITION with OTHERWISE are coded for the same PRINTLINE command, the first input line determines the processing to be done. This situation happens because either one of the WHEN conditions or the OTHERWISE condition is always true for the first line.

Interaction Between the CONDITION Command and the CHANNEL Subcommand

Rule

A condition is checked if its associated PRINTLINE command is processed.

Note: ANSI carriage controls and machine (EBCDIC) carriage controls are processed differently. See the SPACE_THEN_PRINT subcommand in "Long Form Subcommands" on page 275 for more information.

ANSI A skip or space occurs before the line is printed.

Machine

The line is printed and then skipping or spacing is done.

For a CONDITION to be checked, it must be associated with the PRINTLINE command that is used for printing.

ANSI Skipping Consideration

The PRINTLINE command is not processed if a skip-to-channel-n character in the carriage control field causes the specified PRINTLINE command not to be processed.

If a data record contains a character, such as 1, in the carriage control field, and a PRINTLINE command is specified with CHANNEL 1 subcommand, the data
record is processed under the new PRINTLINE command (the one that specified CHANNEL 1). Any CONDITION associated with the old PRINTLINE command is ignored (never even checked). See the following diagram for an example.

The character 1 in the carriage-control field of the fifth input record causes a page end before condition \texttt{cond1} is ever checked. Thus, the fifth input record is processed by using the first PRINTLINE command of the current page format.

<table>
<thead>
<tr>
<th>PPFA Commands</th>
<th>Input Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGEFORMAT PFMTA ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE CHANNEL 1 ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>CONDITION \texttt{cond1}</td>
<td></td>
</tr>
<tr>
<td>START 6 LENGTH 1</td>
<td></td>
</tr>
<tr>
<td>WHEN EQ '{5}'</td>
<td></td>
</tr>
<tr>
<td>AFTER SUBPAGE</td>
<td></td>
</tr>
<tr>
<td>CURRENT NULL;</td>
<td></td>
</tr>
</tbody>
</table>
<pre><code>                                                             |
</code></pre>
<p>|                                                      | 1 2 3 4 5 6   |
|                                                      | LINE LINE LINE |
|                                                      | 1 2 3 4 5 6   |
|                                                      | LINE LINE LINE |
|                                                      | 1 2 3 4 5 6   |
|                                                      | LINE LINE LINE |
|                                                      | 1 2 3 4 5 6   |</p>

\textbf{Considerations}

The PRINTLINE command is not processed if the PRINTLINE command is spaced over; for example, when multiple line spacing causes certain PRINTLINE commands to be bypassed.

If the input-record carriage-control field specifies a double space before print (for example), and a CONDITION command is specified for the spaced line, the CONDITION is ignored (never checked). Because the OTHERWISE subcommand is part of a CONDITION command, the OTHERWISE subcommand is also ignored.

You might expect an OTHERWISE condition to be true if all other conditions failed. In fact, the OTHERWISE condition can be true if it is associated with a PRINTLINE command that is processed. See the following diagram for an example. This situation assumes that ANSI carriage controls are specified for this print file. ANSI carriage control 0 means space two lines before printing.

The fifth input record contains data (character 5 in the sixth position) that would normally satisfy the condition that is specified on the fifth PRINTLINE command. However, the character 0 in the carriage control field of input record 4 causes the fifth PRINTLINE command to be ignored. The fifth input record is processed by the sixth PRINTLINE command; therefore, the condition is not satisfied.
WHEN CHANGE is Always False at Start of a Page Format

Rule
The WHEN CHANGE process compares the contents of a specified field with the contents of the same field in the last record that was processed with the current page format and current condition. Whenever a page format is started (either by a condition that changes page formats or when processing of the data file begins), a WHEN CHANGE condition is always false because the previous record was not processed with the current page format.

Note: The following meanings apply to the terms in the previous paragraph:

changes
Switching to a page format that has a different name.

data file begins
If conditional processing invokes the CURRENT data map, CHANGE information is retained.

Considerations
Ensure that the WHEN CHANGE statement is processed before the switch to a new page format is done. See “WHEN CHANGE is Always False at Start of a Page Format” for an example of how a combination of WHEN CHANGE BEFORE SUBPAGE and WHEN CHANGE AFTER SUBPAGE can lead to unexpected results.

Relationship of CC and TRC fields to the START Subcommand

Rule
The position that is specified by the START subcommand of the CONDITION command is about the start of the data record. The first 1 or 2 bytes of an input record can contain either both a carriage-control character (CC) or a table-reference character (TRC). However, these characters are not considered part of the data record and are not to be counted when the START subcommand value is determined. In the following example, the field that is checked is the seventh character of the input record, but is the sixth character of the data record.
<table>
<thead>
<tr>
<th>PPFA Commands</th>
<th>Input Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGEFORMAT PFMTA ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE CHANNEL 1 ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
<tr>
<td>CONDITION cond1 START 6 LENGTH 1</td>
<td></td>
</tr>
<tr>
<td>WHEN EQ ‘5’ AFTER SUBPAGE CURRENT NULL;</td>
<td></td>
</tr>
<tr>
<td>PRINTLINE ;</td>
<td></td>
</tr>
</tbody>
</table>

Carriage Control

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>I</td>
<td>N</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>N</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>N</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>L</td>
<td>I</td>
<td>N</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>N</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>I</td>
<td>N</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using the CONDITION Command to Select a Copy Group and a Page Format

Rules

1. Within the CONDITION command, a copy group and a page format can be specified by using either a specific name or a parameter (CURRENT or =, FIRST, NEXT), NULL or / can be specified. The use of the NULL or / parameters differs from the use of the others:

   **Others**
   
   When any parameter other than NULL or / is specified, the specifications for the copy group or page format that is selected replace the current specifications. When the current specifications are replaced, the action is referred to as starting or restarting the copy group or page format. In AFP terminology, an Invoke Medium Map (IMM) command is generated for a copy group and an Invoke Data Map (IDM) command is generated for a page format.

   **NULL or /**
   
   When NULL or / is specified, no IMM or IDM is generated and processing continues as if no condition check was present.

2. The COPYGROUP and the PAGEFORMAT parameters are positional. If both parameters are specified, the COPYGROUP parameter must be first. If you want only to specify the copy group, the PAGEFORMAT parameter can be omitted, or specified as NULL or /. However, if you want only to specify the page format, the COPYGROUP parameter must be specified as NULL or /.

Considerations

- **Starting or Restarting a Copy Group**
  
  When a copy group is started (or restarted), the remaining input data is forced to the start on the next sheet. Therefore, if duplex output was expected, but the copy group is restarted while the front side of a sheet is processed, the remaining data starts on the front side of the next sheet rather than on the back side of the current sheet.

  See "Duplex Output with Different Front and Back Print Directions" on page 130 for an example.

  Furthermore, observe that any copy group action except NULL restarts the page format (see the following item).

- **Starting or Restarting a Page Format**
When a page format is started (or restarted), the remaining input data is forced to the start on the next side. Furthermore, that data is processed starting with the first PRINTLINE command in the specified page format. This situation is true even if CURRENT is specified as the page format parameter.

- Not Restarting a Copy Group

If the copy group is not to be restarted, specify NULL or /. Do not specify COPYGROUP NULL or COPYGROUP /. The following example illustrates this point. The command sequence in the Incorrect Format column invokes a copy group named NULL. The command sequence in the Correct Format column leaves the current copy group active.

<table>
<thead>
<tr>
<th>Incorrect Format</th>
<th>Correct Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDITION condname</td>
<td>CONDITION condname</td>
</tr>
<tr>
<td>START ...</td>
<td>START ...</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>WHEN ...</td>
<td>WHEN ...</td>
</tr>
<tr>
<td>COPYGROUP NULL</td>
<td>COPYGROUP NULL</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

- Not Restarting a Page Format

If the page format is not to be restarted, specify NULL or / (or omit the specification). Do not specify PAGEFORMAT NULL or PAGEFORMAT /.

The following example illustrates this point. The command sequence in the Incorrect Format column invokes a page format named NULL. The command sequence in the Correct Format column leaves the current page format active.

<table>
<thead>
<tr>
<th>Incorrect Format</th>
<th>Correct Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDITION condname</td>
<td>CONDITION condname</td>
</tr>
<tr>
<td>START ...</td>
<td>START ...</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>WHEN ...</td>
<td>WHEN ...</td>
</tr>
<tr>
<td>COPYGROUP CGA PAGEFORMAT NULL</td>
<td>COPYGROUP CGA</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

Variable Length Records and the CONDITION Command

Considerations

The CONDITION command inspects a field that starts at a particular position and extends for a certain length. If the entire field is not available within the input record, the condition is always false. If the input file contains variable-length records, the record might not extend the full length that is specified by the START and LENGTH subcommands. In this way, a condition that seems as if it is satisfied can fail.
Truncation of Blanks and the CONDITION Command

Considerations

Truncation occurs when blank characters are removed from the end of records on the spool. If blank truncation is in effect, the result can be the same as if the input file contained variable-length records.

Blank truncation is a consideration at the time the input records are passed to the print server. In the JES2 environment, blank truncation occurs unless the BLNKTRNC=NO parameter is specified. In the JES3 environment, blank truncation occurs unless the TRUNC=NO parameter is specified as part of either the BUFFER or SYSOUT initialization statements. Blank truncation can affect conditional processing because a field might disappear by being truncated and cause a WHEN/OTHERWISE clause not to run.

Conditional Processing Examples

This section provides conditional processing examples. The examples are grouped into functionally similar applications and are increasingly complex. The examples are:

- Jog output based on a change in the input data
- Duplex output with different front and back print directions
- Record reprocessing
- Select paper from an alternate bin
- Multiple CONDITION commands

Jog Output Example

Figure 83 shows how to jog the printed output, based on a change in the input data.

Copy group CGJOG specifies JG=YES. Page format PFJOG contains a CONDITION command that checks for any change in positions 8 - 10. If a change is detected, copy group CGJOG is restarted. Observe that the only result is to start printing on a new sheet and to jog that sheet.

```plaintext
FORMDEF TJOG;
COPYGROUP CGJOG JG=YES;

PAGEDEF TJOG;
PAGEFORMAT PFJOG WIDTH 11 IN HEIGHT 8.5;
PRINTLINE REPEAT 50
CHANNEL 1;
CONDITION NUPAGE START 8 LENGTH 3
WHEN CHANGE BEFORE SUBPAGE
COPYGROUP CGJOG;
```

Figure 83. Jog Output Example

Duplex Output with Different Front and Back Print Directions

Figure 84 on page 131 shows how to establish one print direction on the front side and a different print direction on the back side of a duplex sheet.

The page definition in this example contains two page formats, each of which has a CONDITION statement that always returns a true value. The value is true because the character in position 1 always has a value greater than or equal to hexadecimal zero. Therefore, every time a page change occurs (front to back, or
back to next front) a different page format is started. The different DIRECTION statements in the two page formats change the layout of the text on the page.

Observe that the COPYGROUP parameter is specified as NULL. If a parameter other than NULL or / is specified for COPYGROUP, the copy group restarts every time a page change occurs. Because restarting a copy group forces data to a new sheet, duplex printing does not occur.

FORMDEF XMPDUP
    DUPLEX NORMAL;

PAGEDEF XMPDUP WIDTH 8.5 HEIGHT 11.0;
    PAGEFORMAT P2FRONT DIRECTION ACROSS;
    PRINTLINE CHANNEL 1 POSITION 0.75 TOP;
    CONDITION GOTOBACK START 1 LENGTH 1
    WHEN GE X'00' AFTER SUBPAGE NULL PAGEFORMAT P2BACK;
    PRINTLINE REPEAT 59;

    PAGEFORMAT P2BACK DIRECTION UP;
    PRINTLINE CHANNEL 1 POSITION 0.25 TOP;
    CONDITION GOTOFRNT START 1 LENGTH 1
    WHEN GE X'00' AFTER SUBPAGE NULL PAGEFORMAT P2FRONT;
    PRINTLINE REPEAT 59;

Figure 84. Duplex Output

Record Reprocessing Example

Figure 85 on page 132 uses the BEFORE SUBPAGE function with record reprocessing because the copy group and page format cannot be determined until input record 3 for each subpage is read.

Notes:

1. This example includes two subpages.
2. The CONDITION command specifies that the action to be done is NEWFORM. Therefore, if the condition is satisfied, the data in the current subpage is forced to start on the next form. If the data is already at the start of a new form, no action is done. In other words, a blank page is not generated.
Selecting Paper from an Alternate Bin Example

[Figure 86 on page 133] selects the first sheet from the alternate bin and all other pages from the primary bin. This function is useful when special paper (such as one having the company logo) is used for the first page of a document.

Note: If the form defined to each bin is the same form number, bin selection is overridden by the printer. Only the primary bin is selected.
Multiple CONDITION Commands

Two examples are shown here. The first example shows how two CONDITION commands can interact to give unintended results. The second example shows how to use the two CONDITION commands to achieve the correct results.

Example 1 - Incorrect Solution

The example in Figure 87 on page 134 demonstrates how two CONDITION commands can interact to give unintended results. Specifically, one CONDITION command causes a change of page format and then a second CONDITION command inspects a field with a WHEN CHANGE subcommand.

The purpose of condition:

**NEWREP**

Starts a new report on a new sheet of paper whenever the specified field changes and jogs the output so the report can be easily located.

**SHIFTB** and **SHIFTF**

Handles the situation where all four subpages of the front (or back) contain data.

In this situation, the objective is to change the print direction of the text on the page.

In the situation where both conditions seem to be true at the same time, the results might be unexpected.

**Note:** Condition SHIFTB (or SHIFTF) takes effect after the current subpage; therefore, it precedes the before subpage processing that is defined by condition

---

```c
/* The form definition contains two copy groups -- */
/* ALTBIN - for the first page */
/* PRIBIN - for all other pages */

FORMDEF BINEX
     DUPLEX NO;
     COPYGROUP ALTBIN BIN 2;
     COPYGROUP PRIBIN BIN 1;

PAGEDEF BINEX
     WIDTH 8.3 HEIGHT 10.6;

/* Pageformat for first page - bin 2 */
PAGEFORMAT FIRST;
     PRINTLINE CHANNEL 1
     POSITION MARGIN TOP;
     CONDITION GOTOPRIM START 1 LENGTH 1
     WHEN GE X'00' AFTER SUBPAGE
     COPYGROUP PRIBIN PAGEFORMAT REST;
     PRINTLINE REPEAT 59;

/* Pageformat for all other pages - bin 1 */
PAGEFORMAT REST;
     PRINTLINE CHANNEL 1
     POSITION MARGIN TOP
     REPEAT 60;

Figure 86. Alternate bin example
```
NEWREP. Because condition SHIFTB starts a new page format, condition NEWREP returns a false value, and the expected new report processing is not done.

FORMDEF XMPICO OFFSET 0 0 DUPLEX RTUMBLE JOG YES REPLACE YES;
   COPYGROUP CG1;
   OVERLAY OVLY1;
   OVERLAY OVLY2;
   SUBGROUP OVERLAY OVLY1 FRONT;
   SUBGROUP OVERLAY OVLY2 BACK;

PAGEDEF XMPICO REPLACE YES;
   FONT GT24;
   FONT GT12;

/* Definition of pageformat for front side */
PAGEFORMAT PFFRONT WIDTH 11 IN HEIGHT 8.5 IN DIRECTION UP;
SETUNITS 1 PELS 1 PELS LINESP 16 LPI;
   PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 75 188;
   CONDITION NEWREP START 8 LENGTH 3
   WHEN CHANGE BEFORE SUBPAGE COPYGROUP CG1 PAGEFORMAT PFFRONT;
   PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
ENDSUBPAGE;

   PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 188;
   CONDITION NEWREP START 8;
   PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
ENDSUBPAGE;

   PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 75 1102;
   CONDITION NEWREP START 8;
   PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
ENDSUBPAGE;

   PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 1102;
   CONDITION NEWREP START 8;
   CONDITION SHIFTB START 1 LENGTH 1
   WHEN GE X'00' AFTER SUBPAGE NULL PAGEFORMAT PFBACK;
   PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
ENDSUBPAGE;

/* Definition of pageformat for back side */
PAGEFORMAT PFBACK WIDTH 8.5 IN HEIGHT 11 IN DIRECTION ACROSS;
SETUNITS 1 PELS 1 PELS LINESP 8 LPI;
   PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 61;
   CONDITION NEWREP START 8;
   PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
ENDSUBPAGE;

   PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 1335;
   CONDITION NEWREP START 8;
   CONDITION SHIFTF START 1 LENGTH 1
   WHEN GE X'00' AFTER SUBPAGE NULL PAGEFORMAT PFFRONT;
   PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
ENDSUBPAGE;

Figure 87. INCORRECT Solution Example

Example 2 - Correct Solution
The example in Figure 88 on page 135 demonstrates how two CONDITION commands are used to achieve the correct results. The Figure 88 on page 135 differs from Figure 87 in two significant ways:

- Because the page format for the back side is the first one defined in the page definition, it is the one that is initially active.
• Both CONDITION commands (NEWREP and SHIFTIT) specify that the action must happen before the current subpage is processed.

When processing begins, condition NEWREP fails because a WHEN CHANGE condition is in place and the page format has started. However, condition SHIFTIT returns a true result, and the NEXT page format (PFFRONT) is started. No lines are formatted, so condition SHIFTIT moves to the page format for the front side.

```
FORMDEF XMPCOR OFFSET 0 0 DUPLEX RTUMBLE JOG YES REPLACE YES;
  COPYGROUP CG1;
  OVERLAY OVLY1;
  OVERLAY OVLY2;
  SUBGROUP OVERLAY OVLY1 FRONT;
  SUBGROUP OVERLAY OVLY2 BACK;

PAGEDEF XMPCOR REPLACE YES;
  FONT GT24;
  FONT GT12;
  /* The pageformat for the back side of the form is */
  /* the first pageformat in the PAGEDEF. Therefore, */
  /* it will initially be the active pageformat */
  PAGEFORMAT PFBACK WIDTH 8.5 IN HEIGHT 11 IN DIRECTION ACROSS;
SETUNITS 1 PELS 1 PELS LINESP 8 LPI;
PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 61;
  CONDITION NEWREP START 8 LENGTH 3
    WHEN CHANGE BEFORE SUBPAGE COPYGROUP CG1 PAGEFORMAT PFFRONT;
  CONDITION SHIFTIT START 1 LENGTH 1
    WHEN GE X'00' BEFORE SUBPAGE NULL NEXT;
PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
ENDSUBPAGE;

PRINTLINE REPEAT 1 CHANNEL 1 FONT GT12 POSITION 75 1335;
  CONDITION NEWREP START 8;
PRINTLINE REPEAT 40 FONT GT12 POSITION 75 NEXT;
ENDSUBPAGE;

/* This is the pageformat for the front side of the form. */
PAGEFORMAT PFFRONT WIDTH 11 IN HEIGHT 8.5 IN DIRECTION UP;
SETUNITS 1 PELS 1 PELS LINESP 16 LPI;
PRINTLINE REPEAT 1 CHANNEL 1 FONT GT23 POSITION 75 188;
  CONDITION NEWREP START 8;
  CONDITION SHIFTIT START 1;
PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
ENDSUBPAGE;

PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 188;
  CONDITION NEWREP START 8;
PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
ENDSUBPAGE;

PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 75 1102;
  CONDITION NEWREP START 8;
PRINTLINE REPEAT 40 FONT GT24 POSITION 75 NEXT;
ENDSUBPAGE;

PRINTLINE REPEAT 1 CHANNEL 1 FONT GT24 POSITION 1377 1102;
  CONDITION NEWREP START 8;
PRINTLINE REPEAT 40 FONT GT24 POSITION 1377 NEXT;
ENDSUBPAGE;

Figure 88. CORRECT Solution Example
```
Field Processing When PRINTLINEs Are Repeated

Figure 89 and Figure 90 on page 137 are examples that show the effect of the [LINE | FIELD] parameter on REPEAT n.

Figure 89 is a PRINTLINE example that uses FIELD type repetition.

PAGEDEF rept01 WIDTH 8.0 IN
  HEIGHT 10.5 IN
  LINEONE 0.2 IN 0.2 IN
  DIRECTION ACROSS
  REPLACE YES;

  FONT normal CR10 SBCS ROTATION 0;
  FONT italic CI10 SBCS ROTATION 0;
  FONT bold CB10 SBCS ROTATION 0;

  .
  .
  SETUNITS LINESP 6 LPI;

  PRINTLINE POSITION 1.0 IN 1.0 IN
    DIRECTION ACROSS
    FONT bold
    REPEAT 3 FIELD;
  FIELD POSITION 0.0 IN 0.0 IN
    DIRECTION ACROSS
    FONT normal
    START * LENGTH 20;
  FIELD POSITION 2.5 IN 0.0 IN
    DIRECTION DOWN
    FONT normal
    START * LENGTH 20;
  FIELD POSITION 2.5 IN 2.5 IN
    DIRECTION BACK
    FONT normal
    START * LENGTH 20;
  FIELD POSITION 0.0 IN 2.5 IN
    DIRECTION UP
    FONT normal
    START * LENGTH 20;

Figure 89. REPEAT n type FIELD Example

Figure 90 on page 137 is a PRINTLINE example that shows LINE type repetition.

Note: When LINE type repetition is used, SETUNITS LINESP might need to be set to a higher value to avoid over printing.
Figure 91 shows an example of Input Line Data.

When Figure 91 is processed by the print server, the output in Figure 92 on page 138 and Figure 93 on page 139 is printed.
Figure 93. Input Line Data Sample Output 2

Field Type Repeat
Instance of the same special effects are
Notice that the field, and not the desired.
fields are repeated on the same line. This
command...

for sales tickets. Generally, this type
of repeat needs a

SETUNITS LINESP 3 IN 
...so that lines won’t overlap!

Line Type Repeat
is based on the prior
has advantages if

SETUNITS LINESP

Good printline.
Chapter 7. N_UP Printing

With N_UP printing, which is defined in the form definition, you can print up to four pages on a sheet of paper in simplex mode and up to eight pages in duplex mode. Each page is independent, allowing the use of different page formats and copy groups for each page. This format provides significantly more flexibility and function than the traditional multiple-up capability, which is defined in the page definition. See “N_UP Compared to Multiple-up” on page 161 for more differences between N_UP printing and multiple-up printing.

Two levels of N_UP exist:

- Basic N_UP supported by older AFP printers: 3825, 3827, 3828, 3829, 3835, and 3900-001.
- Enhanced N_UP supported by printers with the Advanced Function Common Control Unit (AFCCU).

N_UP Partitions and Partition Arrangement

A key concept in N_UP printing is the partition. In both basic and enhanced N_UP, each sheet of paper is divided into equal size areas called partitions. Pages are placed in these partitions in sequential order in basic N_UP. Pages are placed in relation to one or more of these partitions in enhanced N_UP. Knowing the partition arrangement is critical to designing applications that use N_UP.

Note: If you are using basic N_UP printing with PSF set to DATACK=BLOCK, data must fall within the boundary of the partition. Any data that is placed outside the edge of the partition boundary is not printed, and no error message is generated. However, enhanced N_UP printing allows pages to overlap partitions. The only limits are that the pages must not extend beyond the boundaries of the physical sheet, and the pages must not exceed the total number of N_UP partitions that are specified for the sheet.

The number, size, and position of partitions are determined by three things:

- N_UP value (1, 2, 3, or 4)
- Size and shape of the sheet of paper
- Form definition presentation options (PRESENT and DIRECTION)

When printing in duplex mode, the same number of partitions is also defined for the back of the sheet. For normal duplex, back partitions are placed as if the sheet were flipped around its right side or y-axis. For tumble duplex, they are placed as if the sheet were flipped around its top edge, or x-axis. See Figure 95 on page 143 and Figure 96 on page 143 for illustrations of duplex partitions.

Figure 94 on page 142 through Figure 105 on page 148 show the partition arrangement that results from every combination of N_UP value, paper size, and presentation option. The hex values (X'00', X'01', X'04', X'05', X'02', X'03') indicate how the Medium Orientation Triplet (X'68') specifies the position and orientation of the medium presentation space on the physical medium.

---

2. You must have the correct level of PPFA to generate basic or enhanced N_UP commands and the correct level of PSF for your operating system to drive the printer in the basic or enhanced N_UP mode.
Use these figures to determine how your N_UP application is formatted by the printer. In the figures, each equal-sized partition has a number that indicates its default presentation sequence. The origin for each partition is in the same relative position as the origin point shown for the medium. This point serves as the upper-left corner for a page that is printed in the ACROSS (or 0°) printing direction.

Figure 94. N_UP 1 Partition Numbering, Front Sheet-Side
Figure 95. N_UP 2 Partition Numbering, Front Sheet-Side

Figure 96. N_UP 3 Partition Numbering, Front Sheet-Side
Figure 97. N_UP 4 Partition Numbering, Front Sheet-Side

Figure 98. N_UP 1 Partition Numbering, Back Sheet-Side, Normal Duplex
<table>
<thead>
<tr>
<th>Portrait (across)</th>
<th>Landscape (across)</th>
<th>Portrait 90 (down)</th>
<th>Landscape 90 (down)</th>
<th>Reverse Portrait</th>
<th>Reverse Landscape</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 00</td>
<td>X 01</td>
<td>X 04</td>
<td>X 05</td>
<td>X 02</td>
<td>X 03</td>
</tr>
</tbody>
</table>

**Cut Sheet**

- 2
- 1

**Wide Fanfold**

- 1
- 2
- 3

**Narrow Fanfold**

- 3
- 2
- 1

*Figure 99. N_UP 2 Partition Numbering, Back Sheet-Side, Normal Duplex*

<table>
<thead>
<tr>
<th>Portrait (across)</th>
<th>Landscape (across)</th>
<th>Portrait 90 (down)</th>
<th>Landscape 90 (down)</th>
<th>Reverse Portrait</th>
<th>Reverse Landscape</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 00</td>
<td>X 01</td>
<td>X 04</td>
<td>X 05</td>
<td>X 02</td>
<td>X 03</td>
</tr>
</tbody>
</table>

**Cut Sheet**

- 2
- 1
- 3

**Wide Fanfold**

- 1
- 2
- 3

**Narrow Fanfold**

- 3
- 2
- 1

*Figure 100. N_UP 3 Partition Numbering, Back Sheet-Side, Normal Duplex*
Figure 101. N_UP 4 Partition Numbering, Back Sheet-Side, Normal Duplex

Figure 102. N_UP 1 Partition Numbering, Back Sheet-Side, Tumble Duplex
Figure 103. N_UP 2 Partition Numbering, Back Sheet-Side, Tumble Duplex

Figure 104. N_UP 3 Partition Numbering, Back Sheet-Side, Tumble Duplex
Basic N_UP Printing

You can specify the N_UP subcommand on either the FORMDEF or COPYGROUP commands in the form definition. The following syntax diagrams show the subcommands and parameters that are enabled with basic N_UP printing.

**FORMDEF Subcommand**

**COPYGROUP Subcommand**
**OVERLAY Subcommand:**

The N_UP subcommand divides the medium into one, two, three, or four partitions, as described in “N_UP Partitions and Partition Arrangement” on page 141. The OVERLAY subcommand prints a page overlay in each partition at a specified offset from the page origin or the partition origin. For more information about page overlays, see “Medium Overlays and Page Overlays” on page 160.

The INVOKE subcommand controls the action that occurs if you invoke a new copy group. You can invoke copy groups by using conditional processing in the page definition or by including an Invoke Medium Map (IMM) structured field in the print data. The default action is to eject to a new sheet. By specifying an INVOKE subcommand on a COPYGROUP command, you can instead eject to a new N_UP partition, which might be on the same sheet. If printing in duplex mode, you can specify whether to eject to a partition on the front or back side of the sheet.

You must use page overlays instead of medium overlays if you want to change overlays while you are ejecting to a new partition. PSF accepts the NEXT, FRONT, and BACK values of the INVOKE subcommand only if the new copy group has the same medium modifications as the previous copy group. Medium modifications include duplexing, bin, page offset, N_UP values, presentation, direction, and medium overlays. If any of these modifications differ, PSF ejects to a new sheet when the copy group is invoked.

By combining INVOKE with the N_UP OVERLAY subcommand, you can place different page overlays in different partitions when you change copy groups. This format is illustrated in “Example 1: Basic N_UP Using INVOKE and OVERLAY” on page 150.

The following examples show the use of basic N_UP. Because each example builds on the previous one, read them in sequential order to better understand basic N_UP. All the pages that are used in the examples are formatted in the ACROSS printing direction. Their orientation on the media is the result of using the available PRESENT and DIRECTION combinations in the FORMDEF command.
**Example 1: Basic N_UP Using INVOKE and OVERLAY**

![Diagram](image)

*Figure 106. Basic N_UP Using INVOKE and OVERLAY*

```plaintext
FORMDEF TWOUPS ;

COPYGROUP A
  N_UP 2
  OVERLAY A
  INVOKE NEXT ;

COPYGROUP B
  N_UP 2
  OVERLAY B
  INVOKE NEXT ;
```

*Figure 107. Form Definition for Basic N_UP Using INVOKE and OVERLAY*

*Figure 106* shows the INVOKE and OVERLAY functions of basic N_UP printing. Specifying INVOKE NEXT on the COPYGROUP command ensures that when the copy group is invoked by an Invoke Medium Map (IMM) structured field with conditional processing, the next page is placed in the next partition of the N_UP form.

The OVERLAY subcommand specifies a *page overlay*, which can be positioned relative to the page origin or relative to the partition origin. In basic N_UP, the OVERLAY subcommand prints the overlay with the page data in every partition.
on the sheet. However, as shown in this example, by using INVOKE NEXT, the application can use different overlays in different partitions.

Example 1 is defined as N_UP 2 simplex with the default PORTRAIT ACROSS presentation, which results in the partitions that are illustrated in Figure 106 on page 150. The application uses different page formats on different application pages. With N_UP, changing page formats ejects to the next partition, just as it ejects to a new page in applications without N_UP.

The application also needs different overlays on different pages. Because the overlays are specified on N_UP in the COPYGROUP subcommand, the application can put different overlay on different pages by changing copy groups. Without the INVOKE subcommand, changing the copy group forces an eject to a new physical sheet. However, because INVOKE NEXT is specified, the eject is to the next partition. Changing to copy group B after page 1 is written places page 2 in partition 2 of the same physical sheet. If the change is made after a page is placed in partition 2, the eject to the next partition is to partition 1 of the next sheet. The page is printed with the overlay that is specified in the new copy group.

Notes:
1. The pages in this example are line-format print data, which is formatted with a page definition. The example is the same for MO:DCA data, except that page formats are not used.
2. You can select the copy groups and page formats by including IMM and IDM structured fields in the print data or by using conditional processing in the page formats.
3. Overlays can be defined as page overlays in the page definition or in the form definition N_UP or PLACE subcommands. Overlays can also be defined as medium overlays in the form definition SUBGROUP command. If you want to change overlays when you are ejecting to a new partition, use page overlays instead of medium overlays. See “Medium Overlays and Page Overlays” on page 160 for information about page and medium overlays.
Example 2: Basic N_UP Normal Duplex

Figure 108. Basic N_UP Normal Duplex

```
FORMDEF NUPDUP
N_UP 2
PRESENT PORTRAIT
DIRECTION ACROSS
DUPLEX NORMAL ;
```

Figure 109. Form Definition for Basic N_UP Normal Duplex

Figure 108 shows the partition order for duplexed pages. This figure also shows the partitions into which the sheet is divided for N_UP 2 with PORTRAIT presentation and ACROSS direction. With normal duplex, the sheet is rotated around its y-axis, which is the right edge of the sheet. The result is that partition 2 for the back side of the sheet is on the back of partition 1 for the front side, and page 4 is on the back of page 1. The tops of pages 3 and 4 are aligned with the tops of pages 1 and 2.
Example 3: Basic N_UP 2 Tumble Duplex

![Diagram of Basic N_UP 2 Tumble Duplex](image)

**Figure 110. Basic N_UP 2 Tumble Duplex**

```
FORMDEF NUPTUM
N_UP 2
PRESENT PORTRAIT
DIRECTION ACROSS
DUPLEX TUMBLE ;
```

**Figure 111. Form Definition for Basic N_UP 2 Tumble Duplex**

Figure 110 shows the partition order for tumble duplex pages. This figure also shows the partitions into which the sheet is divided for N_UP 2 with PORTRAIT presentation and ACROSS direction. With tumble duplex, the sheet is rotated around its x-axis, which is the top of the sheet. The result is that partition 1 of the back of the sheet falls on the back of partition 1 for the front, and page 3 falls on the back of page 1. The tops of pages 3 and 4 are aligned with the bottoms of pages 1 and 2. For more information about normal and tumble duplex printing, see “Normal Duplex and Tumble Duplex” on page 14.

Enhanced N_UP Printing

Enhanced N_UP is supported on AFP continuous forms printers. In addition to all the function of basic N_UP, enhanced N_UP includes the powerful PLACE subcommand.

Using the PLACE subcommand, you can place pages in the partitions in any sequence, specify unique overlays for each page, and rotate both the page and the overlays in the partitions. You can place multiple pages in the same partition and no pages in other partitions, and you can extend pages across partition boundaries. In short, you can place pages of any size at any location on the front or back of the sheet, in any orientation. The only limits are that the pages must not extend...
outside the boundaries of the physical sheet, and the pages must not exceed the total number of N_UP partitions that are specified for the sheet.

You use a single PLACE command to place each page of data on the sheet. You must specify the same number of PLACE commands as the number of N_UP partitions for the sheet. This specification is required for error recovery and restart integrity. If you do not want to place as many pages as partitions, you can specify CONSTANT on a PLACE command to indicate that no data is to be placed in the partition. You can specify the OVERLAY subcommand with the CONSTANT subcommand to place overlays without user’s data. The following syntax diagrams show the subcommands and parameters that are enabled with enhanced N_UP printing.

For most applications, place constant overlays before you place data on the sheet because the overlay is not placed until the next page of data is placed. If your application changes copy groups or runs out of pages on the sheet before it reaches the constant overlay PLACE subcommand, the constant overlay is not printed. However, if you do not want the overlays to print in these cases, place the constant overlay after you place the page data.
PLACE Subcommand:

(1) PLACE n FRONT BACK CONSTANT OFFSET x-pos y-pos

OVERLAY Subcommand

ROTATION 0 90 180 270 VIEW YES NO

Notes:

1. The use of the PLACE subcommand indicates enhanced N_UP printing.

The following examples show enhanced N_UP printing. Read these examples in sequence to better understand enhanced N_UP printing.

**Example 1: Enhanced N_UP Using PLACE**

![Diagram of enhanced N_UP using PLACE subcommand]

Figure 112. Enhanced N_UP Using PLACE

FORMDEF BOOKLT DUPLEX NORMAL
N_UP 2
/* Page 1 */ PLACE 2 FRONT
/* Page 2 */ PLACE 1 BACK
/* Page 3 */ PLACE 2 BACK
/* Page 4 */ PLACE 1 FRONT ;

Figure 113. Form Definition for Enhanced N_UP Using PLACE

Figure 112 shows the function of the PLACE subcommand in specifying the sequence of partitions into which pages are placed. This example is N_UP 2 duplex. The default partition sequence is from left to right. Notice that when printing in normal duplex, partition 1 on the back of the sheet aligns with partition 2 on the front of the sheet. See “Example 2: Basic N_UP Normal Duplex” on page 152 and “Example 3: Basic N_UP 2 Tumble Duplex” on page 153 for information on N_UP duplex partitions.

For this booklet, you do not want to print pages in the default order: partitions 1 and 2 on the front, followed by partitions 1 and 2 on the back. Instead, print the
pages so that when the sheet is folded, you have a booklet, with page 1 on the front outside of the booklet, pages 2 and 3 inside the folded booklet, and page 4 on the back outside of the booklet. The form definition that is shown in Figure 113 on page 155 uses the PLACE subcommand of enhanced N_UP to place pages in the partitions in the order that is needed. The application writes the pages in order, page 1 through page 4, and the N_UP form definition provides the correct sequencing in the partitions.

Example 2: Enhanced N_UP Using CONSTANT and OVERLAY

![Diagram of Enhanced N_UP Using CONSTANT and OVERLAY]

Select copy group PAGE1.
Select page format PAGE1.
Write data for page 1.

Select copy group PAGEON.
Select page format PAGEON.
Write data for page 2.
Write data for page 3.

Write data for page 4.

Select copy group PAGE1.
Select page format PAGE1.
Write data for page 1.

Figure 114. Enhanced N_UP Using CONSTANT and OVERLAY
Introduction to CONSTANT subcommand of enhanced N_UP and shows the functions of the PLACE subcommand, which was described in "Example 1: Enhanced N_UP Using PLACE" on page 155 and the INVOKE subcommand, which was described in "Example 1: Basic N_UP Using INVOKE and OVERLAY" on page 150. This figure represents a user application that is printing customer statements with the values N_UP 2 duplex. The PLACE subcommand places the pages in the correct order for post-processing equipment to cut the sheets into individual pages and interleave them to produce sequential pages. The INVOKE subcommand ensures that one customer’s statement is never printed on the back side of another customer’s statement. The N_UP 2 subcommand, which is combined with the default PORTRAIT ACROSS presentation, divides the sheet into the two partitions that are illustrated in Figure 114 on page 156.

In Figure 114 on page 156, page 1 of each customer’s statement is printed with overlay A. The back side of page 1 is a constant overlay, with no user’s data. The remaining pages of each customer’s statement are printed with overlay B.

The copy groups place the required overlays on both the right and left halves of the sheet so that a new customer statement can begin on either half of the sheet. COPYGROUP PAGON assigns overlay B to all partitions on the sheet. COPYGROUP PAGE1 assigns overlay A to all front partitions and overlay C to all back partitions. The CONSTANT parameter that is used with OVERLAY C means that no user’s data is printed in the partition with the overlay. To ensure that the constant overlay prints whenever page 1 is printed, the PLACE subcommand for the constant overlay is specified before the PLACE subcommand for page 1 print data. The INVOKE subcommand specifies BACK to ensure that the overlay is printed on the back of the partition.

In the application that is shown in Figure 114 on page 156, the copy group is changed to PAGON after page 1 is printed. Because the constant overlay and page 1 were printed with the first two PLACE commands of copy group PAGE1, the third PLACE command in new copy group is used for the next page. Page 2 of statement 1 is placed in partition 2 front, as specified in the third PLACE subcommand of copy group PAGON.

After the fourth and last page of statement 1, the copy group is changed back to PAGE1 to print page 1 of statement 2. Page 4 of statement 1 printed in front partition 1 using the first PLACE subcommand of copy group PAGON. N_UP

```
FORMDEF STATMT DUPLEX NORMAL ;

COPYGROUP PAGE1
   INVOKE BACK
   N_UP 2
   PLACE 2 BACK CONSTANT OVERLAY C
   PLACE 1 FRONT OVERLAY A
   PLACE 1 BACK CONSTANT OVERLAY C
   PLACE 2 FRONT OVERLAY A ;

COPYGROUP PAGON
   INVOKE NEXT
   N_UP 2
   PLACE 1 FRONT OVERLAY B
   PLACE 2 BACK OVERLAY B
   PLACE 2 FRONT OVERLAY B
   PLACE 1 BACK OVERLAY B ;

Figure 115. Form Definition for Enhanced N_UP Using CONSTANT and OVERLAY
```
selects the second PLACE subcommand of copy group PAGE1: PLACE 1 FRONT. But the INVOKE subcommand for copy group PAGE1 specifies BACK. N_UP continues sequentially through the PLACE subcommands of copy group PAGE1 until it finds a BACK partition. The third PLACE subcommand is: PLACE 1 BACK CONSTANT OVERLAY C. The constant overlay is placed in partition 1 on the back of the sheet, then page 1 of the new customer's statement is printed by using the next PLACE subcommand: PLACE 2 FRONT on the front side of the constant overlay.

**Note:** You can use NEXT, FRONT, or BACK on the INVOKE subcommand only when you are switching between copy groups that have identical medium modifications, including identical N_UP values and an identical number of PLACE subcommands. If the copy groups have different values, the INVOKE command causes an eject to a new physical sheet.

### Example 3: Enhanced N_UP Asymmetric Pages

**Figure 116. Enhanced N_UP Asymmetric Pages**

**Sheets:**
- **Sheet 1:** Select copy group FIRST.
  - Write page 1.
  - Write page 2.
  - Write page 3.
  - Write page 4.
  - Write page 5.
  - Write page 6.
- **Sheet 2:** Select copy group SECOND.
  - Write page 8.
  - Write page 9.
  - Write page 10.
  - Write page 11.

*Figure 116 shows the flexibility and power of enhanced N_UP printing. With enhanced N_UP printing, you can place pages relative to any partition on the sheet, front or back, in any sequence. Pages are not limited by partition boundaries. The only limitations are that pages must not print outside the physical form boundaries, and you cannot place more pages on a sheet than the number*
specified in the N_UP subcommand. For an N_UP 4 duplex page, the limit is eight pages total on front and back sides combined. For N_UP 3 duplex, the limit is six pages on the front and back combined.

Note: In the duplex examples in Figure 116 on page 158 all of the pages can be on one side, with the other side blank.

FORMDEF ASYMET DUPLEX NORMAL ;

COPYGROUP FIRST
PRESENT LANDSCAPE DIRECTION ACROSS
N_UP 4
/* Constant*/ PLACE 1 BACK OFFSET 4 0 CONSTANT OVERLAY B
/* Page 1*/ PLACE 1 FRONT OFFSET 0 0 Overlay A
/* Page 2*/ PLACE 1 FRONT OFFSET 12 0
/* Page 3*/ PLACE 1 BACK OFFSET 0 0
/* Page 4*/ PLACE 1 FRONT OFFSET 12 4
/* Page 5*/ PLACE 1 BACK OFFSET 0 4
/* Page 6*/ PLACE 1 FRONT OFFSET 12 8
/* Page 7*/ PLACE 1 BACK OFFSET 0 8 ;

COPYGROUP SECOND
PRESENT PORTRAIT DIRECTION ACROSS
N_UP 3
/* Constant*/ PLACE 1 BACK OFFSET 0 0 CONSTANT OVERLAY C
/* Page 8*/ PLACE 1 FRONT OFFSET 0 0
/* Page 9*/ PLACE 1 FRONT OFFSET 0 4
/* Page 10*/ PLACE 1 FRONT OFFSET 6 4
/* Page 11*/ PLACE 1 FRONT OFFSET 6 8
/* 6th place*/ PLACE 1 BACK OFFSET 0 0 CONSTANT ;

Figure 117. Form Definition for Enhanced N_UP Asymmetric Pages

To achieve the asymmetrical page placement that is shown in this example, place all the pages relative to the origin of partition 1 on the front or the back side of the sheet. You can place the pages relative to the origin of any of the partitions, but if you use partition 1, it simplifies the calculations for page positions.

With N_UP 4, the default PORTRAIT presentation and ACROSS direction place the origin at the upper right of the partition on wide, continuous-form paper. In this example, specifying LANDSCAPE ACROSS sets the origin at the upper-left corner to achieve the correct page arrangement.

The coding of the form definition is shown in Figure 116 on page 158. Copy group FIRST specifies N_UP 4, which requires eight PLACE subcommands for the duplex page. Observe that the constant overlay B on the back of the sheet represents one of the eight PLACE subcommands. COPYGROUP SECOND used for the second sheet specifies N_UP 3. You must use six PLACE subcommands. Four pages are placed on the front side, and a constant overlay is placed on the back, with five of the six PLACE subcommands. A CONSTANT page is specified without an overlay to fill the sixth PLACE subcommand. Nothing is printed with this PLACE subcommand, but it is required to ensure a correct internal page count for recovery and restart.

Note: In each copy group, the PLACE subcommand for the constant overlay is placed in front of all the PLACE subcommands for page data. This placement ensures that the constant overlay prints if any pages are printed on the sheet. Otherwise, if you change copy groups or run out of pages before the PLACE command for the constant overlay, the overlay does not print.
Additional N_UP Considerations

N_UP can affect the scope of other PPFA commands that operate on a page or a medium.

**COPIES**

The COPIES subcommand in the SUBGROUP of the form definition operates on the physical medium. When you specify five copies with N_UP 2, you get five sheets of the N_UP 2 data.

**SUPPRESSION**

The SUPPRESSION subcommand in the SUBGROUP of the form definition operates on the physical medium. The suppression names in the SUBGROUP operate on all N_UP pages on the sheet.

**OVERLAY**

You can specify an OVERLAY subcommand in multiple places in the form definition and can also specify an overlay in the page definition. The result is either a page overlay or a medium overlay. See "Medium Overlays and Page Overlays" for a description of the differences between these commands and the uses of these overlays.

**PRESENT DIRECTION**

You use the PRESENT and DIRECTION subcommands of the form definition with the N_UP subcommand to determine partition arrangement. These commands, which are described in this update guide, now affect all N_UP printers, including cut-sheet printers.

**CONDITION**

You can use the CONDITION command of the page definition with N_UP just as you use it with non N_UP jobs. However, the NEWSIDE and NEWFORM parameters might operate differently than you expect. NEWSIDE, which is equivalent to invoking a new page format, ejects to the next partition, which might not be on a new side of an N_UP sheet. NEWFORM, which is equivalent to invoking a new copy group, ejects to a new sheet with basic N_UP. The effect with enhanced N_UP depends on the coding of the INVOKE subcommand.

Medium Overlays and Page Overlays

An AFP overlay can be used as a page overlay or as a medium overlay. Different actions are done on these two different types of overlays. Page overlays apply to the page and are placed relative to the page origin. Medium overlays always apply to the entire medium and are placed at the medium origin. When used with N_UP, the medium overlay still applies to the entire sheet of paper, not to the individual partitions.

The same overlay can be either a page overlay or a medium overlay, depending on the method that is used to invoke it for printing. An overlay that is invoked by a page definition or by an Include Page Overlay (IPO) structured field is always a page overlay. An overlay that is invoked by a form definition without N_UP is always a medium overlay. When N_UP is specified in the form definition, you can specify commands to invoke a page overlay. The following examples show the ways in which overlays can be invoked.

160 Page Printer Formatting Aid: User's Guide
N_UP Compared to Multiple-up

With the addition of the N_UP capability, AFP now provides two methods to format multiple application pages on a single sheet:

- N_UP as defined in a form definition
- Multiple-up as defined in a page definition

The multiple-up function is available for line-format data that is printed on AFP printers. Multiple-up achieves the appearance of multiple pages on a sheet by formatting multiple groups of print lines as a single AFP page. The output is still a single AFP page on a side of a sheet, and the entire output is formatted by a single page format. If the application pages within that sheet require different print
layouts, you must design a different page format for all possible arrangements of data. For example, if one side of a 2-up sheet has 10 different print layouts, you need 100 different page formats to cover all the possible combinations.

In contrast, N_UP is used to place multiple AFP pages on a side of a sheet, which means that each of the N_UP pages can be formatted with a different page format. You can change page formats between each N_UP page without ejecting to a new side of the sheet. For the same example with N_UP, you need only 10 page formats for a 2-up sheet with 10 different print layouts.

N_UP also means that you can place multiple pages of fully composed AFP data (or MO:DCA data) on a single sheet. This was not possible with the multiple-up function that is defined in the page definition because AFP data does not use a page definition.
Chapter 8. AFP Color Management

You can use various ways to print color data with Advanced Function Presentation (AFP). However, to implement an AFP color printing solution with full color management, you must use color management resources (CMRs). Also, for best results, install all of your color images as data objects and associate CMRs with them.

Color management resources

Color management resources (CMRs) are the foundation of color management in AFP print systems. CMRs are AFP resources that provide all the color management information, such as ICC profiles and halftones, that an AFP system needs to process a print job and maintain consistent color from one device to another.

CMRs share some characteristics with other AFP resources, but are different in some important ways.

CMRs are similar to other AFP resources in these ways:

- CMRs can be associated with elements of a print job at various levels of the hierarchy.
  
  Normal hierarchy rules apply, so CMRs specified at lower levels override the ones that are specified at the higher level. For example, a CMR set on a data object overrides a default CMR set on a print file.

- CMRs can be included in a print job in an inline resource group and referenced in a form definition, page environment, object environment, or an include Object (IOB) structured field.

  Note: CMRs can vary in size from several hundred bytes to several megabytes. If your print job uses relatively few CMRs, including them in the print file might not have an impact on the performance of your system. However, if your print job uses more than 10 CMRs, the size of the print job can increase so much that file transfer rates and network traffic are affected.

- CMRs can be stored centrally in a resource library, so you do not need to include them in every print job. You can configure all your print servers so they can access the CMRs.

- For the print server to find CMRs, the resource library must be listed in the AFP resource search path on the print server.

CMRs are different from other AFP resources in these ways:

- You cannot copy CMRs into a resource library as you can other AFP resources. To store CMRs in a central resource library, you must install them using an application such as AFP Resource Installer.

- CMRs and data objects must be stored in resource libraries that have resource access tables (RATs).

  AFP Resource Installer creates the RAT when CMRs and data objects are installed. For best results, install CMRs and data objects in separate resource libraries and store resources that do not require RATs (such as form definitions, page definitions, and overlays) in other resource libraries.
CMRs installed in a resource library can have names longer than 8 characters, and you can use the names in the print data stream. These names are created when you install the CMR with AFP Resource Installer and are UTF-16BE encoded.

**Types of CMRs**

Different situations call for different types of CMRs. Some CMRs are created by product manufacturers so you can download and use them, while others are created by your printer or other color management software. If you have the appropriate information, you can also create CMRs yourself.

Some CMRs are used to interpret input files (similar to the function done by ICC input profiles), while others are used to prepare the final print job output for a specific printer (similar to the function done by ICC output profiles).

**Color conversion CMR**

Color conversion CMRs are used to convert colors to and from the ICC Profile Connection Space (PCS), a device-independent color space. You can use them to prepare images for color or grayscale printing.

Color conversion CMRs are an essential element of any AFP color management system because they are ICC profiles encapsulated in AFP structures. The AFP structures add information that your color management system can use, but it leaves the ICC profile unaltered.

You can use color conversion CMRs to produce consistent colors on different devices. In a color system, color conversion CMRs help ensure that the colors on your monitor are as close as possible to the ones that are printed. If you move the print job to a different printer, the colors are adjusted again to match the new printer.

In a grayscale system, color conversion CMRs map colors to appropriate shades of gray to produce high-quality black and white images.

Pass-through CMRs are color conversion CMRs that indicate that no color processing is done if the color space of the presentation device is the same as the color space of the CMR. Pass-through CMRs contain no data.

**Link color conversion CMR**

Link color conversion CMRs combine the processing information that is required to directly convert an image from the color space of an input device to the color space of the output device. Essentially, link color conversion CMRs replace a pair of color conversion CMRs.

Converting color images to and from the PCS takes a significant amount of processing resources, in part because the process includes two conversions. Link color conversion CMRs combine the two conversions and make them more efficient. The printer can use the link color conversion CMR to convert colors directly from the color space of the input device to the color space of the output device with the same color fidelity they would have if the printer did both of the conversions. As a result, link color conversion CMRs can improve system performance.

The two types of link color conversion CMRs are:
Link CMRs
Link (LK) CMRs are unique. You cannot create a link CMR yourself and you do not include references to link CMRs in your print jobs. The print system creates and uses link CMRs automatically.

If you use AFP Resource Installer, link CMRs are generated automatically when you create or install a color conversion CMR. As a result, your resource library always contains link CMRs for every combination of color conversion CMRs in audit (input) and instruction (output) processing modes. When link CMRs are created, AFP Resource Installer marks them as capturable, so the printer can save them to be used in other print jobs.

If you do not use AFP Resource Installer, your printer might create link CMRs when it processes print jobs. For example, if you send a print job to an InfoPrint 5000, the printer controller looks at the audit color conversion CMRs that are specified. Then, the print controller looks at the link CMRs that it has available to find one that combines the audit color conversion CMR with the appropriate instruction color conversion CMR. If it does not find one, the print controller creates the link CMR and uses it. The print controller might save the link CMRs that it creates, but they can be removed during normal operation; for example, if the printer runs out of storage or is shut down. If the link is removed, the printer must create a new link CMR the next time it is needed.

When a link CMR is created, the print system evaluates the conversion algorithms to and from the PCS. The system then combines the algorithms, so a data object can be converted directly from one color space to the other without being converted to the PCS.

Device link CMRs
Device link (DL) CMRs use an ICC device link profile to convert directly from an input color space to an output color space without reference to an audit-mode or instruction-mode CMR. An ICC device link profile is a special ICC profile that is used to convert the input device color space to the color space of an output or display device. ICC device link profiles are not embedded in images.

You can create, install, and uninstall device link CMRs yourself. Device link CMRs are referenced in the MO:DCA data stream and take precedence over audit color conversion CMRs. A device link CMR specifies its own rendering intent, which is indicated in the header of the ICC device link profile. This rendering intent overrides any other rendering intent that is active.

The biggest advantage of using device link CMRs is that they preserve the black channel (K component) of the input color space when they are converting from CMYK to CMYK.

Halftone CMRs
Halftone CMRs carry the information that a printer uses to convert print jobs into a pattern of dots that it can put on paper. Halftone CMRs can be used with both color and grayscale print jobs.

Halftone CMRs generally specify the line screen frequency, halftone pattern, and rotation of the halftone that they carry. Device-specific halftone CMRs might also include the printer resolution.
A printer that uses AFP color management to print color or grayscale print jobs must use a halftone CMR to convert the print job into a format that the printer can reproduce in ink or toner. If a halftone CMR is not specified in the print job, the printer applies a default halftone CMR.

**Note:** If you send your color print jobs to an InfoPrint 5000 printer, halftones are applied by the print engine. As a result, the printer ignores halftone CMR requests.

You can associate device-specific halftone CMRs or generic halftone CMRs with print jobs:
- If you know which printer is printing the job, you can associate a device-specific halftone CMR with the print job (or with AFP resources inside the print job). The printer uses the halftone CMR that you specify.
- If you do not know which printer is printing the job, but you want to ensure that it uses a halftone CMR that has certain characteristics, such as a specific line screen frequency, you can associate a generic halftone CMR with the print job.

Because it is difficult to know which halftone CMRs must be used for the current conditions on the current printer, specify halftone CMRs generically and let the printer choose the most appropriate CMR that is available.

**Generic halftone CMRs**
You can use generic halftone CMRs when you want to choose one or more characteristics of the halftone CMR for a print job, but you do not know exactly which halftone CMRs are available.

When a print job specifies a generic halftone CMR, the print server looks in the resource library for halftone CMRs that match the printer device type and model. If the print server finds an appropriate CMR, it sends the device-specific halftone CMR to the printer with the print job. If the print server does not find an appropriate halftone CMR, it sends the generic halftone CMR to the printer.

If a print job that arrives at the printer is requesting a generic halftone CMR, the printer compares the requested characteristics with the available device-specific halftone CMRs. If a match exists, the printer uses the selected device-specific halftone CMR when it processes the print job. If no match exists, the printer uses the halftone CMR whose line screen frequency value is closest to the one requested.

The Color Management Object Content Architecture (CMOCA) defines a variety of generic halftone CMRs, which cover the most common line screen frequencies and halftone types. A print server that supports CMOCA can interpret generic halftone CMRs if it has device-specific halftone CMRs available to it in a resource library. If you use AFP Resource Installer, the generic halftone CMRs are installed in every resource library that you create and populate by using AFP Resource Installer.

Printers that support CMOCA must be able to interpret those generic CMRs and associate them with device-specific halftone CMRs.

**Indexed CMRs**
Indexed (IX) CMRs map indexed colors in the data to presentation device colors or colorant combinations.

Indexed CMRs provide rules about how to render indexed colors. Indexed CMRs apply to indexed colors that are specified by using the highlight color space. They do not apply to indexed colors found within PostScript or other non-IPDS data.
objects. For Indexed CMRs, both instruction and audit processing modes are valid. However, only indexed CMRs with an instruction processing mode are used; indexed CMRs with an audit processing mode are ignored. The tags in the indexed CMR let the CMR use various color spaces in the descriptions. These color spaces can be grayscale, named colorants, RGB, CMYK, or CIELAB.

**Tone transfer curve CMRs**

Tone transfer curve CMRs are used to carry tone transfer curve information for an AFP print job, so you can modify the values of a particular color component and adjust the appearance of some of the colors by increasing or decreasing the amount of ink that is used to emphasize or reduce the effects of dot gain on the final output.

Like halftone CMRs, tone transfer curve CMRs are associated with print jobs specifically or generically. If they are specified generically, the print server looks in the resource library for tone transfer curve CMRs that match the printer device type and model. If the print server finds an appropriate CMR, it sends the device-specific tone transfer curve CMR to the printer with the print job. If the print server does not find an appropriate tone transfer curve CMR, it sends the generic tone transfer curve CMR to the printer.

If a print job that arrives at the printer is requesting a generic tone transfer curve CMR, the printer compares the requested characteristics with the device-specific tone transfer curve CMRs that it has available. If a match exists, the print server or printer uses the selected device-specific tone transfer curve CMR when it processes the print job. If the printer cannot find a good match for the generic tone transfer curve CMR, it ignores the request and uses its default tone transfer curve CMR.

The Color Management Object Content Architecture (CMOCA) defines several generic tone transfer curve CMRs with different appearance values. You can use the appearance values to specify how to print your job in regard to the reported dot gain of the printer.

Generic tone transfer curves can be used to select these appearance values:

- **Dark**  
The output is adjusted to show a dot gain of 33% for a 50% dot.

- **Accutone**  
The output is adjusted to show a dot gain of 22% for a 50% dot.

- **Highlight Midtone**  
The output is adjusted to show a dot gain of 14% for a 50% dot. This appearance might be used to emphasize the brightest part of an image.

- **Standard**  
The output is adjusted enough to account for the effects of dot gain, effectively counteracting the dot gain.

If you use AFP Resource Installer, it installs the generic tone transfer curve CMRs on your system automatically.

**CMR processing modes**

CMR processing modes tell the print system how to apply a CMR to the print data it is associated with. You specify a CMR processing mode whenever you specify a CMR, although not all modes are valid for all CMR types.
Audit processing mode
CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

For example, to take a photograph with a digital camera and then include the photograph in an AFP print job, you can use AFP Resource Installer to:
1. Create a color conversion CMR by using the ICC profile of your camera.
2. Install your photograph in a resource library.
3. Associate the color conversion CMR with the data object, indicating the audit processing mode.

Then, you create a print job that includes the data object. When the print job is processed, the system uses the color conversion CMR to convert the colors in the image into the PCS. The colors can then be converted into the color space of the printer that is printing it.

Instruction processing mode
CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer needs to provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer's website.

If you send a color AFP print job to a printer that supports AFP Color Management, color conversion and tone transfer curve CMRs in instruction processing mode can be associated with the job. When the printer processes the print job, it applies the CMRs in this order:
1. Color conversion CMRs in audit processing mode to convert the resources into the ICC Profile Connection Space (PCS)
2. Color conversion and tone transfer curve CMRs in instruction processing mode to convert the resources into the color space of the printer
3. Halftone CMR in instruction processing mode to convert the job pages from their digital format into the pattern of dots that the printer can produce

In some cases, CMRs that are used as instruction CMRs can be used as audit CMRs. For example, if you send a large print job to a high-speed printer, the images in the print job are converted into the color space of that printer by using a color conversion CMR with the instruction processing mode. However, if you must reprint part of the job on a different printer, the system must convert the print job into the color space of the second printer. In that case, the color conversion CMR of the first printer is used in the audit processing mode to move the images back
into the PCS. Then, the system uses a color conversion CMR of the second printer in instruction mode to convert the images into its color space.

**Link processing mode**

CMRs with the link processing mode are used to link an input color space in the presentation data (sometimes defined by an audit CMR) to the output color space of the presentation device (sometimes defined by an instruction CMR). Only link (LK) and device link (DL) CMRs can be used in link processing mode.

Whenever you install or uninstall audit or instruction color conversion CMRs in your resource library by using AFP Resource Installer or a similar software product, the AFP Resource Installer automatically creates or deletes link (LK) CMRs for every combination of audit and instruction color conversion CMR.

When a print job calls for a specified audit-instruction combination, the print server checks the resource library for a link (LK) CMR for that combination. If the print server finds an appropriate link CMR, it sends the CMR to the printer with the print job. Your printer can use the link (LK) CMRs whenever a print job indicates that it uses a particular combination of audit and instruction CMRs.

If you do not use AFP Resource Installer or a similar program to install your resources, your color printer must either create link (LK) CMRs while it processes your print jobs or convert the colors in your jobs twice, first from the original color space to the PCS and then from the PCS to the color space of the printer.

**CMR creation and installation**

Device manufacturers and groups that support AFP color standards create CMRs that you can use in your color printing systems. You can also create CMRs yourself, based on your needs.

The AFP Consortium, the group that defined the AFP Color Management Object Content Architecture (CMOCA), identified a set of color conversion CMRs that are most often used in audit processing mode. The set includes color conversion CMRs for common color spaces, such as:

- Adobe RGB (1998)
- sRGB
- SMPTE-C RGB
- SWOP CMYK

The standard CMRs are included with AFP Resource Installer, although they are not installed by default. You can install the standard CMRs that you plan to use. In addition,AFP Resource Installer automatically installs all the generic halftone and tone transfer curve CMRs in any resource library you create.

You can download device-specific CMRs for printers, such as the InfoPrint 5000, from the Ricoh Production Print Software (http://rpp.ricoh-usa.com/products/software/)

If you need more CMRs, you can create them using wizards that are provided in AFP Resource Installer. See online help for details about the wizard.

If you use AFP Resource Installer to create a CMR, the software automatically installs the CMR in a resource library. You can also use AFP Resource Installer to install CMRs that you get from your printer manufacturer.
Data objects

Presentation data objects contain a single type of data (such as GIF, JPEG, PNG, and TIFF images) and can be used in your print jobs. These data objects can be placed directly in a page or overlay or can be defined as resources and included in pages or overlays. Using a data object as a resource is more efficient when that object appears more than once in a print job; resources are downloaded to the printer once and referenced as needed.

Data objects can either be included inline with a print job or installed in a resource library by using software such as AFP Resource Installer. If you install your data objects in a resource library, you can associate color conversion CMRs with them.

See "Resource library management" on page 172 for more information about characteristics of resource libraries.

Types of data objects

Image data objects can be stored in a number of different formats, including AFPC JPEG Subset, EPS, GIF, IOCA, PDF, PNG, and TIFF. These image types are device-independent so they can be used by different systems and still be interpreted consistently.

AFPC JPEG Subset (JPEG)
AFPC (AFP Consortium) JPEG Subset files, formerly called JPEG File Interchange Format (JFIF) files, are bitmap image files that are compressed by using Joint Photographic Experts Group (JPEG) compression. As a result, AFPC JPEG Subset files are most commonly referred to as JPEG files. JPEG files most commonly use the file extension .jpg, but can also use .jpeg, .jpe, .jfif, and .jif.

JPEG compression deletes information that it considers unnecessary from images when it converts them. JPEG files vary from having small amounts of compression to having large amounts of compression. The more an image is compressed, the more information is lost. If the image is compressed only once, no noticeable effect typically exists on the image. However, if the image is compressed and decompressed repeatedly, the effects of deleting information become more noticeable.

JPEG compression is commonly used for photographs, especially photographs that are transmitted or displayed on web pages. The compression makes the files small enough to transmit on a network efficiently, but leaves enough information that the image is still visually appealing.

Encapsulated PostScript (EPS)
EPS is a PostScript graphics file format that follows conventions that Adobe Systems defined. EPS files support embedded ICC profiles.

Graphics Interchange Format (GIF)
GIF files are bitmap image files that are limited to a palette of 256 RGB colors. Because of the limited color range that it can contain, GIF is not a good format for reproducing photographs, but it is typically adequate for logos or charts. GIF images are widely used on the internet because they are typically smaller than other image formats. GIF files use the file extension .gif.

Image Object Content Architecture (IOCA)
IOCA is an architecture that provides a consistent way to represent images,
including conventions and directions for processing and exchanging image information. The architecture defines image information independently of all data objects and environments in which it might exist and uses self-identifying terms; each field contains a description of itself along with its contents.

**Portable Document Format (PDF)**

PDF is a standard file format that Adobe Systems developed.

PDF files can be used and stored on various operating systems and contain all the required image and font data. Design attributes in a PDF are kept in a single compressed package.

*Note:* Single-page and multiple-page PDF files can be used as data objects in AFP print jobs.

**Portable Network Graphics (PNG)**

PDF is a standard file format that Adobe Systems developed.

PNG files are bitmap image files that support indexed colors, palette-based images with 24-bit RGB or 32-bit RGBA colors, grayscale images, an optional alpha channel, and lossless compression. PNG is used for transferring images on the internet, but not for print graphics. PNG files use the file extension .png.

**Tagged Image File Format (TIFF)**

TIFF files are bitmap image files that include headers to provide more information about the image.

TIFF files use the file extensions .tif or .tiff. TIFF files support embedded ICC profiles. If an ICC profile is embedded in a file, the characteristics of the input color space are known whenever the file is used; however, the profiles increase the file size. When you save a file in the TIFF format, you can use various compression algorithms.

*Note:* Single-image and multiple-image TIFF files can be used as data objects in AFP print jobs.

Not all printers support all types of data objects.

The embedded ICC profiles in EPS, JPEG, and TIFF files contain the information that a printer uses to convert colors in the image from an input color space into the Profile Connection Space (PCS). The input color space might be an industry-standard space or it can describe the color reproduction capabilities of a device, such as a scanner, digital camera, monitor, or printer.

**Data object creation and installation**

You can use a wide variety of software applications to create or manipulate images to include in print jobs. If you want to store them in central resource repositories, you can use AFP Resource Installer to install them.

**Data object creation**

Most types of data objects are images of some kind. They might be photographs that are taken with a digital camera, charts or diagrams that are generated by a software tool, or digital drawings created with graphics software. Regardless of how images are created, you generally need to manipulate them to include them in print jobs.

The changes include:
Convert the image into a file type that is appropriate for printing. For example, the file types that many graphics applications (such as Adobe Illustrator, CorelDRAW, and Corel Paint Shop Pro) use to store images while you work on them are not appropriate for printing. To use images that you create by using any of those programs, you can save or export those files as a different file type, such as EPS, JPEG, or TIFF.

Make sure that your image files are associated with an appropriate color space or input profile. Follow the instructions that are provided with your graphics software to set up color management, including installing and using ICC profiles for digital cameras and monitors, and customizing color management settings. The instructions also explain how to change the color profile that an image uses and how to save an image with an embedded profile.

Follow the tips and best practices that are provided in “Tips and best practices” on page 173 for creating images and managing them as data object resources.

Data object installation
You can use AFP Resource Installer to install your images in a resource library. AFP Resource Installer includes wizards that can guide you through the process of installing an image as a data object. When you install an EPS, JPEG, or TIFF image with an embedded ICC profile with AFP Resource Installer, you can choose how you want to handle the profile:

- Leave the profile in the file without creating a CMR.
- Leave the profile in the file, but also copy the profile and create a CMR from the copy. Associate the new CMR with the data object.
- Remove the profile from the file (to reduce the file size) and make the profile into a CMR. Associate the new CMR with the data object.

Resource library management
If you store CMRs and data objects in central resource libraries, you must understand some of the characteristics of resource libraries to make sure that your resources are available when and where you need them.

Resource libraries that AFP Resource Installer creates use a resource access table (RAT) as the index of the resource library. The index is stored as a file in the library that it refers to. You must store CMRs in resource libraries that use a RAT. It is also a good idea to store data objects in resource libraries that use a RAT.

When you use AFP Resource Installer to create a resource library, it creates a RAT and stores it in the library. When you install a CMR or data object, AFP Resource Installer updates the RAT with information about the resource. When a print server looks in a resource library for a resource, it first looks in the RAT to see whether the resource is listed.

The print server relies on the RAT; if it is incorrect, the print server cannot find resources in the resource library. As a result, you must always use AFP Resource Installer to manage your resource libraries, including to:

Add CMRs and data objects to a resource library.
Do not copy CMRs or data objects directly into the resource libraries that AFP Resource Installer uses. If you copy CMRs or data objects into these resource libraries, the RAT is not updated so the print server cannot use it to find the CMRs or data objects.

Modify properties of data objects and CMRs listed in the RAT.
Do not directly edit the RAT or any of the files in a resource library. Do
not replace an existing version of a CMR or data object with a new version by copying the new version directly into the resource library; use AFP Resource Installer to update the resource.

**Install CMRs or data objects in a different resource library or replicate a resource library in a different location.**

Do not copy CMRs or data objects from a resource library and store them in another location.

For more information about completing these tasks, see AFP Resource Installer help system.

---

**Tips and best practices**

The following general guidelines about creating and managing images and other color resources can improve the performance of your AFP color printing system.

**Tips for images**

To optimize the performance of your AFP color printing system, follow the following guidelines for creating and including images in print jobs.

When you want to use color images in your print jobs:

- Get the original electronic versions of images instead of scanning existing documents.
  
  Almost unnoticeable specks of color in the background of images that are scanned can greatly increase the size of the image. If you must scan an image, use an image-editing tool to clean up the background as much as possible.
- Save all images in the same standard color space so you need only one input profile for all of them.
  
  Adobe RGB (1998) is the recommended color space for images that are to be printed.
- Flatten multi-layer images (images you create in graphics tools such as Adobe Illustrator and Corel Paint Shop Pro) before you include them in print jobs.
  
  Unflattened images are large and more difficult to work with. Save a copy of the original image for future editing, but flatten the version that you include in your print job.

**Tips for resources**

To optimize the performance of your AFP color printing system, follow the following guidelines for managing color resources.

You can use AFP Resource Installer to:

- Install all the CMRs for your printer in a resource library.
- Install the data objects that you use frequently in a resource library.
- Mark the CMRs and data objects that are reused regularly as non-private, capturable resources so they can be saved on the printer and used for other print jobs without being downloaded every time.

**Note:** This option is not advisable for secure resources, such as signature files.

- Install CMRs and data objects in resource libraries that the print server can access, so they need only to be stored in one place and can be used by all print servers.
• Associate audit color conversion CMRs with data objects that require color management, so the embedded profiles can be removed from the image files.

**CMRTAGFIDELITY Subcommand (FORMDEF)**

The following subcommand on the FORMDEF command describes the exception continuation and reporting rules for Color Management Resource (CMR) tag exceptions. A CMR tag exception is detected when an unsupported CMR tag is encountered in a CMR. Having CMR tag fidelity allows more CMR tags to be added in the future without necessarily causing exceptions in printers that do not support the new tags.

**CMR Tag Fidelity**

```
FORMDEF...

CMRTAGFIDELITY

CONTINUE

NOREPORT

REPORT
```

**FORMDEF**

The full form definition command and all its other non-CMR subcommands are described in [“FORMDEF Command” on page 234](#).

**CMRTAGFIDELITY**

Specify the exception continuation and reporting rules for Color Management Resource (CMR) tag exceptions.

**STOP**

CMR Tag exception rule is “Stop presentation at point of first CMR tag exception and report the exception”.

**CONTINUE**

CMR Tag exception rule is “Do not stop presentation because of CMR tag exceptions and do one of the following:”

**NOREPORT**

Do not report the CMR tag exception to the print server. This is the default if neither NOREPORT or REPORT is coded.

**REPORT**

Report the CMR tag exception.

**Code Example**

In the following example, if a CMR tag exception exists, processing continues and the printer reports the exception to the print server.

```
FORMDEF cmrXm1 REPLACE yes CMRTAGFIDELITY continue report;
```

**DEFINE CMRNAME Subcommand (FORMDEF and all PAGEDEF types)**

```
DEFINE cmr-lname CMRNAME 'cmr-name'
```

**DEFINE CMRNAME**

This command defines a CMR name. A CMR is identified with a name that is specified in a CMR resource object. The name is 73 characters and is based on an architected naming scheme to ensure uniqueness. This naming scheme
includes field components, such as CMR type, manufacturer, device type, and device model number. CMR names are fully described in the *Color Management Object Content Architecture Reference* manual.

**Parameters**

\textit{cmr-lname}

CMR local name. Specify a local name with up to 16 characters. This name is used to reference a CMR with a short user specified name. After you define the CMR name with a local name, it can be used on CMR commands and subcommands.

**Note:** This local name must be unique throughout the entire PPFA source file. That is, the local name cannot be reused in `DEFINE CMRNAME` commands that appear in different form definitions, page definitions, or both if they are in the same source file.

\textit{'cmr-name'}

Specify the full 73 character CMR name. The name is entered as single-byte characters and is translated to UTF-16BE. The `cmr-name` must match the name that is specified for this CMR in the Resource Installer. See “How to copy and paste a name from the AFP Resource Installer” on page 176 for more information.

**Note:** When PPFA does the translation, it uses a fixed table, which means that it translates only from EBCDIC code page 500.

\textit{X16'cmr-name'}

Specify the full CMR name as UTF-16 hex digits. The name is entered in its UTF-16BE encoding, which takes 4 hex digits per character. No translation is needed.

**Code Example**

Three defined CMR names are in the following example:

- Local name \textit{bm1} is defined with a 73 character name in single quotation marks and no text type. The name is translated into UTF-16BE. The example shows the name that is split over 3 PPFA source lines, which might be necessary because of source input record length. The character string can be copied-and-pasted from the AFP Resource installer. See “How to copy and paste a name from the AFP Resource Installer” on page 176.
- Local name \textit{jm4} is specified with X16 data type, which means that the name is entered in UTF-16BE hexadecimal digits.
- Local name \textit{gen1dark} is a generic CMR. It is one of the registered generic CMRs and is entered within single quotation marks, which indicates that it is coded with characters and is translated to UTF-16BE.

CMR \textit{bm1} is defined for the entire print file. It is used to render color for all color resources in the print file unless overridden by a CMR with a more restricted scope. COPYGROUP \textit{cg1} uses CMR \textit{bm1} because it inherits from the form definition. COPYGROUP \textit{cg2} uses CMR \textit{jm4} and COPYGROUP \textit{cg3} uses CMR \textit{gen1dark} to render color resources for the groups of pages they control.

```
FORMDEF cmrX12 REPLACE yes;
DEFINE bm1 CMRNAME
'B111May4HT001.200IBM@0410000'
'PD1940whtg190200000rd00014106000'
'proc00000000000
```
How to copy and paste a name from the AFP Resource Installer

The steps that the user must take to copy and paste the CMR name from the AFP Resource Installer are:

1. On your workstation, start the AFP Resource Installer.
2. Use the Select menu item in the Library menu on the menu bar to open the server resource library where the CMR is located. The server resource library appears in the top pane of the AFP Resource Installer.
3. Select the server resource library in the top pane and use the Expand Selected menu item in the Views menu on the menu bar to show the CMRs that in the server resource library.
4. Select the CMR.
5. Use the Properties menu item in the Actions menu on the menu bar to open the Properties notebook for the selected CMR.
6. Copy the CMR Name that is shown in the Properties notebook.
7. Paste the CMR Name into the PPFA source code.

Note: You might need to break up the name if you are using an operating system that restricts the input source line length to less than 73. For example, PPFA on z/OS restricts the input line to 72 bytes. If the name in the PPFA source code is split over two lines with 42 and 31 characters each, you would first copy the first 42 bytes of the name and paste them into line one, and then copy the remaining 31 bytes of the name into the second line.
Registered Generic Halftone CMRs
The following example is the Registered Generic Halftone CMRs. You can copy the
appropriate lines from this example into PPFA where you would use DEFINE
CMRNAME.

CMR Subcommand (FORMDEF)

CMR Specify a Color Management Resource (CMR), its scope, and its process mode
to be associated with the entire print file or a specific document in the print
file.

Multiple CMR commands are allowed in the form definition.

Parameters
cmr-lname
The CMR local name. This name must be defined with a DEFINE CMRNAME
command.
**scope parameter**

Specify whether the CMR is for the entire print file or a specific document in the print file. This parameter must immediately follow the CMR local name (cmr-lname).

**PRINTFILE**

The scope of this CMR is the entire print file.

**n (document number)**

The scope of this CMR is the specified document.

**processing mode parameter**

Specify the processing mode for the CMR.

**AUDIT**

CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

**INSTR**

CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer's website.

**LINK**

This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

**Code Example**

In the following example, assume that the DEFINE CMRNAME commands define the CMR names with local names (see the DEFINE CMRNAME command for more details).

Two CMRs are defined:

1. The first CMR with local name *Picto550* is an audit CMR for a Picto camera and is to be associated with the entire print file.
2. The second CMR named *dark2* is a generic instruction CMR for the fifth document in the print file.

DEFINE Picto550 CMRNAME
    'Pict1550CC001.001PictV550@@'
    'ES1000000000spac0000000000RGB@'

    DEFINE dark2 CMRNAME
    'ES1000000000spac0000000000RGB@'
'XYZ' ;
DEFINE dark2 CMRNAME '
' ;
FORMDEF cmrXm2 REPLACE yes;
CMR Picto550 PRINTFILE audit;
CMR dark2 5 instr;
COPYGROUP cg1;

NDER Subcommand (FORMDEF)

NDER

Specify the rendering intent (RI) and device output appearance for a print file or individual document.

RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification. Multiple RENDER commands are allowed in the form definition when they are for different scopes.

Device output appearance specifies the appearance to be assumed by the presentation device (printer).

Parameters

scope parameter

Specify whether the rendering intent is for the entire print file or a specific document in the print file. This parameter must immediately follow the RENDER keyword.

PRINTFILE

The scope of this RI is the entire print file.

n (document number)

The scope of this RI is the specified document.

object type parameter

Specify the object type to which the following rendering intent parameters apply. Object type and rendering intent parameter pairs can be repeated to define RI for all object types.

IOCA

The following rendering intent applies to all IOCA objects in the print file or specified document.

OBJC

The following rendering intent applies to all non-OCA object containers in the print file or specified document.

PTOCA

The following rendering intent applies to all PTOCA objects in the print file or specified document.
GOCA
The following rendering intent applies to all GOCA objects in the print file or specified document.

rendering intent parameter
Specify the rendering intent for the preceding object type.

PERCEPTUAL
Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

SATURATION
Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

RELCM
Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

ABSCM
ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

device output appearance parameter
Specify one of a set of architected appearances to be assumed by the presentation device.

DEFAULT
Default appearance. The device assumes its normal appearance. For example, the default appearance of a process-color printer would be to generate full color output.

MONOCH
Monochrome appearance. The device assumes a monochrome appearance such that the device's default color is used for presentation. The device can simulate color values with gray scale by using the default color, or it can simulate color values by substituting the default color, or it can use some combination of the two.

Code Example
Four RENDER subcommands are defined in the following example. Multiple RENDER subcommands for the same scope are not allowed:
1. The first RENDER is for the entire print file, and defines the RI for IOCA objects as perceptual. It also specifies the device output appearance to be monochromatic.
2. The second RENDER is for the document 5, and defines the RI for non-OCA objects as saturation. It also specifies the device output appearance as the device default, which means full color output for a color printer.

3. The third RENDER is for the document 7, and defines the RI for PTOCA objects as media-relative colorimetric. No device output appearance is specified.

4. The fourth RENDER is for the document 9, and defines the RI for all supported objects. It also specifies the device output appearance to be monochromatic.

FORMDEF cmrXm3 REPLACE yes;
   RENDER PRINTFILE IOCA perceptual MONOCH;
   RENDER 5 OBJC saturation DEFAULT;
   RENDER 7 PTOCA relcm;
   RENDER 9 OBJC abscm IOCA relcm PTOCA satur GOCA percp MONOCH;
COPYGROUP cg1;

**CMR Subcommand (COPYGROUP)**

```plaintext
 CMR-cmr-lname . AUDIT
   . INSTR
   . LINK
```

**CMR**

Specify a Color Management Resource (CMR) and its process mode for the collection of pages that are defined by a COPYGROUP. The command is to be placed after the COPYGROUP command for which it is intended.

Multiple CMR commands are allowed in the form definition.

**Parameters**

*cmr-lname*

The CMR local name. This name must be defined with a DEFINE CMRNAME command.

*processing mode parameter*

Specify the processing mode for the CMR.

**AUDIT**

CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

**INSTR**

CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer's website.
LINK

This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

Code Example

The following example shows two copy groups with defined CMRs.

- COPYGROUP picto defines a CMR for presenting digital pictures that are taken with a Picto camera (an audit CMR), and a generic instruction CMR, which is replaced with a device-specific CMR or the default printer CMR.
- COPYGROUP snap defines a CMR for presenting digital pictures that are taken with a Snap camera (an audit CMR), and a generic instruction CMR, which are replaced with a device-specific CMR or the default printer CMR.

FORMDEF cmrX13 REPLACE yes;
DEFINE Picto550 CMRNAME
  'Pict1550CC001.001PictV55000'
  'ES10000000000spac0000000000RGB0'
  'XYZ0000000000'

DEFINE snap1 CMRNAME
  'SnapDSC0CC001.001SNAP000SC-R1'
  'C5000000000spac0000000000RGB0'
  'XYZ0000000000';

COPYGROUP picto ;
CMR picto550 AUDIT;
COPYGROUP snap ;
CMR snap1 AUDIT;

---

**RENDER Subcommand (COPYGROUP)**

**RENDER**

Specify the Rendering Intent (RI) and device output appearance for the collection of pages presented by a COPYGROUP. The command is to be placed after the COPYGROUP command for which it is intended.

**Parameters**

- **object type parameters**
  Specify the object type to which the following rendering intent parameters apply. Object type and rendering intent parameter pairs can be repeated to define RI for all object types.

- **IOCA**
  The following rendering intent applies to all IOCA objects in the pages that are presented by the COPYGROUP.
**OBJC**
The following rendering intent applies to all non-OCA object in the pages that are presented by the COPYGROUP.

**PTOCA**
The following rendering intent applies to all PTOCA objects in the pages that are presented by the COPYGROUP.

**GOCA**
The following rendering intent applies to all GOCA objects in the pages that are presented by the COPYGROUP.

**rendering intent parameter**
Specify the rendering intent for the preceding object type.

**PERCEPTUAL**
Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

**SATURATION**
Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

**RELCM**
Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

**ABSCM**
ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

**device output appearance parameter**
Specify one of a set of architected appearances to be assumed by the presentation device.

**DEFAULT**
Default appearance. The device assumes its normal appearance. For example, the default appearance of a process-color printer would be to generate full color output.

**MONOCH**
Monochrome appearance. The device assumes a monochrome appearance such that the device's default color is used for presentation. The device can simulate color values with grayscale by using the default color, or it can simulate color values by substituting the default color, or it can use some combination of the two.
Code Example

The following example shows two copy groups, one with IOCA RI defined and monochromatic appearance, and one with RI for all object types and the device default appearance.

```
FORMDEF cmrXm4 REPLACE yes;
COPYGROUP cg1;
  RENDER IOCA percp MONOCH ;
COPYGROUP cg2;
  RENDER DEFAULT OBJC abscm IOCA relcm PTOCA satur
     GOCA percp;
```

CMR Subcommand (PAGEFORMAT)

```
CMR-cmr-lname AUDIT
    INSTR
    LINK
```

CMR

Associate a Color Management Resource (CMR) with pages that are presented by a PAGEFORMAT. The command is to follow the PAGEFORMAT command. Multiple CMR commands are allowed in a PAGEFORMAT.

Parameters

`cmr-lname`

The CMR local name. This name must be defined with a DEFINE CMRNAME command.

`processing mode parameter`

Specify the processing mode for the CMR.

**AUDIT**

CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

**INSTR**

CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer's website.
Code Example

In the following example, two CMRs are defined for all pages that are presented with PAGEFORMAT pf5:

1. The first CMR named mycmr is an audit CMR for a digital camera.
2. The second CMR named dark1 is a generic instruction CMR.

```plaintext
 DEFINE mycmr CMRNAME 'Pict1550CC001.001PictV550@@
 'ES1000000000spac0000000000RGB0
 'XYZ00000000';

 DEFINE dark1 CMRNAME
 '00000000Tgeneric0000000000000000000000000000000000000000000000000000000000000000'
 'dark00000000000000000000000000000000';

PAGEDEF cmrXm5 REPLACE yes;

PAGEFORMAT pf5;
 CMR mycmr AUDIT;
 CMR dark1 INSTR;
 PRINTLINE;

PAGEFORMAT pfX;
 PRINTLINE;
```

**RENDER Subcommand (in a PAGEFORMAT)**

```
RENDER
  IOCA
  PERCEPTUAL
  OBJC
  SATURATION
  PTOCA
  RELCM
  GOCA
  ABSCM
```

**RENDER**

Specify the page or overlay scope rendering intent (RI) for the pages that are formatted by this PAGEFORMAT. The RENDER command must follow the PAGEFORMAT command but precede the PRINTLINE, LAYOUT, or XLAYOUT commands.

RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.

**Parameters**

- **object type parameter**

  Specify the object type to which the following rendering intent parameters apply. Object type and rendering intent parameter pairs can be repeated to define RI for all object types.

  - **IOCA**

    The following rendering intent applies to an IOCA object in the page or overlay that are presented by the PAGEFORMAT.

  - **OBJC**

    The following rendering intent applies to a non-OCA object container in the page or overlay that are presented by the PAGEFORMAT.
PTOCA

The following rendering intent applies to a PTOCA object in the page or overlay that are presented by the PAGEFORMAT.

GOCA

The following rendering intent applies to a GOCA object in the page or overlay that are presented by the PAGEFORMAT.

rendering intent parameter

Specify the rendering intent for the preceding object type.

PERCEPTUAL

Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

SATURATION

Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

RELCM

Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

ABSCM

ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

Code Example

Four different pages (PAGEFORMATs) with RENDER commands are defined in the following example:

1. The first page RENDER command defines the RI for IOCA objects as perceptual.
2. The second page RENDER command defines the RI for non-OCA objects as saturation.
3. The third page RENDER command defines the RI for PTOCA objects as media-relative colorimetric.
4. The fourth page RENDER command defines the RI for all supported objects.

```
PAGEDEF cmrXm6  REPLACE yes;

PAGEFORMAT pf6a;
  RENDER IOCA perceptual;
  PRINTLINE;

PAGEFORMAT pf6b;
  RENDER OBJC saturation;
  PRINTLINE;
```
OBJECT Command (Traditional, Record Format, XML)

OBJECT

The full OBJECT command and all its other non-CMR subcommands are described in “OBJECT Command” on page 385.

OBTYPE

Specifies the object type.

PSEG

No change, see “PSEG” on page 389.

GOCA

No change, see “GOCA” on page 389.

BCOCA

No change, see “BCOCA” on page 389.

IOCA

No change, see “IOCA” on page 389.

OTHER

No change, see “OTHER” on page 389.

OBID

No change, see “OBID” on page 389.

RENDER

Subcommand on the OBJECT command to specify the rendering intent (RI) for an object within a page definition.
RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.

Notes:
1. Rendering intent on a BCoca object is fixed as media-relative colorimetric (RELCM), so RENDER does not need to be coded for a BCoca object. But if you do specify something other than RELCM, PPFA flags an error.
2. A page segment (PSEG object type) can contain many object types. If you specify RENDER for a PSEG, PPFA sets that rendering intent type for all object types in the page segment.

rendering intent parameter
Specify the rendering intent for the preceding object type.

PERCEPTUAL
Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

SATURATION
Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

RELCM
Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered concerning the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

ABSCM
ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered concerning only the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

OB2CMR
Specify a Color Management Resource (CMR) and its process mode for a data object within the page definition. CMRs are secondary objects when used at this level. Multiple OB2CMR subcommands are allowed on the OBJECT command.

cmr-lane
The CMR local name. This name must be defined with a DEFINE CMRNAME command.

processing mode parameter
Specify the processing mode for the CMR.
AUDIT
CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

INSTR
CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer’s website.

LINK
This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

NOPRELOAD or PRELOAD
All specified secondary resources are kept. If you want the CMR object to be preloaded before running this job, specify it here.

Code Example
In the following example, an object with CMR is defined. The LAYOUT commands in the example place the object on the page. The CMR name is defined and referenced by the CMR local name. See the DEFINE CMRNAME command for examples and instructions on defining CMR names.

```
PAGEDEF cmr89 replace yes;
  FONT varb gt10 ;  /*Variable data */
  SETUNITS LINESP .25 in ; /* Line spacing */

DEFINE srgb CMRNAME
"sRGBicc_CC001.00000000000000000000000000000000000000000000000000000000000000000000";

Object ocl obxname 'Flowers_with_sRGB_profile'
obtype other obid 23 OBKEEP
ob2cmr srgb audit

PAGEFORMAT rept1 TOPMARGIN 1 in BOTMARGIN 2 in;
  LAYOUT 'startpage' BODY NEWPAGE POSITION 1 in NEXT
  font varb
  object ocl 0 in 3 in obsize 6.5 in 8.5 in;
  LAYOUT 'basicline' BODY POSITION SAME NEXT font varb;
```
FIELD command (All Page Definition Types)

BARCODE CMR subcommand

FIELD...BARCODE... cmr-lname AUDIT INSTR LINK ...

FIELD
The full FIELD command and all its other non-CMR subcommands are described in “FIELD Command” on page 319.

Subcommand

CMR
Specify a Color Management Resource (CMR) and its process mode for the barcode object that is presented.

cmr-lname
The CMR local name. This name must be defined with a DEFINE CMRNAME command.

Note: This parameter must immediately follow the CMR keyword.

processing mode parameter
Specify the processing mode for the CMR.

AUDIT
CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

INSTR
CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer’s website.

LINK
This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA
command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

**Code Example**

In the following example, 2 bar codes are defined with CMRs specified. The bar codes are defined for traditional, record format and XML page definitions.

**Note:** The DEFINE CMRNAMEs for mycmr and dark1 are used in each page definition but defined only once. Page definitions that are in the same source file can define a local CMR name only once because a DEFINE CMRNAME definition is global for all page definitions and form definitions in the same source code file.

```plaintext
DEFINE mycmr CMRNAME 'Pict1550CC001.001PictV550@@CS@@@@@@@@@@@@spac@@@@@@@@@@@RGB@XYZ@@@@@@@@@';
DEFINE dark1 CMRNAME '@@@@@@@@TCgeneric@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@dark@@@@@@@@@@@@@@@@@@@@@@';

/* Traditional Pagedef */
PAGedef cmr10P REPLACE yes;
 PRINTLINE;
   FIELD Start 1 Length 20
   BARCODE TYPE code39 MOD 1
   CMR myCMR audit;
   FIELD Start 21 Length 40
   BARCODE TYPE code39 MOD 1
   CMR dark1 instr;

/* Record Layout Pagedef */
PAGedef cmr10L REPLACE yes;
 Font f1;
 LAYOUT 'l1';
   FIELD Start 1 Length 20
   BARCODE TYPE code39 MOD 1
   CMR myCMR audit;
   FIELD Start 21 Length 40
   BARCODE TYPE code39 MOD 1
   CMR dark1 instr;

/* XML Pagedef */
PAGedef cmr10X REPLACE yes;
 Font f1 TYPE ebcdic;
 XLAYOUT QTAG 'x1';
   FIELD Start 1 Length 20
   BARCODE TYPE code39 MOD 1
   CMR myCMR audit;
   FIELD Start 21 Length 40
   BARCODE TYPE code39 MOD 1
   CMR dark1 instr;
```

**EXTREF Command**

The EXTREF command specifies resources that are to be mapped in the page. It is a way in PPFA to map objects that would not otherwise be mapped. If an object contains another mapped object, the contained object must be mapped, but PPFA does not automatically map that object.
The full EXTREF command and all its other non-CMR subcommands are described in "EXTREF Command" on page 315.

**Subcommands**

**OB2CMR**

```
OB2CMR-lname AUDIT INSTR LINK
```

Specify a Color Management Resource (CMR) and its process mode for a data object specified within an included object. CMRs are secondary objects when used at this level. An object that is specified here is mapped with "object" scope.

- **lname**
  - The CMR local name. This name must be defined with a CMRNAME command.

- **processing-mode-parameter**
  - Specify the processing mode for the CMR.

**AUDIT**

CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

**INSTR**

CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer’s website.

**LINK**

This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

**OB2R**
Specify a secondary object to be mapped.

If an included object contains a reference to one or more secondary objects, you must identify them here. Specify the internal name for the secondary resource as specified in the included resource. If the internal name contains special characters, such as periods or blanks, then quotation marks must surround the name.

\( \text{i2name} \)

- Unquoted name up to 250 characters long is folded to uppercase and translated into EBCDIC if necessary.
- Quoted name up to 250 characters long is accepted as-is with no case folding or translation.
- Quoted name with a C for Character is treated the same as a quoted name of up to 250 characters. No folding or translation is done.
- Quoted name with an E for EBCDIC entered with up to 250 characters is accepted as-is if on an EBCDIC operating system or translated to EBCDIC if on an ASCII operating system. The translation is made with no case folding.
- Quoted name with an A for ASCII entered with up to 250 single-byte characters is accepted as-is if on an ASCII operating system or translated to ASCII if on an EBCDIC operating system. The translation is made with no case folding.
- Quoted name with an X for Hexadecimal entered with up to 500 hexadecimal characters. The characters are translated to hexadecimal, but no assumptions of data type are made.

\( \text{OB2XNAME} \) \( x2name \)

Specifies the external name for a secondary resource object. The name can be up to 250 characters. If the name contains special characters or blanks, then quotation marks must surround the name.

\( x2name \)

- Unquoted name up to 250 characters long is folded to uppercase and translated into EBCDIC if necessary.
- Quoted name up to 250 characters long are accepted as-is with no case folding or translation.
C'x2name'
Quoted name with an C for Character is treated the same as a quoted name up to 250 characters. No folding or translation is done.

E'x2name'
Quoted name with an E for EBCDIC entered with up to 250 single-byte characters is accepted as-is if on an EBCDIC operating system or translated to EBCDIC if on an ASCII operating system. The translation is made with no case folding.

A'x2name'
Quoted name with an A for ASCII entered with up to 250 single-byte characters are accepted as-is on an ASCII operating system or translated to ASCII if on an EBCDIC operating system. The translation is made with no case folding.

X'hhhh'
Quoted name with an X for Hexadecimal entered with up to 500 hexadecimal characters. The characters are translated to hexadecimal, but no assumption of data type is made.

U8'x2name'
Quoted name with a U8 for UTF-8 entered with up to 250 single-byte characters are translated to UTF-8.

X8'hhhh'
Quoted name with an X8 for UTF-8 HEX entered with up to 500 single-byte hexadecimal characters are translated to hexadecimal and assumed to be data type UTF-8. A multiple of 2 hexadecimal characters must be entered.

U16'x2name'
Quoted name with a U16 for UTF-16 entered with up to 125 single-byte characters are translated to UTF-16.

X16'hhhh'
Quoted name with an X16 for UTF-16 HEX entered with up to 500 single-byte hexadecimal characters are translated to hexadecimal and assumed to be data type UTF-16. A multiple of 4 hexadecimal characters must be entered.

\begin{verbatim}
OB2ID n | type-name
\end{verbatim}

Component type identifier for secondary resource uses an object type number as specified in Object type list adjustments. Use an object type number from the Component-ID column or a type name from the Type-Name of the following table:

\begin{table}[h]
\centering
\caption{Object Types that can be referenced as Secondary Resources}
\begin{tabular}{|l|c|l|}
\hline
Type-Name & Component-ID & Description of OID Type-Name \\
\hline
PDFRO & 26 & PDF Resource Object (new) \\
RESCLRPRO & 46 & Resident Color Profile Resource Object \\
IOCAF545RO & 47 & IOCA FS45 Resource Object Tile (new) \\
\hline
\end{tabular}
\end{table}

Page Printer Formatting Aid: User's Guide

194
**Code Example**

In the following example, the CMR `rtvc` is mapped in the Object Environment Group (OEG) of an object that is being included in the page.

Note: If you code `rtvc` with a CMR command (as in the commented out CMR command), it is mapped and is active for the entire page; however, you want it to be active only for the object in whose OEG it is mapped.

```plaintext
DEFINE rvtc CMRNAME 'RevVideoTC001.000@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@'
                       '000000' DNXCNR;
SETUNITS LINESP .25 in ; /* Line spacing */
PAGEFORMAT rept1 TOPMARGIN .25 in;
EXTREF OB2CMR rvtc instr;
```

**DRAWGRAPHIC Command (Record Format and XML)**

**PAGEDEF - CMR and RENDER Subcommands on DRAWGRAPHIC command**

```
DRAWGRAPHIC
  BOX...
  LINE
  CIRCLE
  ELLIPSE
  RENDER
    PERCEPTUAL
    SATURATION
    RELCM
    ABSCM

CMR-cmr-lname
  AUDIT
  INSTR
  LINK
```

**DRAWGRAPHIC**

The full DRAWGRAPHIC command and all its other non-CMR subcommands are described in:

- “DRAWGRAPHIC - BOX Command (Record Format and XML only)” on page 289
- “DRAWGRAPHIC - LINE Command (Record Format and XML only)” on page 296
- “DRAWGRAPHIC - CIRCLE Command (Record Format and XML only)” on page 301
- “DRAWGRAPHIC - ELLIPSE Command (Record Format and XML only)” on page 307

**Subcommands**

**RENDER**

Subcommand on the DRAWGRAPHIC command to specify the rendering intent (RI) for the graphics object being presented.

RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.
rendering intent parameter
Specify the rendering intent for the defined graphic (GOCA) object.

**PERCEPTUAL**
Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

**SATURATION**
Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

**RELCM**
Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

**ABSCM**
ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

**CMR**
Specify a Color Management Resource (CMR) and its process mode for a graphics object within the page definition.

*cmr-lname*
The CMR local name. This name must be defined with a DEFINE CMRNAME command.

**Note:** This parameter must immediately follow the CMR keyword.

**processing mode parameter**
Specify the processing mode for the CMR.

**AUDIT**
CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

**INSTR**
CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.
The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer's website.

**LINK**

This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

**Code Example**

The following examples show how to define CMRs and rendering intent for graphics objects. Rendering intent and a CMR are defined for Record Format and XML page definitions, which are the only two page definitions types for which DRAWGRAPHIC commands are legal.

```plaintext
DEFINE mycmr CMRNAME
  'Pict1550CC001.001PictV55000'
  'CS0000000000spac0000000000RGB0'
  'XYZ0000000000';

PAGEDEF cmr11L REPLACE yes;
  FONT f1;
  LAYOUT '11';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
  RENDER relcm CMR myCMR audit;

PAGEDEF cmr11X REPLACE yes;
  FONT f1 TYPE ebc dic;
  X LAYOUT QTAG 'x1';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
  RENDER relcm CMR myCMR audit;
```

Chapter 8. AFP Color Management  197
Part 3. PPFA Commands and Syntax

This section contains information about:

- PPFA command syntax
- Form definition commands
- Page definition commands
Chapter 9. PPFA Command Syntax

PPFA controls are made up of four elements: commands, subcommands, parameters, and literals.

- **Commands** are controls that represent the major functions of PPFA and are separated from other commands by semicolons. Each command has its own entry in *Chapter 10, “Form Definition Command Reference,” on page 207* and in *Chapter 11, “Page Definition Command Reference,” on page 271*.
- **Subcommands** fall within commands and specify the function of that command.
- **Parameters** specify the values for one subcommand.
- **Literals** consist of fixed text included in a field definition or as constant data for comparison in a conditional processing definition.

Rules for Creating a PPFA Command Stream

When you create a PPFA command stream, follow these rules:

- Before PPFA processes the commands, it converts lowercase characters into uppercase characters, except the ones in literals. Thus, it does not discriminate between uppercase and lowercase characters. For example, `OVERLAY abc` and `overlay ABC` produce the same results because both `overlay` and `abc` are converted to uppercase.
- User names for form definitions and page definitions must not be the same as PPFA command names and subcommand names, which are reserved words. For a list of the reserved words, see *Appendix F, “PPFA Keywords,” on page 561*. For example, `REPEAT` or `CHANNEL` must not be form-definition names.
- The subcommands that are governed by a command can be entered in any order; however, the name of a font or form definition, for example, must come immediately after the object that is named. Parameters that are defined in a subcommand must be entered immediately after the subcommand.
- Commands must end with a semicolon.
- A command or subcommand can start in any column and can continue on the next line without a continuation indicator.
- More than one form definition and page definition can be specified in a job stream.
- PPFA does not check or set default values for items that depend on printer hardware.

Token Rules

Tokens are character strings, within a set of PPFA commands, that PPFA recognizes as units. Tokens include:

- Both local names and user-access names for fonts, form definitions, page definitions, overlays, and suppressions
- Commands
- Subcommands
- Parameters
- Literals
- Special characters
The only PPFA element that is not a token is a blank. A token cannot be split between two lines.

To create a token, you must separate a string from the previous token by either a special character or a blank. See the list of special characters in "Character Set." Thus, A+B is the same as A + B because + is a special character. But AB is not the same as A B. The blank in A B creates two tokens.

**Character Set**

The four types of characters are alphabetic, numeric, blank, and special. Characters of each type are as follows:

**PPFA alphabetic characters**

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

# @ $

**PPFA numeric characters**

0 1 2 3 4 5 6 7 8 9

**Blank character**

Character code of X'20' in ASCII (which is the data stream that is used for creating the form definition or page definition).

Note: In EBCDIC data, the blank character has a character code of X'40'.

**PPFA special characters**

. ( *) - \ ' = ; / &

**EBCDIC shift-out and shift-in codes**

X'0E', the shift-out (SO) code

X'0F', the shift-in (SI) code

Other character codes are also allowed within comments and literals. See "Comments" on page 203 and "Literals" on page 204 for details of what can be included.

**Command Delimiters**

A command always ends with a semicolon. One command can extend over several lines and does not end until a semicolon appears.

**Blanks and Blank Lines**

Blanks and blank lines can occur anywhere and have no effect on the processing of PPFA. The ; is the command delimiter.

**Names**

The maximum number of alphanumeric characters in a PPFA name varies. Table 8 shows the number of characters that are allowed in the PPFA names.

<table>
<thead>
<tr>
<th>Type of Name</th>
<th>Number of Characters Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Definition</td>
<td>1 – 8</td>
</tr>
<tr>
<td>COPYGROUP</td>
<td></td>
</tr>
<tr>
<td>DEFINE CMRNAME (local name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>DEFINE CMRNAME (user-access name)</td>
<td>73</td>
</tr>
</tbody>
</table>

---

202  Page Printer Formatting Aid: User's Guide
Table 8. Character Length for PPFA Names  (continued)

<table>
<thead>
<tr>
<th>Type of Name</th>
<th>Number of Characters Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMDEF</td>
<td>1 – 6</td>
</tr>
<tr>
<td>OVERLAY (local name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>OVERLAY (user-access name)</td>
<td>1 – 6</td>
</tr>
<tr>
<td>SUPPRESSION</td>
<td>1 – 8</td>
</tr>
</tbody>
</table>

Page Definition

<table>
<thead>
<tr>
<th>Type of Name</th>
<th>Number of Characters Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARCODE</td>
<td>1 – 8</td>
</tr>
<tr>
<td>CONDITION</td>
<td>1 – 8</td>
</tr>
<tr>
<td>DEFINE COLOR</td>
<td>1 – 10</td>
</tr>
<tr>
<td>DEFINE CMRNAME (local name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>DEFINE CMRNAME (user-access name)</td>
<td>73</td>
</tr>
<tr>
<td>DEFINE QTAG (local name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>DEFINE QTAG (XML tag names)</td>
<td>1 – 125</td>
</tr>
<tr>
<td>DOFONT (local name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>DOFONT (user-access name)</td>
<td>1 – 125</td>
</tr>
<tr>
<td>FONT (local name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>FONT (user-access name)</td>
<td>1 – 6</td>
</tr>
<tr>
<td>OBJECT (local-name)</td>
<td>1 – 16</td>
</tr>
<tr>
<td>OBJECT (external name) (See note)</td>
<td>1 – 250</td>
</tr>
<tr>
<td>Secondary OBJECT (internal name) (See note)</td>
<td>1 – 250</td>
</tr>
<tr>
<td>Secondary OBJECT (external name) (See note)</td>
<td>1 – 250</td>
</tr>
<tr>
<td>OVERLAY</td>
<td>1 – 6</td>
</tr>
<tr>
<td>PAGEDEF</td>
<td>1 – 6</td>
</tr>
<tr>
<td>PAGEFORMAT</td>
<td>1 – 8</td>
</tr>
<tr>
<td>SEGMENT</td>
<td>1 – 6</td>
</tr>
</tbody>
</table>

Note: This name is further restricted to 250 bytes. For data type UTF-16, the maximum is 125 characters or less if surrogate characters are required. Some operating systems have a limit of 8 characters.

Comments

Programmer comments that are used to document PPFA command streams are allowed anywhere within the command stream. Comments must be enclosed with the delimiters /* and */. A comment is allowed anywhere that a blank is allowed and can continue for any number of lines.

The following example shows the available variations in comment formats:

```plaintext
FIELD /* comment */ FONT GT10 /* comment,
   multiline comment,
   more comment */ START * + 10 LENGTH 5 ;
FIELD LENGTH 10 ; FIELD START * + 10 LENGTH 15 ;
```
2. For VSE, a comment must not start at the beginning of the line. A /* specified as the first 2 bytes of a record in PPFA running under VSE is interpreted as the end of system input.

3. Double-byte character codes in comments must be enclosed within SO (X'0E') and SI (X'0F') on EBCDIC operating systems.

**Literals**

A literal is any string that is specified in single quotation marks. Literals can be used within a:
- TEXT subcommand to create fixed text for a page definition
- WHEN subcommand to define constant text for comparison

Literals can contain any characters in any position, except the ones that have special syntactic meanings. Single quotation marks can be used within a literal only if they are entered in pairs (''). PPFA translates a pair of single quotation marks into one quotation mark. For example, 'JOAN''S' yields JOAN'S.

A literal can continue for any number of lines. For example:

```
TEXT 'THIS IS ' 'A LITERAL' /* The four separated */
     'THE TEXT SPANS' /* text elements will produce*/
     'THREE LINES' ; /* one sequence of text */

TEXT X'0101' /* Hexadecimal literals */
     X'ABAB' /* spanning three lines */
     X'BBBB' ;

TEXT K'100,200' /* kanji numbers */
     K'321,400' ; /* specified sequentially */
```

This example is invalid:

```
TEXT 'THIS IS'
     K'100,200' ; /* Mixing single-byte and
double-byte characters in one
field is not allowed */
```

A double-byte literal must be enclosed within apostrophe shift-out (X'7D0E') and shift-in apostrophe (X'0F7D').

**Numeric Values**

Numeric variables are specified as decimal numbers; up to three decimal places can be specified.

**Units of Measurement**

Numbers that are used to specify dimensions in form definitions and page definitions can be in any of five units of measurement. They are specified in a command stream as follows:

- **IN** Inches
- **MM** Millimeters
- **CM** Centimeters
- **POINTS**
  - Points are a common measurement in printing that are used to measure character height, as in 20-point type. A point is approximately 1/72 inch.
- **PELS**
  - Equates to L-units. The number of pels per inch is a user-specified parameter. The default is 240 pels per inch.
These measurement units can be used in the SETUNITS command:

- **LPI**
  - Lines per inch
- **CPI**
  - Characters per inch

The parameters in PPFA that define a measurement can include any of the first five units of measurement that are shown in the previous list. For example:

```
POSITION 1 IN 1 IN ;
```

or

```
POSITION 1 MM 1 MM ;
```

However, PPFA converts all measurements to logical units (L-units) as the common measurement. (Normally, 1 inch equals 240 L-units, but this number can be changed by the user.) If a fraction exists, the first decimal point is truncated. A SETUNITS command defines a unit of measurement that is to be used as the default for any parameter that does not specify a dimension. This default is in effect until another SETUNITS command is encountered. The following example shows part of a PPFA command stream in which a SETUNITS command sets the units of measurement to one inch for a subsequent POSITION (or OFFSET or LINEONE) subcommand.

```
SETUNITS 1 IN 1 IN ;
```

```
POSITION (or OFFSET or LINEONE) 1 1 ;
```

In the following example, SETUNITS can be used as a multiplier. The SETUNITS command sets two-inch \( x \) and \( y \) default values. The POSITION subcommand values are multiplied by the default values creating a position four inches horizontally and four inches vertically from a given reference point. See “SETUNITS Command” on page 265 for a more detailed explanation.

```
SETUNITS 2 IN 2 IN ;
```

```
POSITION 2 2 ;
```

### Diagram Shorthand

These terms are used in the command definitions:

- **\( x\)-pos**
  - A horizontal position that uses a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 204.

- **\( y\)-pos**
  - A vertical position that uses a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 204.
Chapter 10. Form Definition Command Reference

This section includes:
• Sequence of commands for form definitions
• Form definition commands that are listed alphabetically
• Detailed information about each command
• Descriptions of the applicable subcommands and parameters for each command

Sequence of Commands for Form Definitions

[SETUNITS ...]
FORMDEF
[SUPPRESSION ...]
[DEFINE CMRNAME...]
[CMR...]
[CMTAGFIDELITY...]
[RENDER...]
[COPYGROUP ]
[CMR ...]
[OVERLAY ...]
[RENDER ...]
[SUBGROUP ...]

Rules:
1. SUPPRESSION commands must be specified immediately after FORMDEF commands. The exception is the SETUNITS command (see note 5).
2. One file can contain multiple sets of form definitions.
3. OVERLAY and SUBGROUP commands must be specified under their associated COPYGROUP command. The OVERLAY commands must be specified immediately after a COPYGROUP command.
   • The OVERLAY command is required only to designate an overlay that is to be kept in the 3800 printer as raster data, or to specify a local name for referencing an overlay in a SUBGROUP command. If you do not code the OVERLAY command, you can still specify an overlay in a SUBGROUP command by using its user-access name.
   • Overlays also can be specified by using the N_UP subcommand of the FORMDEF or COPYGROUP command, or by using the PRINTLINE command in the page definition. If the overlay is specified in one of these ways, it must not also be coded on the OVERLAY or SUBGROUP commands that are shown here. For more information, see "Medium Overlays and Page Overlays" on page 160.
   • The appearance of a misplaced OVERLAY command before the first COPYGROUP command causes a default COPYGROUP to be generated as the first COPYGROUP.
   • If the form definition has only one copy group, the COPYGROUP command can be omitted. The OVERLAY command then follows any SUPPRESSION command.
4. The first COPYGROUP command can be omitted in a form definition if it contains only one copy group and no OVERLAY commands. If it is omitted, the FORMDEF command parameters are used to define the copy group.
5. A SETUNITS command can be placed before any PPFA command. The values set are in effect until the next SETUNITS command.
6. Each command can appear more than once under one FORMDEF command.
7. To do an INSERT finishing task, select a COPYGROUP that specifies the dedicated INSERT bin number from which the pages are to be inserted and apply (typically dummy) print data to that page. Observe that nothing is printed on the inserted page.

Note: The INSERT bin number is printer-specific. See the documentation for the specific printer that is used.

COPYGROUP Command
OVERLAY Subcommand:

PLACE Subcommand:

Notes:
1. The use of the PLACE subcommand indicates enhanced N_UP printing.

Form-size Parameters:

Copy groups are subsets of a form definition. A form definition can contain one or several copy groups. Copy groups are nested within a form definition that follows any SUPPRESSION command. COPYGROUP subcommands have no fixed defaults; if any subcommand is omitted, its value is selected from the corresponding subcommand in the FORMDEF command.

Notes:
1. Subsets of copy groups are called subgroups.
2. If you specified DUPLEX NO anywhere in the copy group, output is simplex regardless of any other DUPLEX subcommand within the same copy group.
3. If a form definition has only one copy group, the COPYGROUP command can be omitted. If omitted, a name is automatically assigned by PPF to the copy group, by using the form definition resource name, including the F1 prefix. All values for the copy group are given the values from the FORMDEF command and subcommands. You need to know this name if you use conditional processing and need to invoke this copy group by name. Copy groups are placed within the form definition in the order in which they are generated.
4. To change copy groups during formatting, use conditional processing.
5. Another way to change copy groups after the resource is stored is to insert an Invoke Medium Map structured field into your print data file (copy groups are known to the print server as medium maps). If no Invoke Medium Map structured field is found and no conditional processing is being done, the first copy group in the form definition is used for the job.
COPYGROUP name

Defines an alphanumeric name of 1 - 8 characters. This name must be unique in a single form definition. If any names are duplicated, PPFA issues an error message and does not create the form definition.

Subcommands

OFFSET

```
OFFSET—rel-x - rel-y
```

Specifies the relative offset of the logical page for both the front and back pages in regard to the media origin. The media origin is printer-dependent. For more information about media origin, see your printer publications.

If you specify offset values for the back of the page, you must also specify the front offset values.

Notes:

1. The OFFSET subcommand does not affect the position of medium overlays.
2. If OFFSET is not specified, the OFFSET default is 0.1 IN 0.1 IN
3. You can specify this offset as negative to crop the top, left, or both of an image.

rel-x

Specifies the relative horizontal offset of the logical page on the front or back side of the copy group relative to the media origin. The valid options for rel-x are described in the SETUNITS command for the horizontal value.

If no unit is specified, a default setting is:
- Taken from the last SETUNITS command
- IN (inch) if no SETUNITS command is issued

rel-y

Specifies the relative vertical offset for the logical page for the front or back side of the page. The valid options for rel-y are described in the SETUNITS command for the vertical value.

Note: The vertical offset for the 3800 must be 0.5 inches or greater.

If no unit is specified, a default setting is:
- Taken from the last SETUNITS command
- IN (inch) if no SETUNITS command is issued

BIN

```
BIN—n
```

```
BIN
```

```
MEDIANAME—qstring
```

```
COMPID—m
```

```
MANUAL—
```

```
ENVELOPE
```

Specifies the paper source. This subcommand must be used only for printers that have more than one paper source.
1 Selects the primary paper source.

2 - 255
Selects another paper source. If the specified bin does not exist on your printer, the default paper source for that printer is used. For more information about paper sources on your printer, see your printer publications.

MANUAL
Selects manual feed as a paper source on those printers that support manual feed. For more information, see your printer publications.

ENVELOPE
Selects an envelope paper source on those printers that support this function. For more information, see your printer publications.

MEDIANAME
Selects a media source by specifying an agreed-upon name for the bin.

qstring
Up to 12 characters within single quotation marks, specifying the media source name. On some printers, this name is preset into the printer; on other printers, it can also be entered into the printer by the user. For a current list of the valid media names, see Appendix G, “PPFA Media Names,” on page 563. See your printer publications for more information.

Notes:
1. BIN selection is overridden by the printer if the form defined to each bin is the same form number. Only the primary bin is selected.

2. The primary source usually contains either letter-size (US) or A4 (ISO) paper. Other paper sources are used for less common paper sizes (such as legal-size) and for special paper (such as colored stock or preprinted letterhead on heavy bond).

3. If duplexing is requested and you select from the front side from one bin and the back side from another bin, a warning message is issued and the printer takes the paper from the bin that is specified on the front side.

COMPID
Selects a bin based on the component ID

m For a current list of component IDs, see Appendix G, “PPFA Media Names,” on page 563. Component IDs 12,288 - 268,435,455 are reserved for the user.

OUTBIN n

Specifies the destination bin number for any pages that are directed by this COPYGROUP. Subgroups in this form definition that do not specify an output bin number inherit this one.

n Output bin number

CONSTANT
CONSTANT

Specifies whether the constant-forms function is on or off and whether
constant form is to be printed on the front or back sides of a sheet.

NO  Specifies that the constant forms function is off.

BACK  Specifies that a constant form is to be printed on the back side without
variable data.

FRONT  Specifies that a constant form is to be printed on the front side without
variable data.

BOTH  Specifies that a constant form is to be printed on both sides without
variable data.

DIRECTION

ACROSS  Specifies that the pages are formatted in the ACROSS printing direction.

DOWN  Specifies that the pages are formatted in the DOWN printing direction.

REVERSE  Specifies that the pages are formatted in the REVERSE printing direction.

If the DIRECTION subcommand is specified, you must specify the PRESENT
subcommand. The default for DIRECTION is determined by the value that is
specified for PRESENT.

The direction default of PORTRAIT is ACROSS; the direction default of
LANDSCAPE is DOWN. If PRESENT and DIRECTION are not specified, the
default is PRESENT PORTRAIT and DIRECTION ACROSS.

PRESENT
Specifies, along with the DIRECTION subcommand, how the data is oriented on printers whose media origin can be changed.

The PRESENT and DIRECTION subcommands are only supported by cut-sheet printers when you specify the N_UP subcommand or the CUTSHEET subcommand with the YES parameter. See Figure 94 on page 142 to Figure 97 on page 144 to determine the effect of the PRESENT and DIRECTION subcommands when you use them with the N_UP subcommand.

**PORTRAIT**
Specifies that the pages are printed in the portrait page presentation, with their short edges at the top and bottom and their long edges at the sides.

**LANDSCAPE**
Specifies that the pages are printed in the landscape page presentation, with their long edges at the top and bottom and their short edges at the sides.

**DUPLEX**

```
+--------+------+
| DUPLEX | NO   |
| NORMAL | TUMBLE |
| RNORMAL| RTUMBLE|
```

Specifies whether printing is done on both sides of the sheet. This subcommand must be used only for page printers that have duplex capability.

**NO**
Duplex printing is not done.

**NORMAL**
Duplex printing is done, with the tops of both sides printed along the same edge for side binding.

**TUMBLE**
Duplex printing is done with the top of one side and the bottom of the other printed along the same edge of the sheet for top binding.

**RNORMAL**
Rotated normal. Duplex printing is done with the tops of both sides printed along the same edge. Used with landscape pages, N_UP 2, and N_UP 3.

**RTUMBLE**
Rotated tumble. Duplex printing is done with the top of one side printed along the same edge of the sheet as the bottom of the other. Used with landscape pages, N_UP 2, and N_UP 3.

**CUTSHEET**

```
+--------+------+
| CUTSHEET | NO   |
| CUTSHEET | YES  |
```

If you are using a cut-sheet printer, this subcommand specifies whether the medium orientation information, which specifies the DIRECTION, PRESENT, or both subcommands, is to be passed to the printer. The default value is NO.
YES
Specifies the rotation data is to be passed.

NO  Specifies the rotation data is not to be passed unless N_UP is coded.

Notes:
1. If you have a continuous form printer, the medium orientation information is passed. If you have a cut-sheet printer and N_UP is coded, the orientation information is passed.
2. If you have a cut-sheet printer and CUTSHEET YES is coded, the orientation information is passed, providing you also have a level of the print server that supports that feature.
3. You must have a printer that allows its media origin to be changed to use this subcommand.

Example: In the following example, the CUTSHEET subcommand is coded on the form definition to give COPYGROUP c1 and c2 CUTSHEET YES behavior and COPYGROUP c3 CUTSHEET NO behavior. The COPYGROUP c1 inherits its behavior from the form definition.

```
FORMDEF cut1 REPLACE YES CUTSHEET YES;
COPYGROUP c1;
COPYGROUP c2 CUTSHEET YES ;
COPYGROUP c3 CUTSHEET NO ;
```

A finishing operation is to be done on this COPYGROUP. This option is to be used only on a document, set of documents, or an entire print file.

SCOPE
Determines to which sheets the finishing operation is applied.

Note: SCOPE can be repeated within a FINISH subcommand, but only one SCOPE of a particular type is allowed in each COPYGROUP command. For example, only one SCOPE BEGCOLL is allowed in a COPYGROUP command.

single sheet scope
Operations with this SCOPE are applied to a single sheet.

SHEET
Single sheet Medium-map level scope. The specified finishing operation is applied to each sheet individually.

collection scope
Collection/Medium-map level scope. All sheets that are generated by this medium map are collected and the specified finishing operations are applied to this collection.
**Note:** Some finishing operation combinations are not compatible. Compatible combinations are dependent upon the presentation-device.

**BEGCOLL**
Begin medium-map level collections, which causes a sheet eject and starts a medium-map-level media collection for the specified operation. If a collection for the same finishing operation is already in progress from a previous medium map, that collection is ended and its specified finishing operation is applied. The media collection started with BEGCOLL continues until:
1. The end of the document is reached.
2. A medium map is invoked that is not CONTINUE COLLECTION for this same operation command.

When a finishing collection is ended for either reason, the specified finishing operation is applied.

**CONTCOLL**
Continue medium-map level collection, which continues a medium-level media collection that was started for the same finishing operation by a previous medium map. The media collection started with CONTCOLL continues until:
1. The end of the document is reached.
2. A medium map is invoked that is not CONTINUE COLLECTION for this same operation command.

When a finishing collection is ended for either reason, the specified finishing operation is applied.

**OPERATION parameters**

**OPERATION**
Specifies the type of FINISH operation and parameters.

**Notes:**
1. Compatible operations can be repeated with a specified SCOPE.
2. Your print server might have a limit on the number of collection operations that are allowed at one time.
3. The default for OPERATION is ZFOLD. It is necessary to code OPERATION only if REFERENCE is coded.

**AFP**
Specifies Advanced Function Presentation (AFP) operations as defined in the *Mixed Object Document Content Architecture Reference* and the *Intelligent Printer Data Stream Reference*.
Sheet Operations

These operations operate on a single sheet of paper and are valid for SCOPE SHEET.

**Cfoldin**

Center Fold In. Specifies that the media is folded inward along the center line that is parallel to the finishing operation axis. After this operation, the back side of the last sheet of the collection is outside. The OPCOUNT and OPPOS parameters are ignored for this operation. Cfoldin is applied to collected media, not to individual media.

**Note**: The data stream pages must already be properly ordered for the Cfoldin operation.

**Cut**

Specifies that a separation cut is applied to the media along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation.

**Fold**

Specifies that the media is folded along the axis of the finishing operation. The folding is done along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation. This operation is applied to collected media, not to individual media.

**Perforate**

Specifies that a perforation cut is applied to the media along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation.

**Punch**

Specifies that one or more holes are to be punched or drilled into the media along the finishing axis. Punch is applied to the collected media, not to individual media.

**Zfold**

Perform a ZFOLD operation along the finishing edge (axis). Z-Folding causes the sheet to first be folded in half inwards (the front side of the sheet is now inside the fold) along a line parallel to the reference edge. The half of the sheet originally furthest from the reference edge is again folded in half outwards along a line parallel to the reference edge. For example, when Z-Folding is applied to an 11 by 17-inch sheet with the reference edge along a short side, the result is an 8.5 by 11 inch fold-out. The OPOFFSET, OPCOUNT, and OPPOS parameters are ignored for this operation. This operation is applied to an individual sheet.

**Note**: REFERENCE is the only parameter that is allowed for Zfold and the only reference edges that are allowed are DEFAULT, TOP, BOTTOM, LEFT, and RIGHT.

---

3. The PAGE scope is obsolete. The SHEET subcommand must be used instead of PAGE subcommand. For compatibility purposes, the PAGE command is accepted as an alias for SHEET.
_afp collection operations_

These operations operate on a collection of sheets and are valid with BEGCOLL and CONTCOLL.

**CFOLDIN**
Center Fold In. Specifies that the media is folded inward along the center line that is parallel to the finishing operation axis. After this operation, the back side of the last sheet of the collection is outside. The OPCOUNT and OPPOS parameters are ignored for this operation. CFOLDIN is applied to collected media, not to individual media.

**Note:** The data stream pages must already be properly ordered for the CFOLDIN operation.

**CORNER**
Specifies that one staple is driven into the media at the reference corner (see REFERENCE parameter). For corner staples, the offset and angle of the staple from the selected corner is device-dependent. The OPCOUNT and OPPOS parameters are ignored for this operation. This operation is applied to collected media, not to individual media.

**CUT**
Specifies that a separation cut is applied to the media along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation.

**EDGE**
Specifies that one or more staples are driven into the media along the axis of the finishing operation. This operation is applied to collected media, not to individual media.

**FOLD**
Specifies that the media is folded along the axis of the finishing operation. The folding is done along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation. This operation is applied to collected media, not to individual media.

**PERFECTBIND**
This operation specifies a type of book binding that glues the sheets of the group together at the reference edge (spine). When you specify PERFECTBIND, the OPOFFSET, OPCOUNT, and OPPOS parameters are ignored.

**PERFORATE**
Specifies that a perforation cut is applied to the media along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation.

**PUNCH**
Specifies that one or more holes are to be punched or drilled into the media along the finishing axis. PUNCH is applied to the collected media, not to individual media.

**RINGBIND**
This operation specifies a type of book binding when the sheets of the group are loosely connected at the reference edge (spine) by first drilling or punching a set of holes along the
reference edge and then inserting a wire pattern through the holes. When you specify RINGBIND, the OPOFFSET, OPCOUNT, and OPPOS parameters are ignored.

**SADDLE (same as SADDLEOUT)**

Specifies that one or more staples are driven into the media along the axis of the finishing operation, which is positioned at the center of the media, parallel to the reference edge (see REFERENCE parameter). The OPOFFSET parameter is ignored for this operation. This operation also includes a fold of the media outward along the finishing operation axis so that the front side of the first sheet in the collection is on the outside of the media collection. This operation is applied to collected media, not to individual media.

**SADDLEIN**

Specifies that one or more staples are driven into the media along the axis of the finishing operation, which is positioned at the center of the media, parallel to the reference edge (see REFERENCE parameter). The OPOFFSET parameter is ignored for this operation. This operation also includes a fold of the media inward along the finishing operation axis so that the front side of the first sheet in the collection is on the inside of the media collection. This operation is applied to collected media, not to individual media.

**Note:** The datastream pages must already be properly ordered for the SADDLEIN operation.

**UP3I parameters**

```
UP3I XType X'FFFF'
XOpn X'hh'
XOpn 'X' 'hhhh'
```

**UP3i**

Specifies that these operations are passed to the printer by using the Universal Printer Pre- and Post-Processing Interface (UP3i) finishing interface as specified in the “Form Finishing Operation Triplet” in the UP3i specification document. UP3i is an open standard intelligent interface that is intended for printers, pre-processors, post-processors, and other related applications.

**Notes:**

1. To use this function, you must have printer server support and an attached UP3i device for the specified operation.

2. The complete UP3i specification document, which includes the “Form Finishing Operation Triplet”, can be viewed at the [AFP Consortium web page](http://www.afpcinc.org).

**UP3i explicit operations**

Specifies the explicit values for the “Finishing Operation Type” and the “Finishing Operation Parameter” that go in the UP3i Form Finishing Operation Triplet.
Notes:

1. PPFA does not check that the XType, XOper, or operation parameters are contextually correct. Therefore, new UP3i operations and parameters can be coded without having to install a new PPFA module. However, it also allows contextually incorrect operation and parameter values to be entered.

2. See Table 9 on page 222 for values of XType and XOper.

**XType**

Explicit Operation Type. Specify in hexadecimal or a decimal equivalent number the Finishing Operation Type. A value of 0 specifies a No Operation/Pass through paper operation. When 0 is coded in this field, the XOper field is ignored. Enter 2 hexadecimal digits or a decimal number less than or equal to 255.

**XOper**

Explicit Operation or Operation Parameter. Specify in hexadecimal or a decimal equivalent number the Finishing Operation Type. A value of X'FFFF' specifies the device default operation for the specified finishing operation for the specified Finishing Operation Type in the XType parameter. Enter 4 hexadecimal digits or a decimal number less than or equal to 65535.

More OPERATION parameters

<table>
<thead>
<tr>
<th>REFERENCE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFERENCE</td>
<td>TOP</td>
</tr>
<tr>
<td>LEFT</td>
<td></td>
</tr>
<tr>
<td>TOPLEFT</td>
<td></td>
</tr>
<tr>
<td>TOPRIGHT</td>
<td></td>
</tr>
<tr>
<td>BOTLEFT</td>
<td></td>
</tr>
<tr>
<td>BOTRIGHT</td>
<td></td>
</tr>
<tr>
<td>OPCOUNT</td>
<td></td>
</tr>
<tr>
<td>OPOFFSET</td>
<td></td>
</tr>
</tbody>
</table>

These operation parameters apply to both AFP and UP3i Operations with the noted exceptions.

**REFERENCE**

Selects the reference edge or corner for the finishing operation. The REFERENCE subcommand is optional and, when omitted, the DEFAULT attribute is the default.

**DEFAULT**

Specifies that the device default edge determines the reference edge.

**TOPLEFT**

Specifies that the reference corner is positioned at the top in the left corner. This REFERENCE parameter can be used only for CORNER operations.

**TOPRIGHT**

Specifies that the reference corner is positioned at the top in the right corner. This REFERENCE parameter can be used only for CORNER operations.
BOTRIGHT
Specifies that the reference corner is positioned at the bottom in the right corner. This REFERENCE parameter can be used only for CORNER operations.

BOTLEFT
Specifies that the reference corner is positioned at the bottom in the left corner. This REFERENCE parameter can be used only for CORNER operations.

TOP
Specifies that the reference is positioned along the top edge.

BOTTOM
Specifies that the reference edge is positioned along the bottom edge.

RIGHT
Specifies that the reference edge is positioned along the right edge.

LEFT
Specifies that the reference edge is positioned along the left edge.

OPCOUNT \( n \)
Use OPCOUNT to request a specific number of finishing operations; valid values are 1-122. Do not specify OPPOS values with OPCOUNT. If OPPOS is specified for corner staple, separation cut, perforation cut, or fold, this OPCOUNT value is ignored. The printer determines the positions of the operations. The default is 0 (zero).

OPPOS \( n \)
Use OPPOS to specify the offset of finishing operations along the finishing operations axis that is measured from the point where the finishing operation axis intersects the bottom edge or left edge of the medium toward the center of the medium. Each consecutive OPPOS parameter is used to position a single finishing operation that is centered on the specified point on the finishing operation axis.

For AFP, the subparameter is an integer value in the range of 0 - 32,767 specified in millimeters.

For UP3i, the subparameter is an integer value in the range of 0 - 999999999 specified in millipoints (1/72000 inch).

Do not specify the unit of measure. Do not specify OPCOUNT when you use OPPOS. If OPPOS is specified for corner staple, fold, separation cut, or perforation cut, the OPCOUNT value is ignored.

OPOFFSET \( n \)
Specifies the offset of finishing operation axis from the reference edge that is measured from the reference edge toward the center of the medium.

For AFP, the subparameter is an integer value in the range of 0 - 32,767 specified in millimeters.
For UP3i, the subparameter is an integer value in the range of 0 - 999999999 specified in millipoints (1/72000 inch). Do not specify OPOFFSET for corner staple or saddle stitch; the corner staple or saddle stitch values are ignored when specified with OPOFFSET.

Table 9 shows how to specify finishing operations.

<table>
<thead>
<tr>
<th>XType Finishing Operation</th>
<th>XType Value</th>
<th>XOper Finishing Operation Parameter</th>
<th>XOper Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Operation / Pass-through paper</td>
<td>X'00'</td>
<td>Not applicable</td>
<td></td>
</tr>
<tr>
<td>Paper Input / Page Interpose (not used for AFP/IPDS)</td>
<td>X'01'</td>
<td>Interpose from bin xx Stock Number</td>
<td>X'0001' - X'00FE'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default Bin/Stock</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Fold</td>
<td>X'03'</td>
<td>Folding parameters from fold catalog</td>
<td>X'100D'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No Fold</td>
<td>X'0000'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Staple / Stitch</td>
<td>X'04'</td>
<td>Corner Staple</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Saddle Stitch In</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Saddle Stitch Out</td>
<td>X'0003'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Edge Stitch</td>
<td>X'0004'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Cut</td>
<td>X'05'</td>
<td>Separation Cut</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perforation Cut</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cross Cut</td>
<td>X'0003'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Trim</td>
<td>X'06'</td>
<td>Front Edge</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 Edge</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Edge</td>
<td>X'0003'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 Edge</td>
<td>X'0004'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Offset / Group Separator / Job Separator</td>
<td>X'07'</td>
<td>Offset to Left</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Offset to Right</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Stack</td>
<td>X'08'</td>
<td>Alternate Offset Stack</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Rotate</td>
<td>X'09'</td>
<td>90° Clockwise</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180° Clockwise</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>270° Clockwise</td>
<td>X'0003'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Punch</td>
<td>X'0A'</td>
<td>Round Hole</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rectangular Hole</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
</tbody>
</table>
Table 9. XType and XOper values (continued)

<table>
<thead>
<tr>
<th>XType Finishing Operation</th>
<th>XType Value</th>
<th>XOper Finishing Operation Parameter</th>
<th>XOper Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind</td>
<td>X'0B'</td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Merge</td>
<td>X'0C'</td>
<td>Handle Most Left Page First</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Handle Most Right Page First</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Banding</td>
<td>X'0D'</td>
<td>Single Band Wrap</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double Band Wrap</td>
<td>X'0002'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Crossing Band Wrap</td>
<td>X'0003'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Shrink Wrap</td>
<td>X'0E'</td>
<td>Shrink Wrap</td>
<td>X'0001'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Device Default</td>
<td>X'FFFF'</td>
</tr>
<tr>
<td>Special Handling</td>
<td>X'F0'</td>
<td>Specific Parameter (undefined by UP3i)</td>
<td>X'0000' - X'FFFF'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not Applicable</td>
<td>X'FFFF'</td>
</tr>
</tbody>
</table>

**Notes:**

1. Your printer must have the appropriate finishing hardware to do finishing operations.
2. The default OPERATION is ZFOLD, and the default REFERENCE is DEFAULT.
3. Your print server might have a limit on the number of collection operations that can be open at one time.
4. For the finishing operation, changing the orientation of the medium presentation space does not change the finishing position. For instance, the finishing reference edge (corner) is not affected by DIRECTION or PRESENT values.
5. If more than one finishing operation is specified, the operations are applied in the order in which they are specified. Identical finishing operations for the same SCOPE are not supported.

The following are examples of finishing operations:

- ZFOLD pages (for which the xyz COPYGROUP is in effect), specifying the left edge of the document as the reference edge:
  ```
  COPYGROUP xyz
  FINISH OPERATION ZFOLD REFERENCE LEFT
  ;
  ```

- ZFOLD pages that specify the default edge of the document:
  - COPYGROUP xyz FINISH;
  - COPYGROUP xyz FINISH OPERATION ZFOLD;
  - COPYGROUP xyz FINISH OPERATION ZFOLD REFERENCE DEFAULT;

- A COPYGROUP finishing command where COPYGROUP 1 begins the finishing collection for corner stapling, folding, and separation cut. COPYGROUP 2 continues the fold, cut, and corner operations and stops all other operations. COPYGROUP 3 continues any corner stapling, begins a new punch and fold group, and stops all other operations.
COPYGROUP 1
FINISH
SCOPE BEGCOLL OPERATION corner REFERENCE topleft
OPERATION fold
OPERATION cut;

COPYGROUP 2
FINISH
SCOPE CONTCOLL OPERATION fold
OPERATION cut
OPERATION corner;

COPYGROUP 3
FINISH
SCOPE CONTCOLL OPERATION corner REFERENCE topleft
SCOPE BEGCALL OPERATION punch
OPERATION fold;

• A COPYGROUP finishing command where COPYGROUP 1 begins a
finishing collection for a punch, separation cut, and corner stapling (by
using the UP3i interface), and stops all other operations in progress.
COPYGROUP 2 continues any UP3i corner stapling, but stops all other
operations in progress. COPYGROUP 3 continues any UP3i corner
stapling, stops all other operations in progress, and begins collecting
sheets to punch and cut.
FORMDEF FinXmp Replace Yes;

COPYGROUP 1
FINISH
SCOPE BEGCOLL OPERATION punch
OPERATION cut
OPERATION UP3i XType 4 XOper 1 REFERENCE topleft

COPYGROUP 2
FINISH
SCOPE CONTCOLL OPERATION UP3i XType 4 XOper 1 REFERENCE topleft

COPYGROUP 3
FINISH
SCOPE CONTCOLL OPERATION UP3i XType X04 XOper X0001 REFERENCE topleft
SCOPE BEGCALL OPERATION AFP punch
OPERATION cut;

• COPYGROUP finishing commands with PRESENT and DIRECTION:
FORMDEF MOGD01 replace yes
PRESENT landscape DIRECTION down;
COPYGROUP cg00
PRESENT portrait DIRECTION across;

COPYGROUP cg01
PRESENT landscape DIRECTION across;

COPYGROUP cg02
PRESENT portrait DIRECTION reverse;

COPYGROUP cg03
PRESENT landscape DIRECTION reverse;

COPYGROUP cg04
PRESENT portrait DIRECTION down;

COPYGROUP cg05
PRESENT landscape DIRECTION down;
Finishing Operation Nesting Rules:
When more than one finishing operation that involves a collection of media is specified for some portion of the print file, a nesting of the operations is defined first by the scope of the operation and second by the order of the operation in the data stream.

Finishing operations with a broader scope are nested outside of finishing operations with a narrower scope. The following scopes are listed in descending order:
1. Print-file level finishing (SCOPE PRINTFILE)
2. Document-level finishing, each document in the print file (SCOPE ALL)
3. Document-level finishing, a selected document in the PRINTFILE (SCOPE n)
4. Medium-map-level finishing, a collection of sheets (SCOPE BEGCOLL)

Finishing Operation Implementation Notes:
1. AFP environments limit the number of finishing operations that can be nested at the medium map (COPYGROUP) level. Check your PSF documentation for these limits.
2. In AFP environments, the nesting of identical finishing operations at the medium-map-level is not supported. Two finishing operations are identical if the OPERATION, REFERENCE, OPCOUNT or OPOSS, and OPOFFSET are the same.
3. For some printers, the JOG function cannot be combined with a finishing operation. In this case, the JOG function is ignored. Check your printer documentation.

\textit{ADJUST} \textit{n}

Establishes the range of horizontal adjustment for the print area on the sheet.

\textit{n} The adjustment range can be set 0 - 20 L-units. After a value is set, it is the maximum amount available in both directions, plus and minus.

Notes:
1. If you specify \textit{ADJUST}, the maximum logical page size (in the horizontal direction) is reduced by the amount you specified here.
2. The \textit{ADJUST \textit{n}} subcommand is used only on the IBM 3800 printers.

\textit{INVOKE}

Specifies where the next page of data is placed when this copy group is activated by conditional processing or by an Invoke Medium Map structured field.
INVOKE SHEET, which is the default, places the next page of data on a new sheet. The NEXT, FRONT, and BACK parameters place the next page in a subsequent partition on the same sheet or, if no partitions are available, on the next sheet. If FRONT or BACK is specified, INVOKE selects only partitions on the front or back.

The print server honors the NEXT, FRONT, and BACK values of the INVOKE subcommand only if the new copy group has the same medium modifications as the previous copy group. Some examples of medium modifications are duplexing, input bin, output bin, page offset, N_UP values, presentation, direction, medium (not page) overlays, text suppression, processing functions, print quality, finishing, jogging, and constant forms control. See the Media Eject Control Triplet (X'45') section in the Mixed Object Document Content Architecture Reference for a full description of the factors that allow a conditional eject to the next partition instead of the next sheet.

If any of these modifications differ, the print server ejects to a new sheet when the copy group is invoked. If you want to change overlays when you are ejecting to a new partition, use page overlays instead of medium overlays. See “Medium Overlays and Page Overlays” on page 160 for information about page and medium overlays.

When you use PLACE subcommands, the NEXT, FRONT, and BACK parameters place the next page by using the next sequential PLACE subcommand that matches the requirement (next, front, or back). For example, if you print by using the second PLACE subcommand of copy group A, and then you change to copy group B, you start with the third PLACE subcommand of copy group B.

A CONSTANT parameter on the PLACE subcommand does not alter the selection process. The selection is complete, even though the selected PLACE subcommand does not place the data. N_UP does the constant modification and continues until it finds a PLACE subcommand that does not specify CONSTANT. The data is placed with this subcommand. Observe that this PLACE subcommand need not match the FRONT or BACK specifications of the INVOKE subcommand.

**SHEET**

Specifies that data is placed in the first selected partition of the sheet.

**NEXT**

Specifies that data is placed in the next selected partition.

**FRONT**

Specifies that data is placed in the next selected front partition.

**BACK**

Specifies that data is placed in the next selected back partition.

**JOG**

Specifies whether a JOG subcommand is sent to the printer when this COPYGROUP is selected by an IMM structured field, or through conditional processing. When the JOG subcommand is sent, a printer either offsets (jogs) or prints copymarks. For cut-sheet printers, or for continuous-forms printers with burster-trimmer-stacker enabled, the JOG subcommand causes the first
sheet that is controlled by this COPYGROUP to be stacked offset from the previous sheets. For continuous forms printers without a burster-trimmer-stacker, the JOG subcommand causes an increment in the copymark that is printed on the carrier strip. JOG subcommands also are sent to the printer at the beginning of each data set or at the beginning of each job, depending on host parameters. For more information about copymarks, see the system programming guide for your host print server.

**YES**

Specifies that a JOG subcommand be sent to the printer. The first sheet that is printed is offset or the copymark is incremented.

**NO**

Specifies that no JOG subcommand be sent to the printer. The first sheet that is printed is not offset; the copymark is not incremented.

**QUALITY**

```
QUALITY—n—
```

Specifies the print quality. This subcommand is recognized only on printers that can produce more than one level of print quality. The default is determined by the printer model. (On some printers, the default can be set at the printer itself.) For more information, see your printer publications.

\[ n \]

You can select a level of print quality by entering any whole number 1 - 10. Higher numbers correspond to higher levels of print quality; lower numbers correspond to lower levels. For more information, see your printer publications.

Print quality is determined by a numerical code in the range of 1 - 254 (hexadecimal X'01' - X'FE'). The codes corresponding to the possible QUALITY parameters are:

- 1 = 15 (X'0F')
- 2 = 40 (X'28')
- 3 = 65 (X'41')
- 4 = 90 (X'5A')
- 5 = 115 (X'73')
- 6 = 140 (X'8C')
- 7 = 165 (X'A5')
- 8 = 190 (X'BE')
- 9 = 215 (X'D7')
- 10 = 240 (X'F0')

**PROCESSING**

```
PROCESSING
```

Specifies more post processing capabilities for selected printers and attached equipment. This option can be used only on a single sheet or collection of sheets. This subcommand expects 1 - 3 of the following keywords:
MEDIA_INFO n

This parameter specifies the ID of fixed medium information that a printer or printer-attached device applies to a sheet. Examples include color plates logos, letter heads, and other fixed images.

The numeric values that can be included are:

0 - 254 These numeric values select a particular fixed medium local ID that the printer or printer-attached device applies to a sheet. One or more IDs can be specified within this range.

255 This value selects all the current fixed medium local IDs that the printer or printer-attached devices applies to a sheet.

PERFORATE

Specifies a perforation cut at one or more fixed locations on the sheet according to the printer or printer-attached device.

CUT

Specifies a separation cut at one or more fixed locations on the sheet according to the printer or printer-attached device.

N_UP

{ 1 | 2 | 3 | 4 }

Specifies the number 1, 2, 3, or 4 of equal-size partitions into which the sheet is divided. See the list of printers that support the N_UP subcommand.

If you do not specify the N_UP subcommand in the COPYGROUP command, the N_UP subcommand from the FORMDEF command is the default for the COPYGROUP command. You can mix N_UP printing and non-N_UP printing by specifying or not specifying the N_UP subcommand in each copy group and by not specifying N_UP in the FORMDEF command.

OVERLAY Subcommand

name

Specifies the user access name (up to 6 characters) of an overlay to be placed with every page in each of the N_UP partitions. You can specify a maximum of 254 OVERLAY subcommands in a copy group.

Notes:
1. The prefix ‘O1’ is not part of the six-character user-access name. The overlay name can be an alphanumeric.
2. This name is not related to names as defined on the OVERLAY command.

rel-x rel-y
Specifies the horizontal and vertical adjustment to the position of the overlay. This adjustment is in addition to any offset values built into the overlay. The x and y values can be positive (+) or negative (-). You can specify them in inches (IN), millimeters (MM), centimeters (CM), POINTS, or PELS. If you do not specify a unit value, PPFA uses the unit value that is specified in the last SETUNITS command or uses a default unit value of inches.

Note: This OVERLAY subcommand cannot be specified if the PLACE subcommand is specified.

PARTITION
Specifies that the overlay is to be placed relative to the partition origin.

OVROTATE
Specifies the rotation of the placed overlay in regard to the x-axis of the page.

Example: Assuming the overlay has (0,0) placement coordinates, this causes page overlay 01x2 to be placed 1.5 inches to the right and 2.7 inches below the beginning of the page and rotated 90 degrees clockwise in regard to the page.

Formdef xmpl
N_UP 1
PLACE 1 FRONT
OVERLAY x2 1.5 in 2.7 in
OVROTATE 90;

PLACE Subcommand

Places a page of data or a constant modification relative to a partition. Each PLACE subcommand specifies the number n of a partition on either the front or back side of the sheet. FRONT is the default, if you do not specify this subcommand. You must specify the same number of PLACE subcommands as the number of partitions on the sheet. The sequence of the PLACE subcommands is the sequence in which incoming pages are placed in the partitions.

Notes:
1. The use of the PLACE subcommand indicates enhanced N_UP printing.
2. The PLACE subcommand is valid only on printers that support enhanced N_UP printing. If PLACE is not specified, pages are placed in partitions in the default partition sequence.
Specifies the numbered partition (1 - 4) into which the page of data is placed. See Figure 94 on page 142 through Figure 97 on page 144 for the locale of each numbered partition.

**FRONT**
- Specifies that this partition is placed on the front side of the sheet.

**BACK**
- Specifies that this partition is placed on the back side of the sheet.

**CONSTANT**
- Specifies that no page data is placed by this PLACE subcommand.

Use CONSTANT when you are placing overlays without user's data or are placing fewer data pages on the sheet than the number of partitions that are specified in the N_UP subcommand.

For an example of using the CONSTANT parameter with overlays and to understand how the ordering of the PLACE subcommand affects overlays, see “Example 3: Enhanced N_UP Asymmetric Pages” on page 158.

**OFFSET**
- Specifies a relative offset of the page horizontally (x) and vertically (y) from the partition origin.

\[ \text{rel-x rel-y} \]

The default value is 0.1 inches for both x and y offsets. This OFFSET parameter overrides any other OFFSET parameters that are specified on the FORMDEF or COPYGROUP command. You can specify the units in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPF A uses the unit value that is specified in the last SETUNITS command or uses a default unit value of inches.

**Note:** You can specify this offset as negative to crop the top, left, or both of an image.

**OVERLAY Subcommand**
- See OVERLAY Subcommand

**ROTATION \{ 0 | 90 | 180 | 270 \}**
- Specifies the clockwise rotation of the page and associated page overlays placed by this PLACE command.

Rotation turns the page and its associated page overlays around their fixed origin points. If you rotate the page without moving its origin point, you might rotate it off the physical medium. To prevent this situation, always offset the page origin to the place you want it to be for the rotated page, as shown in Figure 125 on page 231.
VIEW
Determines whether this N_UP PLACE page is viewable. VIEW is relevant only when the page is being presented on a display. VIEW is ignored if the page is being printed. If VIEW is not coded, it is equivalent to specifying VIEW YES.

YES
Specifies that this N_UP page is viewable and is presented.

NO
Specifies that this N_UP page is not to be presented.

OVERLAY Subcommand
See OVERLAY Subcommand for the syntax. Specifies the user access name (up to 6 characters) of an overlay to be placed with this PLACE subcommand. The overlay is placed relative to the page origin or, if the PARTITION keyword is specified, to the partition origin. You can specify multiple OVERLAY parameters in each PLACE subcommand.

Note: This OVERLAY subcommand cannot be specified if the PLACE subcommand is specified.

rel-x rel-y
Specifies the horizontal and vertical adjustment to the position of the overlay. This adjustment is in addition to any offset values built into the overlay. The $x$ and $y$ values can be positive (+) and negative (-). You can specify them in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPFA uses the unit value that is specified in the last SETUNITS command or uses a default value of inches.

PARTITION
Specifies that the previous offset is from the partition origin. If not present, the offset is from the page origin, which is subject to the OFFSET parameter.

OVROTAPE \{ 0 | 90 | 180 | 270 \}
Specifies the rotation of the placed overlay in regard to the x-axis of the page.

PELSPERINCH $n$
Specifies the Logical Units in pels per inch for this COPYGROUP. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

$n$ Specifies an integer number 1 - 3,276, which determines the Logical Units in pels per inch.

Note: If the L-Units are not specified on the copy group, they are inherited from the form definition. See Figure 126 on page 253 for more information.

Form-size Parameters

Specifies the medium presentation space (also known as the medium size or form length and form width).

Notes:
1. This function requires both printer server and printer support. See your print server and printer documentation.
2. The printer does not adjust the size of your media-presentation space to be larger than the paper size (or what is defined in the printer as the paper size).
3. Some printers (such as the InfoPrint 1145 and the InfoPrint 4100) do not support the IPDS Set Media Size (SMS) command. The form size cannot be set with the form definition. Do not use the XMSIZE and YMSIZE subcommands for the printers that do not support the SMS commands.
4. Other printers (such as the 6400, 4247, and 4230) do not support the Set Media Origin (SMO) command. The media origin does not change. For the 6400, 4247, and 4230 printers form length is always YMSIZE and form width is always XMSIZE.
5. For all other printers, use the settings that are shown in Table 10 on page 233 and Table 11 on page 233. For these other printers, whether the XMSIZE or YMSIZE is form length or form width depends on the medium presentation space orientation, type of form, and NUP setting. The following examples are from Table 10 on page 233 and Table 11 on page 233. See the table for other media combinations.
   - Wide fanfold paper, PRESENT=Landscape, DIRECTION=ACROSS, and no-NUP - The form length is YMSIZE.
   - Narrow fanfold paper, PRESENT=Landscape, DIRECTION=ACROSS, and no-NUP - The form length is XMSIZE.
   - Cutsheet paper, PRESENT=Landscape, DIRECTION=ACROSS, and no-NUP - The form length is XMSIZE.
6. Only two choices exist. If you try one that does not work, try the other. For example, if you try XMSIZE for the form length and it does not create a longer form, use YMSIZE.

XMSIZE

Specifies the medium presentation space along the X-axis (also known as...
the medium's size in the X-direction). If this subcommand is specified on
the FORMDEF command, it becomes the default for all copy groups that
do not specify XMSIZE on the COPYGROUP command. If this
subcommand is not specified on the FORMDEF command, the printer's
current default X-axis becomes the default for all copy groups that do not
specify XMSIZE on the COPYGROUP command.

\[ x \] Enter a number with 0 - 3 decimal places and optional units.

\[ \text{YMSIZE} \]
Specifies the medium presentation space along the Y-axis (also known as
the medium’s size in the Y-direction). If this subcommand is specified on
the FORMDEF command, it becomes the default for all copy groups that
do not specify YMSIZE on the COPYGROUP command. If this
subcommand is not specified on the FORMDEF command, the printer's
current default Y-axis becomes the default for all copy groups that do not
specify YMSIZE on the COPYGROUP command.

\[ y \] Enter a number with 0 - 3 decimal places and optional units.

\[ \text{units} \]
Enter IN for inches, CM for centimeters, MM for millimeters, or PELS for
pels. If \text{units} is not specified, the default is to the most recent setting of the
SETUNITS command or inches if no SETUNITS command is coded.

**Table 10. Form Length (LEN) and Form Width (WID) for Cutsheet and Narrow Fanfold Paper**

| DIRECTION | PRESENT | ACROSS | | DOWN | | REVERSE | | |
|-----------|---------|--------| | | | | | |
| | | Portrait | Landscape | Portrait | Landscape | Portrait | Landscape | |
| | | LEN | WID | LEN | WID | LEN | WID | LEN | WID | LEN | WID | |
| No NUP | Ym | Xm | Ym | Xm | Ym | Xm | Ym | Ym | Xm | Xm | Ym | |
| 1-UP | Ym | Xm | Xm | Ym | Xm | Ym | Xm | Ym | Xm | Xm | Ym | |
| 2-UP | Xm | Ym | Ym | Xm | Ym | Xm | Ym | Ym | Xm | Ym | Xm | |
| 3-UP | Xm | Ym | Ym | Xm | Ym | Xm | Ym | Ym | Xm | Ym | Xm | |
| 4-UP | Ym | Xm | Xm | Ym | Xm | Ym | Xm | Ym | Xm | Xm | Ym | |

**Table 11. Form Length (LEN) and Form Width (WID) for Wide Fanfold Paper**

| DIRECTION | PRESENT | ACROSS | | DOWN | | REVERSE | | |
|-----------|---------|--------| | | | | | |
| | | Portrait | Landscape | Portrait | Landscape | Portrait | Landscape | |
| | | LEN | WID | LEN | WID | LEN | WID | LEN | WID | LEN | WID | |
| No NUP | Xm | Ym | Ym | Xm | Ym | Xm | Ym | Ym | Xm | Xm | Ym | |
| 1-UP | Xm | Ym | Ym | Xm | Ym | Xm | Ym | Ym | Xm | Xm | Ym | |
| 2-UP | Ym | Xm | Xm | Ym | Ym | Xm | Ym | Ym | Xm | Xm | Ym | |
| 3-UP | Ym | Xm | Xm | Ym | Ym | Xm | Ym | Ym | Xm | Xm | Ym | |
| 4-UP | Xm | Ym | Xm | Ym | Xm | Ym | Xm | Ym | Xm | Ym | Xm | |

**Examples:**
FORMDEF FMSZX1 Replace Yes
PRESENT Landscape Direction Across
XMSIZE 8.5 in YMSIZE 11.0 in;
COPYGROUP cp1;
COPYGROUP cp2;
In the first example, the printer is a 4400 thermal printer, which supports both SMS and SMO IPDS commands. The form definition that is named FMSZX1 defines a form length of 8.5 inches and form width of 11.0 inches. 

- In the second example, the printer is a 6400 printer and you want to define the form length. The form definition that is named FMSZX2 defines form length as 17 inches and leaves the form width as the printer default.

If this example is run on z/OS with FORMLEN defined in the JCL, the JCL definition is used.
Chapter 10. Form Definition Command Reference
OVERLAY Subcommand:

```
OVERLAY—name rel-x rel-y PARTITION OVRotate—0
```

PLACE Subcommand:

```
PLACE n FRONT BACK CONSTANT OFFSET rel-x rel-y
```

Notes:
1. The use of the PLACE subcommand indicates enhanced N_UP printing.

Form-size Parameters:

```
XMSIZE x units YMSIZE y units
```

A form definition is a resource that contains all the controls that relate to the physical sheet. A FORMDEF command must be specified when you define a new form definition. When subcommands (except for the REPLACE, PRESENT, and DIRECTION subcommands) are specified, they become the defaults for all COPYGROUP commands nested within this form definition.

**FORMDEF**

Identifies the form definition to be used with the print job.

**name**

Defines an alphanumeric name of 1 - 8 characters for the form definition. When you create a form definition, PPF assigns a prefix of F1 to the name you specify. F1nnnnnn is the external resource name in the form-definition library.

**Subcommands**

**OFFSET**

```
OFFSET—rel-x rel-y
```

Specifies the offset of the logical page for both the front and back pages in regard to the media origin. The media origin is printer-dependent. For more
information about media origin, see your printer publications or Advanced Function Presentation: Printer Information.

If you specify offset values for the back of the page, you must also specify the front offset values.

Notes:
1. The OFFSET subcommand does not affect the position of medium overlays.
2. You can specify this offset as negative to crop the top, left, or both of an image.

\textit{rel-x}

Specifies the relative horizontal offset (negative or positive) of the logical page on the front or back side of the copy group relative to the media origin. The valid options for \textit{rel-x} are described in the SETUNITS command for the horizontal value.

The default unit is:
- Taken from the last SETUNITS command
- IN (inch) if no SETUNITS command is issued
- 0.1 IN

\textit{rel-y}

Specifies the relative vertical offset (negative or positive) for the logical page for the front or back side of the page. The valid options for \textit{rel-y} are described in the SETUNITS command for the vertical value.

The default unit is:
- Taken from the last SETUNITS command
- IN (inch) if no SETUNITS command is issued
- 0.1 IN

\textbf{Note}: The vertical offset for the 3800 must be 0.5 inches or greater.

\textbf{REPLACE}

\begin{center}
\begin{tabular}{l}
\textbf{REPLACE—NO} \hspace{1cm} \textbf{REPLACE—YES} \\
\end{tabular}
\end{center}

Specifies whether this form definition is to replace an existing one with the same resource name in the library.

\textbf{YES}

Replace an existing form definition of the same name in the library if there is one. If a form definition with the same name does not exist in the library, then store this form definition.

\textbf{NO}

Do not replace an existing form definition of the same name. If a form definition with the same name does not exist in the library, then store this form definition.

This is the default.
Specifies which paper source is to be used on printers with more than one paper source. The value range is 1 - 255. (This subcommand must be used only for printers that have more than one paper source.)

**Note:** If you specify the BIN subcommand, you must also specify at least one of the legal parameters.

- **n** An integer number 1 - 255 that is the Media Source ID (also known as the bin number).
- **1** Selects the primary paper source.
- **2 - 255** Selects another paper source. If the specified bin does not exist on your printer, the default paper source for that printer is used. For more information about paper sources on your printer, see your printer publications. Using a value of 100 is the same as specifying MANUAL.

**MANUAL**
Selects manual feed as a paper source on those printers that support manual feed. For more information, see your printer publications.

**ENVELOPE**
Selects an envelope paper source on those printers that support this function. For more information, see your printer documentation.

**MEDIANAME**
Selects a media source by specifying an agreed upon name for the bin. For a list of the valid media names, see [Appendix G, “PPFA Media Names,” on page 563](#).

- **qstring** Up to 12 characters within single quotation marks that specify the media source name. On some printers, this name is preset in the printer; on others, it also can be entered in the printer by the user. See your printer documentation for further information.

**COMPID m**
Selects a bin based on the component ID.

**Note:** For a current list of component IDs, see [Appendix G, “PPFA Media Names,” on page 563](#). Component IDs 12,288 - 268,435,455 are reserved for the user.

**Notes:**
1. BIN selection is overridden by the printer if the form defined to each bin is the same form number. Only the primary bin is selected.
2. The primary source usually contains either letter-size (US) or A4 (I.S.O.) paper. Other paper sources are used for less common paper sizes (such as legal-size) and for special paper (such as colored stock or preprinted letterhead on heavy bond).
3. If duplexing is requested and you select from the front side from one bin and the back side from another bin, a warning message is issued and the printer takes the paper from the bin that is specified on the front side.

**OUTBIN** \( n \)

![OUTBIN diagram]

Specifies the destination bin number for any pages that are directed by this form definition. Copy groups and subgroups in this form definition that do not specify an output bin number inherit this bin number.

\( n \) Specifies the output bin number.

**CONSTANT**

![CONSTANT diagram]

Specifies whether the constant-forms function is on or off and whether constant form is to be printed on the front or back sides of a sheet.

**NO** Specifies that the constant forms function is off.

**BACK** Specifies that a constant form is to be printed on the back side without variable data.

**FRONT** Specifies that a constant form is to be printed on the front side without variable data.

**BOTH** Specifies that a constant form is to be printed on both sides without variable data.

**CUTSHEET**

![CUTSHEET diagram]

If you are using a cut-sheet printer, this subcommand specifies whether the medium orientation information, which is coded by using the DIRECTION, PRESENT, or both subcommands, is to be passed to that printer. Not coding the CUTSHEET subcommand is equivalent to coding CUTSHEET NO.

**NO** Specifies that the rotation data is not to be passed unless N_UP is coded.

**YES** Specifies that the rotation data is to be passed.

**Notes:**

1. As always: If you have a continuous form printer, the medium orientation information is passed. If you have a cut-sheet printer and N_UP is coded,
the orientation information is passed. The default for a COPYGROUP for which no CUTSHEET subcommand is coded is to inherit the behavior of the FORMDEF.

2. **New:** If you have a cut-sheet printer and CUTSHEET YES is coded, the orientation information is passed if you also have a level of print server that supports the CUTSHEET feature.

3. **In all cases:** Before you use this command, you must have a printer that allows its media origin to be changed.

**DIRECTION**

```
DIRECTION

ACROSS

DOWN

REVERSE
```

Determines, along with the PRESENT subcommand, how data is oriented on printers whose media origin can be changed. See the list of printers under the PRESENT subcommand.

If you are printing line data, you usually specify the same value for the DIRECTION subcommand as is specified for the DIRECTION subcommand in the page definition.

**ACROSS**

Specifies that the pages are formatted in the ACROSS printing direction.

**DOWN**

Specifies that the pages are formatted in the DOWN printing direction.

**REVERSE**

Specifies that the pages are formatted in the REVERSE printing direction.

If the DIRECTION subcommand is specified, you must specify the PRESENT subcommand. The default for DIRECTION is determined by the value that is specified for PRESENT.

The direction default of PORTRAIT is ACROSS; the direction default of LANDSCAPE is DOWN. If PRESENT or DIRECTION is not specified, the default is PRESENT PORTRAIT and DIRECTION ACROSS.

Examples of FORMDEF finishing commands with PRESENT and DIRECTION:

```
FORMDEF fd00
  PRESENT portrait  DIRECTION across;

FORMDEF fd01
  PRESENT landscape DIRECTION across;

FORMDEF fd02
  PRESENT portrait  DIRECTION reverse;

FORMDEF fd03
  PRESENT landscape DIRECTION reverse;

FORMDEF fd04
  PRESENT portrait  DIRECTION down;

FORMDEF fd05
  PRESENT landscape DIRECTION down;
```

**DUPLEX**
Specifies whether printing is done on both sides of the sheet. This subcommand must be used only for page printers that have duplex capability.

**NO**  Duplex printing is not done.

**NORMAL**  
Duplex printing is done, with the tops of both sides printed along the same edge for side binding.

**TUMBLE**  
Duplex printing is done with the top of one side and the bottom of the other printed along the same edge of the sheet for top binding.

**RNORMAL**  
Rotated normal. Duplex printing is done with the top of one side printed along the same edge of the sheet as the bottom of the other. Used with landscape pages, N_UP 2, and N_UP 3.

**RTUMBLE**  
Rotated tumble. Duplex printing is done with the tops of both sides printed along the same edge. Used with landscape pages, N_UP 2, and N_UP 3.

**PRESENT**

**PRESENT**  
Specifies, along with the DIRECTION subcommand, how the data is oriented on printers whose media origin can be changed.

The PRESENT and DIRECTION subcommands are only supported by cut-sheet printers when you specify the N_UP subcommand or the CUTSHEET subcommand with the YES parameter. See Figure 94 on page 142 and Figure 97 on page 144 to determine the effect of the PRESENT and DIRECTION subcommands when you use them with the N_UP subcommand.

**PORTRAIT**  
Specifies that the pages are printed in the portrait page presentation, with their short edges at the top and bottom and their long edges at the sides.

**LANDSCAPE**  
Specifies that the pages are printed in the landscape page presentation, with their long edges at the top and bottom and their short edges at the sides.

**FINISH**
Specifies where the media is to be stapled, folded, cut, or perforated.
This option can be used only on a document, set of documents, or an entire print file. Finishing operations are device-dependent; check your printer documentation before you use the FINISH subcommand.

Notes:
1. The FINISH operation is used for printers with finisher attachments.
2. The finishing operation must be specified at least once, and can occur more than once. It specifies finishing operations to be applied to the collected media.
3. If more than one finishing operation is specified, the operations are applied in the order in which they are specified. Identical finishing operations for the same SCOPE are not supported.
4. FINISH positions are not affected by DIRECTION or PRESENT values.
5. Changing the orientation of the medium presentation space does not change the finishing corners or edges.
6. For continuous forms media, the carrier strips are not considered to be part of the physical media.
7. For saddle stitch operation, the staples are placed along the center of the media, parallel to the reference edge. Any offset value is ignored. If no OPCOUNT or OPPOS values are specified, the device default count is used.
8. User-specified OPCOUNT and OPPOS values are ignored for FOLD, CUT, or PERFORATE operations.

SCOPE
Determined how the finishing operation is applied.

Note: SCOPE can be repeated within a FINISH subcommand, but only one SCOPE of a particular type is allowed for each FORMDEF command. For example, only one SCOPE ALL is allowed for each FORMDEF FINISH command.

PRINTFILE
Determine that the specified finishing operations for the OPERATION subcommand are applied to the complete print file, excluding header pages, trailer pages, and message pages.

ALL
Determine that the specified finishing operations for the OPERATION subcommand are applied individually to all documents in the print file.

n
Use the n to apply the finishing operation to a specific document. Use a value of 1 to apply the finishing operation to the first document in a print file. Use the value 2 to apply the finishing operation to the second document in a print file, and so on. The range of values includes 1 - 32,767.

OPERATION Parameters
OPERATION

Specifies the type of finishing operation and parameters for that operation.

Notes:
1. Compatible Operations can be repeated within a specified SCOPE.
2. Your print server can have a limit on the number of collection operations that are allowed to be active at one time.

AFP

Specifies Advanced Function Presentation (AFP) operations as defined in the Mixed Object Document Content Architecture Reference and the Intelligent Printer Data Stream Reference.

CFOLDIN

Center Fold In. Specifies that the media is folded inward along the center line that is parallel to the finishing operation axis. After this operation, the back side of the last sheet of the collection is on the outside. The OPCOUNT and OPPOS parameters are ignored for this operation. CFOLDIN is applied to collected media, not to individual media.

Note: Pages of the datastream must already be properly ordered for this operation.

CORNER

Specifies that one staple is driven into the media at the reference corner (see REFERENCE parameter). For corner staples, the offset and angle of the staple from the selected corner is device-dependent. The OPOFFSET, OPCOUNT, and OPPOS parameters are ignored for this operation. This operation is applied to collected media, not to individual media.

CUT

Specifies that a separation cut is applied to the media along the axis of the finishing operation. The OPCOUNT and OPPOS parameters are ignored for this operation.

EDGE

Specifies that one or more staples are driven into the media along the axis of the finishing operation. This operation is applied to collected media, not to individual media.

FOLD

Specifies that the media is folded inward on the front sheet side of the first sheet of the collection. The folding is done along the axis
of the finishing operation. The OPOFFSET and OPPOS parameters are ignored for this operation. This operation is applied to collected media, not to individual media.

**PERFECTBIND**
This operation specifies a type of book binding that glues the sheets of the group together at the reference edge (spine). When you specify PERFECTBIND, the OPOFFSET, OPCOUNT, and OPPOS parameters are ignored.

**PERFORATE**
Specifies that a perforation cut is applied to the media along the axis of the finishing operation. The OPOFFSET and OPPOS parameters are ignored for this operation.

**PUNCH**
Specifies that one or more holes are to be punched or drilled into the media along the finishing axis. PUNCH is applied to the collected media, not to individual media.

**RINGBIND**
This operation specifies a type of book binding when the sheets of the group are loosely connected at the reference edge (spine) by first drilling or punching a set of holes along the reference edge and then inserting a wire pattern through the holes. When you specify RINGBIND, the OPOFFSET, OPCOUNT, and OPPOS parameters are ignored.

**SADDLE (same as SADDLEOUT)**
Specifies that one or more staples are driven into the media along the axis of the finishing operation, which is positioned at the center of the media, parallel to the reference edge (see REFERENCE parameter). The OPOFFSET parameter is ignored for this operation. This operation also includes a fold of the media outward along the finishing operation axis so that the front side of the first sheet in the collection is on the outside of the media collection. This operation is applied to collected media, not to individual media.

**SADDLEIN**
Specifies that one or more staples are driven into the media along the axis of the finishing operation, which is positioned at the center of the media, parallel to the reference edge (see REFERENCE parameter). The OPOFFSET parameter is ignored for this operation. This operation also includes a fold of the media inward along the finishing operation axis so that the front side of the first sheet in the collection is on the outside of the media collection. This operation is applied to collected media, not to individual media.

**UP3i Parameters**

```plaintext
UP3i XType n X'FFFF' XOper X'hh' More OPERATION parameters
```

**UP3i**
Specifies that these operations are passed to the printer by using the Universal Printer Pre- and Post-Processing Interface (UP3i)
finishing interface as specified in the “Form Finishing Operation Triplet” in the UP3i specification document. UP3i is an open standard intelligent interface that is intended for printers, pre-processors, post-processors, and other related applications. The complete UP3i specification document, which includes the “Form Finishing Operation Triplet” can be viewed at the AFP Consortium web page (http://www.afpcinc.org).

**UP3i explicit operations**

Specifies the explicit values for the “Finishing Operation Type” and the “Finishing Operation Parameter” that go in the UP3i Form Finishing Operation Triplet.

**Note:** See Table 9 on page 222 for values of XType and XOper as defined in the UP3i Specification.

**XType**

Explicit Operation Type. Specify in hexadecimal or a decimal equivalent number the Finishing Operation Type. A value of 0 specifies a No Operation/Pass through paper operation. When 0 is coded in this field, the XOper field is ignored. Enter 2 hexadecimal digits or a decimal number less than or equal to 255.

**XOper**

Explicit Operation or Operation Parameter. Specify in hexadecimal or a decimal equivalent number the Finishing Operation Type. A value of X'FFFF' specifies the device default operation for the specified finishing operation for the specified Finishing Operation Type in the XType parameter. Enter 4 hexadecimal digits or a decimal number less than or equal to 65535.

**More OPERATION Parameters**

**REFERENCE**

Determines the reference corner or edge of the finishing operation.

**DEFAULT**

Specifies that the device default determines the reference corner or edge.

**TOPLEFT**

Specifies that, for the finishing operation, the reference corner is positioned at the top in the left corner. This REFERENCE parameter can be used only for CORNER operations.

**TOPRIGHT**

Specifies that, for the finishing operation, the reference corner
is positioned at the top in the right corner. This REFERENCE parameter can be used only for CORNER operations.

**BOTRIGHT**

Specifies that, for the finishing operation, the reference corner is positioned at the bottom in the right corner. This REFERENCE parameter can be used only for CORNER operations.

**BOTLEFT**

Specifies that, for the finishing operation, the reference corner is positioned at the bottom in the left corner. This REFERENCE parameter can be used only for CORNER operations.

**TOP**

Specifies that, for the finishing operation, the reference edge is positioned at the top.\(^4\)

**BOTTOM**

Specifies that, for the finishing operation, the reference edge is positioned at the bottom.\(^5\)

**LEFT**

Specifies that, for the finishing operation, the reference edge is positioned at the left.\(^5\)

**RIGHT**

Specifies that, for the finishing operation, the reference edge is positioned at the right.\(^5\)

**OPCOUNT n**

Use OPCOUNT to request a specific number of finishing operations; valid values are 1 - 122. Do not specify OPPOS values with OPCOUNT. If OPPOS is specified for corner staple, separation cut, perforation cut or fold, this OPCOUNT value is ignored. The printer determines the positions of the operations. The default is 0 (zero).

**OPPOS n**

Use OPPOS to specify the offset of finishing operation along the finishing operation axis that is measured from the point where the finishing operation axis intersects the bottom edge or left edge of the medium toward the center of the medium. Each consecutive OPPOS parameter is used to position a single finishing operation that is centered on the specified point on the finishing operation.

For AFP, the subparameter is an integer value in the range of 0 - 32,767, specified in millimeters.

For UP3i, the subparameter is an integer value in the range of 0 - 999999999, specified in millipoints (1/72000 inch).

Do not specify the unit of measure. Do not specify OPCOUNT when you use OPPOS. If OPPOS is specified for corner staple, fold, separation cut, or perforation cut, the OPCOUNT value is ignored.

---

4. This REFERENCE parameter can be used only for edge type operations (for example, SADDLE, EDGE, FOLD, CFOLDIN, PUNCH, SADDLEIN, CUT, PERFORATE).

5. This REFERENCE parameter can be used only for edge type operations.
**OPOFFSET n**

Specifies the offset of finishing operation axis from the reference edge, which is measured from the reference edge toward the center of the medium.

For AFP, the subparameter is an integer value in the range of 0 - 32,767, specified in millimeters.

For UP3i, the subparameter is an integer value in the range of 0 - 999999999, specified in millipoints (1/72000 inch).

Do not specify OPOFFSET for corner staple or saddle stitch; the corner staple or saddle stitch values are ignored when specified with OPOFFSET.

The following examples show how to specify finishing operations.

- To request scope as the entire print job with one corner staple in the top-left corner, specify:
  ```
  FINISH SCOPE PRINTFILE OPERATION CORNER REFERENCE TOPLEFT;
  ```

- Sometimes a user wants to request multiple finishing operations. To request that the fifth document in the job stream is finished with top-left corner staple and the ninth document is edge stitched only at the print default location, specify:
  ```
  FINISH SCOPE 5
  OPERATION CORNER
  REFERENCE TOPLEFT
  SCOPE 9
  OPERATION EDGE;
  ```

- The following example requests that SCOPE 5 (the fifth document in the job stream):
  1. Use the UP3i interface, is punched at the device default reference edge, and is offset with the device default number and type of holes.
  2. Use the normal AFP interface to staple the top-left corner.

The following example also requests that SCOPE 9 (the ninth document in the job stream):
  1. Use the UP3i interface to trim on the front.
  2. Use the normal AFP interface to edge stitch at the printer default location and offset with the device default number and type of staples.

  FORMDEF FinSm2 Replace Yes
  FINISH
  SCOPE 5 OPERATION UP3i XType X'0A'
  OPERATION CORNER REFERENCE TOPLEFT
  SCOPE 9 OPERATION UP3i XType 6 XOper 1
  OPERATION AFP EDGE;

**Finishing Operation Nesting Rules:**

When more than one finishing operation that involves a collection of media is specified for some portion of the print file, a nesting of the operations is defined first by the scope of the operation and second by the order of the operation in the data stream.

Finishing operations with a broader scope are nested outside of finishing operations with a narrower scope. The following scopes are listed in descending order:

1. Print-file level finishing (SCOPE PRINTFILE)
2. Document-level finishing, each document in the print file (SCOPE ALL)
3. Document-level finishing, a selected document in the PRINTFILE (SCOPE n)
4. Medium-map-level finishing, a collection of sheets (SCOPE BEGCOLL)

Finishing Operation Implementation Notes:
For some printers, the JOG function cannot be combined with a finishing operation. In this case, the JOG function is ignored. Check your printer documentation.

**ADJUST** \( n \)

Establishes the range of horizontal adjustment for the printed area on the sheet. The default is 0. The adjustment range can be set 0 - 20 L-units. After a value is set, it is the maximum amount available in both directions, plus and minus.

**Notes:**
1. If you specify ADJUST, the maximum logical page size (in the horizontal direction) is reduced by the amount you specified here.
2. The ADJUST \( n \) subcommand is used only on the IBM 3800 printers.

**INVOKE**

Specifies where the next page of data is placed when this copy group is activated by conditional processing or by an Invoke Medium Map structured field.

INVOKE SHEET, which is the default, places the next page of data on a new sheet. The NEXT, FRONT, and BACK parameters place the next page in a subsequent partition on the same sheet or, if no partitions are available, on the next sheet. If FRONT or BACK is specified, INVOKE selects only partitions on the front or back.

Print servers accept the NEXT, FRONT, and BACK values of the INVOKE subcommand only if the new copy group has the same medium modifications as the previous copy group. Some examples of medium modifications are duplexing, input bin, output bin, page offset, N_UP values, presentation, direction, medium (not page) overlays, text suppression, processing functions, print quality, finishing, jogging, and constant forms control. See the Media Eject Control Triplet (X'45') section in the Mixed Object Document Content Architecture Reference for a full description of the factors that allow a conditional eject to the next partition instead of the next sheet.

If any of these modifications differ, the print server ejects to a new sheet when the copy group is invoked. If you want to change overlays when you are
When you use PLACE subcommands, the NEXT, FRONT, and BACK parameters place the next page by using the next sequential PLACE subcommand that matches the requirement (next, front, or back). For example, if you print by using the second PLACE subcommand of copy group A, and then you change to copy group B, you start with the third PLACE subcommand of copy group B.

A CONSTANT parameter on the PLACE subcommand does not alter the selection process. The selection is complete, even though the selected PLACE subcommand does not place the data. N_UP does the constant modification and continues until it finds a PLACE subcommand that does not specify CONSTANT. The data is placed with this subcommand. Observe that this PLACE subcommand need not match the FRONT or BACK specifications of the INVOKE subcommand.

**SHEET**
Specifies that data is placed in the first selected partition of the sheet.

**NEXT**
Specifies that data is placed in the next selected partition.

**FRONT**
Specifies that data is placed in the next selected front partition.

**BACK**
Specifies that data is placed in the next selected back partition.

**JOG**

Yes

No

Specifies whether a JOG subcommand is sent to the printer when this FORMDEF is selected by an IMM structured field, or through conditional processing. When the JOG subcommand is sent, a printer either offsets (jogs) or prints copymarks. For cut-sheet printers, or for continuous-form printers with burster-trimmer-stacker enabled, the JOG subcommand causes the first sheet that is controlled by this FORMDEF to be stacked offset from the previous sheets. For continuous forms printers without a burster-trimmer-stacker, the JOG subcommand causes an increment in the copymark that is printed on the carrier strip. JOG subcommands also are sent to the printer at the beginning of each data set or at the beginning of each job, depending on host parameters. For more information about copymarks, see the system programming guide for your host print server.

**YES**
Specifies that a JOG subcommand be sent to the printer. The first sheet that is printed is offset or the copymark is incremented.

**NO**
Specifies that no JOG subcommand be sent to the printer. The first sheet that is printed is not offset; the copymark is not incremented.

**QUALITY n**
QUALITY \( n \)

Specifies the print quality. This subcommand is recognized only on printers that can produce more than one level of print quality. The default is determined by the printer model. (On some printers, the default can be set at the printer itself.) For more information, see your printer publications.

\( n \) You can select a level of print quality by entering any whole number 1 - 10. Higher numbers correspond to higher levels of print quality; lower numbers correspond to lower levels. For more information, see your printer publications.

Print quality is determined by a numerical code in the range of 1 to 254 (hexadecimal X'01' - X'FE'). The codes corresponding to the possible QUALITY parameters are:

1 = 15 (X'0F')
2 = 40 (X'28')
3 = 65 (X'41')
4 = 90 (X'5A')
5 = 115 (X'73')
6 = 140 (X'8C')
7 = 165 (X'A5')
8 = 190 (X'BE')
9 = 215 (X'D7')
10 = 240 (X'F0')

CMRTAGFIDELITY

Specify the exception continuation and reporting rules for Color management resource (CMR) tag exceptions.

Note: See Chapter 8, “AFP Color Management,” on page 163 for more information about using the CMRTAGFIDELITY subcommand.

STOP
CMR Tag exception rule is “Stop presentation at point of first CMR tag exception and report the exception”.

CONTINUE
CMR Tag exception rule is “Do not stop presentation because of CMR tag exceptions and do one of the following:”

NOREPORT
Do not report the CMR tag exception to the print server. This is the default if NOREPORT or REPORT is not coded.

REPORT
Report the CMR tag exception.

PROCESSING
Specifies more post-processing capabilities for selected printers and attached equipment. This option can be used only on a single page or a set of pages. The subcommand expects one to three of the following keywords:

**MEDIA_INFO n**
This parameter specifies the ID of fixed medium information that a printer or printer-attached device applies to a page. Examples such as color plates logos, letter heads, and other fixed images.

The numeric values that can be included are:

**0 - 254**
These numeric values select a particular fixed medium local ID that the printer or printer-attached device applies to a sheet. One or more IDs can be specified within this range.

**255**
This value selects all the current fixed medium local IDs that the printer or printer-attached devices applies to a sheet.

**PERFORATE**
Specifies a perforation cut at one or more fixed locations on the sheet according to the printer or printer-attached device.

**CUT**
Specifies a separation cut at one or more fixed locations on the sheet according to the printer or printer-attached device.

**COMMENT qstring**

Specifies a string comment. Use COMMENT to mark a form definition with a user comment. The string is placed in the NOP structured field of the form definition.

**qstring**
Specifies a quoted set of strings up to a total of 255 characters.

**Note:** In PPFA, a keyword or parameter (token) cannot extend across a line. Therefore, you must break the string into several strings to have a comment string that is longer than what fits on one line. Each string must be a complete token with beginning and ending quotation marks. Be sure to add a blank before the ending quotation mark if you want the comment lines separated. For example:
FORMDEF replace yes
COMMENT 'first line of comment '
       'second line of comment';

PPF composes the comment to be:
first line of comment second line of comment

and places it in a separate NOP structured field in the form definition.

COMPATPGP1

Specifies that a Page Position structured field of type Format-1 (PGP-1) is
generated when a PGP of type Format-2 (PGP-2) is not required. A PGP-1 is
generated when all of the following conditions exist:
• The keyword COMPATPGP1 is coded on the form definition
• The form definition is simplex
• The form definition is not enhanced NUP (with the PLACE subcommand)
• The form definition is not simple NUP (with the PARTITION subcommand.)

Note: If it does not matter which internal structures PPFA uses, you do not
need to use this function.

PELSPERINCH n

Specifies the Logical Units in pels per inch for this form definition. Use the
PELSPERINCH parameter to tell PPFA the pel resolution of your printer to
generate more exact object placements.

n     Specifies an integer number 1 - 3,276, which determines the Logical Units
      in pels per inch.

Note: If the L-Units are not specified on this form definition, they are
defaulted to 240 pels per inch.

In Figure 126 the form definition xmp01 specified L-Units as 300 pels per inch.
Because the COPYGROUP C1 does not specify L-Units, it inherits 300 pels per
inch. COPYGROUP C2 does specify L-Units as 1200 pels per inch.

FORMDEF xmp01 replace yes
PELSPERINCH 300 ;

COPYGROUP C1
  offset 2 in 3 in;

COPYGROUP C2
  offset 2 in 3 in
  PELSPERINCH 1200;

Figure 126. PELSPERINCH example
The code in COPYGROUP C1 (offset 2 in 3 in) produces internal and structured field values for \( x \) and \( y \) of 600 and 900, whereas in COPYGROUP C2 the same code produces values of 2400 and 3600 because of the difference in L-Units.

**BINERROR**

![BINERROR Diagram]

Tells the printer whether you want to stop printing if the wrong media is loaded on the printer or the bin number is not found.

This subcommand is displayed only on the FORMDEF command, not the COPYGROUP or the SUBGROUP commands, since the scope of the subcommand is throughout the FORMDEF. Printing control is based on the status of the media that is loaded as it pertains to the BIN subcommand in effect at the time.

**STOP**

If the specified input bin is in error, stop the print job and hold it in a state from which it can be resubmitted.

**CONTINUE**

If the specified input bin is in error, continue printing by using the printer default input bin.

**COLORVALUERR**

![COLORVALUERR Diagram]

When the form definition contains color values that the printer cannot render exactly as specified, you can request that the printer substitute colors and continue job processing, or you can request the printer to stop. If you request STOP, the printer issues an error and stops processing. If you request CONTINUE, you can ask for an error report.

**STOP**

Specifies that an error is issued by the printer and the job is ended if the printer reports a color exception. A color exception is reported if the color specification in the data stream cannot be rendered as specified. Also, a color exception is reported if the host print server supports color fidelity and the target printer does not.

**CONTINUE**

Specifies that an exception condition is ignored. Also, the printer substitutes colors for any that it cannot render, and the job continues.

**REPORT**

Specifies that the error is reported by the printer.

**NOREPORT**

Specifies that the error is not to be reported by the printer. NOREPORT is the default if COLORVALUERR CONTINUE is coded and REPORT or NOREPORT is not coded.
**Note:** When the printer reports a color value exception, the following actions are taken:

- If the print server and the printer both support Color Fidelity and the `COLORVALUERR` subcommand is coded, printing occurs as previously described.
- If the print server and the printer both support Color Fidelity and the `COLORVALUERR` subcommand is not coded, the print server instructs the printer to reset to defaults at the beginning of the job.
- Whenever the print server supports Color Fidelity but the printer does not, the following rules apply:
  1. If no `COLORVALUERR` subcommand is issued, printing continues. However, color exception errors are reported and ignored.
  2. If the `COLORVALUERR` subcommand is issued, you might receive print server errors or the command might be ignored, depending on the level of PSF you installed and your operating system (for example, z/OS, VM, or VSE). Therefore, you must not use the `COLORVALUERR` subcommand if you do not have a host print server that supports it.

- Whenever the printer supports Color Fidelity but the print server does not, the following rules apply:
  1. If no `COLORVALUERR` subcommand is issued, printing continues. However, color exception errors are reported and ignored.
  2. If either `COLORVALUERR STOP` or `COLORVALUERR CONTINUE NOREPORT` is coded, the print server issues an error and stops printing, even if a color exception error does not exist.
  3. If `COLORVALUERR CONTINUE REPORT` is coded, the print server continues printing. However, color exception errors are reported and ignored.

**FINERROR**

If both the host PSF and target printer support finishing fidelity, you can control job continuation and error reporting with the `FINERROR` subcommand on the FORMDEF command. If a form definition requests a finishing operation that is not available with the printer, you can request that the job continue processing or cause it to stop printing.

`FINERROR` covers only operations that the printer cannot process. For example, a stapling operation is specified on a device that is not equipped with a stapler. It does not cover temporary exceptions that require operator intervention, such as an empty stapler.

**STOP**

Use `STOP` to specify that the job is ended when a finishing exception is detected by the printer. A finishing exception that stops presentation is reported and the print file is put on hold to be resubmitted when the finishing operation can be done.
CONTINUE
Use CONTINUE to specify that the exception condition is ignored and the job continues without applying the unavailable finishing operation.

NOREPORT
Use NOREPORT to specify that the error is not reported by the printer. NOREPORT is the default if FINERROR CONTINUE is specified without specify REPORT or NOREPORT.

REPORT
Use REPORT to specify that the error is reported to the printer.

Notes:
1. If finishing fidelity is requested with the FINERROR subcommand and it is supported by the printer and the print server, the job is processed as specified with the STOP, CONTINUE, REPORT, and NOREPORT parameters.
2. If finishing fidelity is requested and supported by the print server, but is not supported by the printer, the request is processed by the print server as follows:
   • If you specify FINERROR STOP, the print server issues an error message and stops processing.
   • If you specify FINERROR CONTINUE, the print server prints the job and either issues a message if REPORT is specified or does not issue a message if NOREPORT is specified.
3. If finishing fidelity is not requested with the FINERROR subcommand or the print server does not support finishing fidelity, the job is printed and the finishing operations that cannot be done are not applied. Finishing exceptions are reported.

Examples:
FORMDEF xmp01 FINERROR STOP
   REPLACE YES;
   Copygroup X
   ... ;
FORMDEF xmp02 FINERROR Continue NoReport;
   Copygroup &
   ... ;

If both the print server and the printer support finishing fidelity:
• In the first example, FORMDEF xmp01 specifies a STOP parameter if a finishing error is encountered. If the specified finishing operation is not available, the printer reports an error, does not print the job, and places the job on hold to be resubmitted when the finishing operation can be done.
• In the second example, FORMDEF xmp02 specifies a CONTINUE parameter if a finishing error is encountered and a REPORT is made. If a specified finishing operation is not available, the printer continues processing the print job without applying the unavailable finishing operation or reporting the error.

FONTFID

<table>
<thead>
<tr>
<th>FONTFID—NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FONTFID—YES</td>
</tr>
</tbody>
</table>
Indicates to the print server whether the form definition accepts the fidelity of the specified fonts when a raster font of a specified resolution and metric-technology cannot be found on the printer. To get the print server to accept this command you also must specify font resolution on either the FONT command or externally (for example, on the JCL). Not coding FONTFID is equivalent to coding FONTFID NO.

**YES**

Specifies that no substitution is allowed and the print server issues an error message if it cannot find the font that matches the specified resolution and metric.

**NO**

Specifies that the print server does not enforce font fidelity. The print server does not check for a match of the specified resolution and metric with the font found on the system.

**Notes:**

1. The FONTFID subcommand is used in concert with the RESOLUTION and METRICTECHNOLOGY subcommands on the FONT command, which are used to rigorously specify the font characteristics.

2. This subcommand assists the user who created a form definition and page definition for printing with a raster font on a printer of one resolution (for example, a 240 pel printer), and moved that application to a printer of another resolution (for example, a 300 pel printer). When the print server cannot match the raster font, it substitutes an outline font, which often causes the placed text to overflow or underflow the intended space on the page. If this happens, the user can specify the actual metric and resolution of the font that is used to print the text, and also specify FONTFID YES so that the print server does not substitute another font.

**TONERSAVER**

Specifies whether the printer's toner saver mode is activated. When activated, it might degrade print quality, and might also impact performance. If DEVSETTING is specified, the specific device's setting is used. If TONERSAVER ON or OFF is specified, it overrides any QUALITY parameters. This function is device-specific. Make sure that your printers support this feature by checking the printer's documentation.

**N_UP**

The N_UP subcommand is used to specify the number of copies of a page to be printed. It is used with the OVERLAY and PLACE subcommands to control the placement of overlays and text on the page.
{ 1 | 2 | 3 | 4 }

Specifies the number (1, 2, 3, or 4) of equal-size partitions into which the sheet is divided. See the list of printers that support the N_UP subcommand.

If you do not specify the N_UP subcommand in the COPYGROUP command, the N_UP subcommand from the FORMDEF command is the default for the COPYGROUP command. You can mix N_UP printing and non-N_UP printing by specifying or not specifying the N_UP subcommand in each copy group and by not specifying N_UP in the FORMDEF command.

OVERLAY Subcommand

Specifies the name of an overlay to be placed with every page in each of the N_UP partitions. The overlay is placed relative to the page origin, or if the PARTITION parameter is specified, relative to the partition origin. You can specify a maximum of 254 OVERLAY subcommands in a copy group.

rel-x rel-y

Specifies the horizontal and vertical adjustment to the position of the overlay. This adjustment is in addition to any offset values built into the overlay. The x and y values can be positive (+) and negative (-). You can specify them in inches (IN), millimeters (MM), centimeters (CM), points, or pels. If you do not specify a unit value, PPFA uses the unit value that is specified in the last SETUNITS command or uses a default unit value of inches.

Note: This OVERLAY subcommand cannot be specified if the PLACE subcommand is specified. Use the OVERLAY parameter of the PLACE subcommand instead.

PARTITION

Specifies that the overlay is to be placed relative to the partition origin.

OVROTATE { 0 | 90 | 180 | 270 }

Specifies the rotation of the placed overlay in regard to the x-axis of the page.

Example: Assuming the overlay has (0,0) placement coordinates, this causes page overlay x2 to be placed 1.5 inches to the right and 2.7 inches below the beginning of the page and rotated 90 degrees clockwise in regard to the page.

Formdef f1

PLACE 1 FRONT
OVERLAY x2 1.5 in 2.7 in
OVROTATE 90;

PLACE Subcommand
Places a page of data or a constant modification relative to a partition. Each PLACE subcommand specifies the number \( n \) of a partition on either the front or back side of the sheet. FRONT is the default, if you do not specify this subcommand. You must specify the same number of PLACE subcommands as the number of partitions on the sheet. The sequence of the PLACE subcommands is the sequence in which incoming pages are placed in the partitions.

**Notes:**

1. The use of the PLACE subcommand indicates enhanced N_UP printing.
2. The PLACE subcommand is valid only on printers that support enhanced N_UP printing. If PLACE is not specified, pages are placed in partitions in the default partition sequence.

\( n \) Specifies the numbered partition (1 - 4) into which the page of data is placed.

**FRONT**

Specifies that this partition is placed on the front side of the sheet.

**BACK**

Specifies that this partition is placed on the back side of the sheet.

**CONSTANT**

Specifies that no page data is placed by this PLACE subcommand.

Use CONSTANT when you are placing overlays without user's data or are placing fewer data pages on the sheet than the number of partitions that are specified in the N_UP subcommand.

For an example of using the CONSTANT parameter with overlays and to understand how the ordering of the PLACE subcommand affects overlays, see ["Example 3: Enhanced N_UP Asymmetric Pages" on page 158.](#)

**OFFSET** \( rel-x \) \( rel-y \)

Specifies a relative offset of the page horizontally \( (x) \) and vertically \( (y) \) from the partition origin. If OFFSET is not coded, PPF uses the value of 0.1 inches for both the \( x \) and \( y \) offsets. This OFFSET parameter overrides any other OFFSET parameters that are specified on the FORMDEF or COPYGROUP command. You can specify the units in inches (in), millimeters (mm), centimeters (cm), points, or pels. If you do not specify a unit value, PPF uses the unit value that is specified in the last SETUNITS command or uses a default unit value of inches.

**Note:** You can specify this offset as negative to crop the top, left, or both of an image.

**OVERLAY Subcommand**

See [OVERLAY Subcommand](#) for the syntax.

\( name \)

Specifies the name of an overlay to be placed with this PLACE subcommand. The overlay is placed relative to the page origin or,
if the PARTITION keyword is specified, to the partition origin. You can specify multiple OVERLAY parameters in each PLACE subcommand.

\texttt{rel-x rel-y}

Specifies the horizontal and vertical adjustment to the position of the overlay. This adjustment is in addition to any offset values built into the overlay. The \textit{x} and \textit{y} values can be positive (+) and negative (-). You can specify them in inches (IN), millimeters (MM), centimeters (CM), points, or pels. If you do not specify a unit value, PPFA uses the unit value that is specified in the last SETUNITS command or uses a default value of inches.

\textbf{PARTITION}

Specifies that the previous offset is from the partition origin. If not present, the offset is from the page origin, which is subject to the OFFSET parameter.

\textbf{OVROTATE} \{ \textbf{0} | \textbf{90} | \textbf{180} | \textbf{270} \}

Specifies the rotation of the placed overlay in regard to the \textit{x}-axis of the page.

\textbf{ROTATION} \{ \textbf{0} | \textbf{90} | \textbf{180} | \textbf{270} \}

Specifies the clockwise rotation of the page and associated page overlays placed by this PLACE command.

Rotation turns the page and its associated page overlays around their fixed origin points. If you rotate the page without moving its origin point, you might rotate it off the physical medium. To prevent this rotation, always offset the page origin to the place you want it to be for the rotated page, as shown in \textbf{Figure 127}.

```
Figure 127. Offsetting the Page Origin for Rotated Pages
```

\textbf{VIEW}

Determines whether this N_UP PLACE page is viewable. VIEW is relevant only when the page is being presented on a display. VIEW is ignored if the page is being printed. If VIEW is not coded, it is equivalent to specifying VIEW YES.

\textbf{YES}

Specifies that this N_UP page is viewable and is presented.
NO  Specifies that this N_UP page is not to be presented.

VFYSETUP  verificationID ...

Use to propagate the setup IDs to all medium maps (copy groups) in the form definition. Do not specify VFYSETUP on the COPYGROUP command. Before you use the VFYSETUP subcommand, verify that your version of print server supports FORMDEF setup verification.

To use VFYSETUP, specify one or more 2-character (4-digit hexadecimal) identifier sets that match the Setup Verification IDs defined at the printer operator’s console for the specific print job. For example, if the Setup Verification IDs defined at the printer were X’012F’, X’0521’, and X’938A’, specify these:

FORMDEF vfy7 REPLACE YES VFYSETUP 012F 0521 938A;

When the print server processes the print job, it compares the setup verification IDs in the form definition to the IDs that are active in the printer. If the active IDs in the printer do not match the IDs required by the form definition, or if the printer does not support FORMDEF setup verification IDs, the job is held.

VFYSETUDP  decimal/verificationID ...

Use to propagate the setup IDs to all medium maps (copy groups) in the form definition. Do not specify VFYSETUDP on the COPYGROUP command. Before you use the VFYSETUDP subcommand, verify that your version of print server supports FORMDEF setup verification.

To use VFYSETUDP, specify one or more decimal numbers that match the Setup Verification IDs defined at the printer operator’s console for the specific print job. For example, if the Setup Verification IDs defined at the printer were 303, 1313, and 37770, specify these:

FORMDEF vfy7 REPLACE YES VFYSETUDP 303 1313 37770;

When the print server processes the print job, it compares the setup verification IDs in the form definition to the IDs that are active in the printer. If the active IDs in the printer do not match the IDs required by the form definition, or if the printer does not support FORMDEF setup verification IDs, the job is held.
You use the Text Fidelity subcommand to specify what happens when the printer reports a text exception. A text exception is reported if the printer encounters a text control sequence it doesn’t recognize.

**Note:** If the printer and print server both support Text Fidelity, the following occurs.

**STOP**
When a text exception occurs, this parameter specifies that the job is ended and the text exception is reported.

**Note:** When presentation is ended, the print file is put into a state where it can be resubmitted when the text can be rendered without exceptions.

**CONTINUE**
When a text exception occurs, this parameter instructs the printer to skip the text control sequence it did not recognize and continue processing the print job.

**REPORT**
Specifies that the error is reported by the printer.

**NOREPORT**
Specifies that the error is not reported by the printer. This is the default if TEXERROR CONTINUE is coded.

**Form-size Parameters**

```
XMSIZE <x> units

YMSIZE <y> units
```

Specifies the medium presentation space. This is also known as the medium size or form length and form width.

**Notes:**
1. This function requires both Printer Server and Printer support.
2. The printer does not adjust your media presentation space size to be larger than the paper size (or what the printer thinks is the paper size).
3. Some printers (such as the InfoPrint 1145 and the InfoPrint 4100) do not support the IPDS Set Media Size (SMS) command. The form size cannot be set with the form definition. Do not use the XMSIZE and YMSIZE subcommands for those printers that do not support the SMS commands.
4. Other printers (such as the 6400, 4247, and 4230) do not support the Set Media Origin (SMO) command. The media origin does not change. For the 6500, 6400, 4247, and 4230 printers form length is always YMSIZE and form width is always XMSIZE.
5. For all other printers, use the settings that are shown in Table 10 on page 233. For these other printers, whether the XMSIZE or YMSIZE is form length or form width depends on the medium presentation space orientation, type of form, and NUP setting. The following examples are from Table 10 on page 233. See the table for other media combinations.
   - Wide fanfold paper, PRESENT=Landscape, DIRECTION=ACROSS, and no-NUP - The form length is YMSIZE.
   - Narrow fanfold paper, PRESENT=Landscape, DIRECTION=ACROSS, and no-NUP - The form length is XMSIZE.
• Cutsheet paper, PRESENT=Landscape, DIRECTION=ACROSS, and no-NUP - The form length is XSIZE.

6. Only two choices exist. If you try one that doesn’t work, try the other. For example, if you try XMSIZE for the form length and it doesn’t create a longer form, use YMSIZE.

**XSIZE**
Specifies the medium presentation space along the X-axis (also known as the medium’s size in the X-direction). If this subcommand is specified on the FORMDEF command, it becomes the default for all copy groups that do not specify XSIZE on the COPYGROUP command. If this subcommand is not specified on the FORMDEF command, the printer’s current default X-axis becomes the default for all copy groups that do not specify XSIZE on the COPYGROUP command.

\[ x \]
Enter a number with 0 - 3 decimal places and optional units.

**YSIZE**
Specifies the medium presentation space along the Y-axis (also known as the medium’s size in the Y-direction). If this subcommand is specified on the FORMDEF command, it becomes the default for all copy groups that do not specify YSIZE on the COPYGROUP command. If this subcommand is not specified on the FORMDEF command, the printer’s current default Y-axis becomes the default for all copy groups that do not specify YSIZE on the COPYGROUP command.

\[ y \]
Enter a number with 0 - 3 decimal places and optional units.

**units**
Enter IN for inches, CM for centimeters, MM for millimeters, or PELS for pels. If units is not specified, the default is the most recent setting of the SETUNITS command, or inches if no SETUNITS command is coded.

**Examples:**

FORMDEF FMSZX1 Replace Yes
PRESENT Landscape Direction Across
XSIZE 8.5 in YMSIZE 11.0 in;
COPYGROUP cp1;
COPYGROUP cp2;

FORMDEF FMSZX2 Replace Yes YMSIZE 17.0 in;
COPYGROUP cp3;
COPYGROUP cp4;

• In the first example, the printer is a 4400 thermal printer, which supports both SMS and SMO IPDS commands. The form definition that is named FMSZX1 defines a form length of 8.5 inches and form width of 11.0 inches. COPYGROUP cp1 and cp2 inherit those sizes from the form definition.

• In the second example, the printer is a 6400 printer and you want to define the form length. The form definition that is named FMSZX2 defines form length as 17 inches and leaves the form width as the printer default. COPYGROUP cp3 and cp4 inherit those sizes from the form definition.

• If this example is run on z/OS with FORMLEN defined in the JCL, the JCL definition is used.
OVERLAY Command

This OVERLAY command identifies an electronic medium overlay to be used in one or more subgroups of a copy group. See "Medium Overlays and Page Overlays" on page 160 for more information. When you are using the OVERLAY command, follow these guidelines:

- An OVERLAY command comes after the COPYGROUP command.
- A separate OVERLAY command must be specified for each electronic overlay that is used in a subgroup.
- A maximum of 254 OVERLAY commands can be specified for coded overlays per copy group.
- The overlay that is named here must be referenced in a SUBGROUP command to be printed (see SUBGROUP OVERLAY Subcommand).

Notes:
1. Overlays contain their own positioning controls.
2. This does not define page overlays that are placed with the N_UP subcommand. See "Medium Overlays and Page Overlays" on page 160 for more information.

OVERLAY [ name1 ] name2

Identifies an electronic overlay to be used in one or more subgroups of a copy group.

name1

Specifies an alphanumeric name of 1 - 16 characters (local name) for the overlay. It must conform to the token rules and must be unique within a copy group.

Note: If name1 is omitted, name2 is used as the local name and is the name that is used in the subgroup command.

name2

Specifies an alphanumeric name of 1 - 6 characters (user-access name) for this overlay. A prefix of 01 is added by PPFA to identify the overlay resource.

Subcommands

RASTER or NORASTER

Specifies overlays as raster or not raster data.

RASTER

Specifies that this overlay is to be kept in the printer as raster data. If this overlay is to be used several times, the printer does not need to recompile it each time.

NORASTER

Specifies that this is a coded overlay. A maximum of 254 coded overlays can be specified per copy group.

Note: The RASTER or NORASTER subcommands are used only on the IBM 3800 printers.
SETUNITS Command

The SETUNITS command specifies the value and the unit of measurement that is the default for any subsequent measurement parameter in all of the commands and subcommands. These values remain the default values until another SETUNITS command is specified. The SETUNITS command must be specified as the first command in a form definition. If this command or a measurement parameter is not specified, the defaults that are identified within the following description are used.

SETUNITS
Specifies the value and the unit of measurement that is the default for any subsequent measurement parameter in all of the commands and subcommands.

x Specifies the number that is used for horizontal measurement. A number with up to three decimal places can be used. The default is 1. The unit choices are IN, MM, CM, POINTS, PELS, or CPI.

Note: This value affects subsequent OFFSET subcommands.

y Specifies the number that is used for vertical measurement. A number with up to three decimal places can be used. The default is 1. The unit choices are IN, MM, CM, POINTS, PELS, or LPI.

Note: This value affects subsequent OFFSET subcommands.

Using CPI and LPI Units of Measurement: The CPI and LPI units of measurement make it possible to write the following command:
SETUNITS 10 CPI 6 LPI ;

This command sets the units of measurement for horizontal and vertical spacing in terms of characters per inch and lines per inch. You can then use the OFFSET subcommand specifications to increment the spacing one character or one line at a time. The distance that is specified by \( n \) characters over and by \( n \) lines down is defined in the governing SETUNITS command. In this example, 10 characters per inch (CPI) and 6 lines per inch (LPI) exist.

Subcommand

LINESP \( n \)
This subcommand is to be used within a page definition to set up default line spacing; it serves no purpose when used within a form definition.
The SUBGROUP command specifies the number of copies of a single page that are to be printed and any modifications (consisting of overlays, suppressions, type of duplexing, and forms flash) to be made to the copies. A SUBGROUP command follows a COPYGROUP command; a maximum of 127 SUBGROUP commands can be specified within each copy group.

Notes:
1. The BOTH subcommand causes two subgroups to be generated. Thus, a maximum of 63 subgroups can be specified when the BOTH subcommand is used.
2. When you specify the DUPLEX subcommand (with a parameter other than NO) in the COPYGROUP command, you must include one SUBGROUP command for each side of a sheet, or you can specify the BOTH subcommand in a single SUBGROUP command.

Subcommands

COPIES \( n \)

Specifies how many copies of each page are to be printed.

\( n \) Defines the number of copies (the maximum number is 255). When BACK is specified within a SUBGROUP command, the system counts the front pages printed (the actual number of sheets) not copies made (front and back). The default is 1.

FLASH
Specifies whether to use forms flash.

**NO** specifies that forms flash does not occur.

**YES** specifies that forms flash occurs.

**Notes:**
1. When forms flash is used, its name must be specified in the job control language for the print job. The operator must place the correct negative into the 3800 when the job is ready to print.
2. The FLASH subcommand is used only on the IBM 3800 printers.

**Side subcommands**

- **BOTH**
- **BACK**
- **FRONT**

These optional subcommands specify whether the subgroup is for both sides of a sheet or for only the front or the back side.

**Rules:**
1. Subgroups must specify FRONT and BACK if an overlay, suppression, or forms flash appears on one side but not on the other.
2. The FRONT and BACK subgroups must have the same number of copies.
   - If the number of copies differs, the COPIES parameter of the BACK subgroup is ignored, and a warning message is issued.
3. The FRONT and BACK subcommands must occur in pairs.
4. If the FRONT and BACK subcommands are specified with DUPLEX NO (in the FORMDEF or COPYGROUP commands), PPFA issues an error message and does not create the form definition.

**BACK**

Specifies this SUBGROUP command is for the back sides of the sheets.

A subgroup with a BACK subcommand must have a FRONT subcommand in the preceding subgroup.

**FRONT**

Specifies this subgroup is for the front sides of the sheets.

If a DUPLEX subcommand in a FORMDEF or COPYGROUP command is specified with a parameter other than NO and the FRONT subcommand is specified in a SUBGROUP command, the next SUBGROUP command must have a BACK subcommand.

**BOTH**

Specifies this subgroup is used for both sides of the sheet.

This is the default when DUPLEX is specified in the copy group.

If BOTH is specified with DUPLEX NO (in a FORMDEF or COPYGROUP command), PPFA issues a warning message and ignores the BOTH subcommand.
Specifies the paper source. This subcommand is used only for printers that have more than one paper source.

**Note:** If you specify the BIN subcommand, you must also specify at least one of the legal parameters.

*n* An integer number 1 - 255 that is the Media Source ID (also known as the bin number).

1 Selects the primary paper source.

2 – 255
Selects another paper source. If the specified bin does not exist on your printer, the default paper source for that printer is used. For more information about paper sources on your printer, see your printer publications. Using a value of 100 is the same as specifying MANUAL.

**MANUAL**
Selects manual feed as a paper source on those printers that support manual feed. For more information, see your printer publications.

**ENVELOPE**
Selects an envelope paper source on those printers that support this function. For more information, see your printer documentation.

**Notes:**
1. BIN selection is overridden by the printer if the form defined to each bin is the same form number. Only the primary bin is selected.
2. The primary source usually contains either letter-size (US) or A4 (I.S.O.) paper. Other paper sources are used for less common paper sizes (such as legal-size) and for special paper (such as colored stock or pre-printed letterhead on heavy bond).
3. If duplexing is requested and you select from the front side from one bin and the back side from another bin, a warning message is issued and the printer takes the paper from the bin that is specified on the front side.

**MEDIANAME**
Selects a media source by specifying an agreed upon name for the bin. For a current list of the valid media names, see Appendix G, “PPFA Media Names,” on page 563.

qstring
Up to 12 characters within single quotation marks that specify the media source name. On some printers, this name is pre-set into the printer; on others, it also can be entered into the printer by the user. See your printer documentation for more information.

**COMPID** *m*
Selects a bin based on the component ID.
Note: For a current list of component IDs, see Appendix G, “PPFA Media Names,” on page 563. Component IDs 12,288 - 268,435,455 are reserved for the user.

**OUTBIN n**

Specifies the destination bin number for any pages that are directed by this form definition. Copy groups and subgroups in this form definition that do not specify an output bin number inherit this bin number.

**OVERLAY**

\[ \text{OVERLAY } \text{name} \]

Specifies the electronic overlay that is to be used with this subgroup.

\( \text{name} \)

Specifies either the local or user-access name. A maximum of eight names can be specified within a subgroup.

**Notes:**
1. If the local name is used, it must be defined in an OVERLAY command before it can be referenced.
2. PPFA does not check for duplicate user-access names.

**SUPPRESSION**

\[ \text{SUPPRESSION } \text{name} \]

Specifies that the named field is suppressed.

\( \text{name} \)

Specifies an alphanumeric name of 1 - 8 characters (local name) of the text field to be suppressed. A maximum of eight names can be specified within a subgroup.

The suppression field that is named here must be defined in a SUPPRESSION command that follows the FORMDEF command before it can be referenced. See “SUPPRESSION Command.”

**Note:** This subcommand is for text only fields.

### SUPPRESSION Command

\[ \text{SUPPRESSION } \text{name}; \]
A SUPPRESSION command, if used, must immediately follow the FORMDEF command. It names the suppression that is specified in the FIELD command of a page definition that is associating the form definition and the page definition.

**name**

Identifies an alphanumeric name of 1 to 8 characters (local name). The name must conform to the token rules.

You must specify the area to be suppressed in a FIELD command or a SUBGROUP command by using one of the names that are specified within this series of SUPPRESSION commands for the suppression to be effective.

**Notes:**
1. The SUPPRESSION command is for text only fields. It does not work for bar codes or other non-text fields.
2. A maximum of eight suppressions can be specified for one SUBGROUP command, and a maximum of 127 suppressions can be specified within one form definition.
Chapter 11. Page Definition Command Reference

This section includes:
- Sequence of commands for page definitions
- Page definition commands that are listed alphabetically
- Detailed information about each command
- Descriptions of the applicable subcommands and parameters for each command

Sequence of Traditional Commands for Page Definitions with PRINTLINE

```
[ SETUNITS ... ]
PAGEDEF
[ DEFINE CMRNAME ...]
[ FONT ...]
[ DOFONT ... ]
[ OBJECT ... ]
[DEFINE COLOR ... ]
[ PAGEFORMAT ]
[ TRCREF ...]
[ SEGMENT ...]
[ RENDER ...]
[ OVERLAY ...]
[ EXTREF ...]
[ CMR ...]
PRINTLINE [ FIELD | CONDITION ...]
[ ENDSUBPAGE ] |
[ PRINTLINE [ FIELD | CONDITION ...] ] ...
[ PAGEFORMAT ]
[ TRCREF ...]
[ SEGMENT ...]
[ OVERLAY ...]
PRINTLINE [ FIELD | CONDITION ...]
[ ENDSUBPAGE ] |
[ PRINTLINE [ FIELD | CONDITION ...] ] ...
```

Rules:
1. FONT and DOFONT commands must be specified immediately after a PAGEDEF command. The exception is the SETUNITS command.
2. OBJECT commands must be specified immediately after any FONT commands and before any PAGEFORMAT or other commands, except the SETUNITS command.
3. A SETUNITS command can be placed before any other PPFA command. The values set are in effect until the next SETUNITS command.
4. TRCREF, SEGMENT, and OVERLAY commands must be specified under their associated PAGEFORMAT command.
5. The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.
6. At least one PRINTLINE command is required per page format for Traditional Line Data Page definition. PRINTLINE and LAYOUT commands cannot be used within the same page definition.
7. An ENDSUBPAGE command can occur anywhere in a page definition that a PRINTLINE command can occur, except it cannot occur between a PRINTLINE command and its associated FIELD and CONDITION commands.

8. One file can contain multiple sets of page definitions.

Sequence of Record Formatting Commands for Page Definitions with LAYOUT

```
[ SETUNITS ...]
PAGEDEF
FONT
[ DOFONT ... ]
[OBJECT ... ]
[DEFINE COLOR... ]
[ PAGEDOFORMAT ]
[ SEGMENT ...]
[ OVERLAY ...]
[ RENDER ...]
[ EXTREF ...]
[ CMR ...]
[ LAYOUT ...]
[ CONDITION ...]
[ FIELD ...]
[ DRAWGRAPHIC ...]
[ ENDDOFORMAT ]
[ PAGEDOFORMAT ]
[ SEGMENT ...]
[ OVERLAY ...]
[ LAYOUT ...]
[ CONDITION ...]
[ FIELD ...]
[ DRAWGRAPHIC ...]
[ ENDDOFORMAT ]
```

Rules:
1. LAYOUT, XLAYOUT, and PRINTLINE commands cannot be mixed within the same PAGEDEF. At least one LAYOUT command is required per page format for a record formatting page definition.

2. FONT and DOFONT commands must be specified immediately after a PAGEDEF command.

3. A SETUNITS command can be placed before any other PPFA command. The values set are in effect until the next SETUNITS command.

4. SEGMENT, OVERLAY, and EXTREF commands must be specified under their associated PAGEFORMAT command.

5. The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.

6. One file can contain multiple sets of page definitions.

7. At least one FONT or DOFONT command is required for each PAGEDEF command.

Sequence of Commands for XML Page Definitions with XLAYOUT

```
[ SETUNITS ...]
PAGEDEF
FONT
[ DOFONT ... ]
[OBJECT ... ]
```
Rules:
1. LAYOUT, XLAYOUT, and PRINTLINE commands cannot be mixed within the same PAGEDEF. At least one XLAYOUT command is required per page format for an XML page definition. At least one FONT or DOFONT command is required for each PAGEDEF command.
2. FONT and DOFONT commands must be specified immediately after a PAGEDEF command.
3. A SETUNITS command can be placed before any other PPFA command. The values set are in effect until the next SETUNITS command.
4. SEGMENT, OVERLAY, and EXTREF commands must be specified under their associated PAGEFORMAT command.
5. The first PAGEFORMAT command can be omitted in a page definition, if the page definition contains only one page format. If the PAGEFORMAT command is omitted, the PAGEDEF command parameters are used to define the page format.
6. One file can contain multiple sets of page definitions.
7. DOFONT commands must be after the PAGEDEF command and before the first PAGEFORMAT command.
8. EXTREF must be after a PAGEFORMAT (the scope if for that page format) and before the first PRINTLINE, LAYOUT, or XLAYOUT.

Diagram Shorthand

These terms are used in the command definitions:

\textit{x-pos}

A vertical position that uses a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 204.

\textit{y-pos}

A horizontal position that uses a numeric number followed optionally by a unit. For the available units, see “Units of Measurement” on page 204.
**CONDITION Command**

**Long Form**

```
<table>
<thead>
<tr>
<th>CONDITION</th>
<th>condname</th>
<th>START (Traditional)</th>
<th>START (Record Format and XML)</th>
</tr>
</thead>
</table>

WHEN
- CHANGE
  - EQ 'text'
  - NE
  - GT
  - LT
  - LE

BEFORE
- SUBPAGE
- NEWFORM
- NEWSIDE
- CURRENT or =
- FIRST
- NULL or /
- NEXT
- COPYGROUP cgname
- PAGEFORMAT pfname

AFTER
- SUBPAGE
- NEWFORM
- NEWSIDE
- CURRENT or =
- FIRST
- NULL or /
- NEXT
- COPYGROUP cgname
- PAGEFORMAT pfname
```

**START (Traditional):**

```
START n LENGTH n
   SPACE_THEN_PRINT YES
   SPACE_THEN_PRINT NO
```

**START (Record Format and XML):**

```
START n LENGTH n
   FLDNUM n
       START - 1 LENGTH longest
       START n LENGTH n
```

**Short Form**

```
<table>
<thead>
<tr>
<th>CONDITION</th>
<th>condname</th>
<th>START n</th>
</tr>
</thead>
</table>
```

**CONDITION**

The CONDITION command examines data in an input record and specifies actions to be taken based on the result of the examination.

- The `condname` parameter must come before any subcommands.
- No WHEN subcommand can follow an OTHERWISE subcommand in the same CONDITION command.

`condname`

Names the condition. The name must contain 1 - 8 alphanumeric characters.
PPFA allows cross-referencing to the *condname*. The cross-reference is done by using the short form of the CONDITION command (second format in the syntax table). By specifying a previously defined *condname*, PPFA uses the specifications from that command. When the condition is reused, the point where you want the comparison to begin might be at a different point in the record. By specifying the optional START subcommand, you can change the starting point of the comparison but not the field length. If the START subcommand is not specified, the starting point is the same as defined in the original CONDITION command.

**Note:** When you are comparing text in fields that contain delimiters, the comparison text must not contain the delimiter. The delimiter is not part of the data.

**Long Form Subcommands**

**START** \( n \)

Specifies the starting byte of the comparison field within the data record where the comparison is to be done.

\( n \) Specifies the number of bytes from the first data byte in the record as the starting point of the comparison field. The first data byte position of an input record is 1.

**Note:** The carriage-control character and the table-reference character are not considered data.

**LENGTH** \( n \)

Specifies the length of the comparison field.

\( n \) Specifies the number of bytes in the data record to be compared, beginning with the position specified in START. Valid values are numbers 1 - 8000. The length of the constant text must be the same as defined in this parameter or the results are invalid.

Comparisons are done on a byte-by-byte basis. Because the comparison field and the constant text must have the same lengths, padding is not necessary.

**Note:** If any part of the comparison field that is specified by the combination of START and LENGTH is outside the boundaries of the data record, no conditional processing is done. No WHEN is started. If an OTHERWISE is present, it is not started either.

**FLDNUM (Record Format and XML only)**

---

Chapter II. Page Definition Command Reference 275
Field number to be used in comparison. This keyword is used only if the DELIMITER field was used in the LAYOUT command. Fields cannot be counted without specifying delimiters in the database. When fields are counted, the first field after the record ID is to be considered FLDNUM 1.

To identify a part of a field that is numbered, you can specify the starting position (from the delimiter) and the length of the field to be used in the WHEN condition (the default of the longest parameter is the length of the longest condition or when no specific condition is specified, it is from the starting position to the end of the field.)

**SPACE_THEN_PRINT (Traditional only)**

```
SPACE_THEN_PRINT—YES
SPACE_THEN_PRINT—NO
```

Specifies whether ANSI carriage controls for spacing are enabled for the first record on the new logical page that follows the execution of the CONDITION command. The abbreviation of this command is SPACE.

**YES**

Specifies that the ANSI carriage-control character in the first print record of the new page is enabled for spacing. The spacing action that is specified in the carriage control is done after the eject to the new page. For example, if the carriage-control byte in the first record of the new page is a blank (skip one line before printing), then the first record skips the first line of the new page and prints at the second printline position.

**NO**

Specifies the ANSI carriage-control character spacing action is suppressed for the first print record of the new page. If this record contains a carriage-control spacing value, such as blank, 0, or –, the spacing is ignored and the record prints at the first printline position on the new page. Channel code values are not ignored. If the first print record contains a valid channel code value of 1 - 9, or A - C, then the first record on the new page prints at the printline defined with that channel code.

**Note:** This subcommand is effective for print files that contain ANSI carriage controls. It is not used for data files that contain machine carriage controls, or a mixture of ANSI and machine carriage controls.

**WHEN**

```
WHEN
    CHANGE
    BEFORE
    SUBPAGE
    NEWFORM
    NEWSIDE
    CURRENT or =
    FIRST
    NEXT
    COPYGROUP cgname
    PAGEFORMAT pfname
```

Marks the start of the conditional comparison parameters. At least one WHEN subcommand is required.

**comparison type**

Specifies the type of comparison that is to be done between the data in the comparison field (the portion of the record that is specified by START and LENGTH) and the constant text that is defined in the text parameter.
The values are:

EQ  Equal to
NE  Not equal to
GT  Greater than
GE  Greater than or equal to
LT  Less than
LE  Less than or equal to

text
Specifies constant text for comparison with the comparison field text. The constant text length must be the same as the value on the LENGTH subcommand, with a maximum length of 8000 bytes. Examples of valid text are:

2C(3) 'AB'
K '321,400'
x '41FE7799' 2 'CHARS'

Any values or parameters that are valid for the TEXT subcommand within the FIELD command can be used as text.

CHANGE
Specifies that the contents of the comparison field in this record are to be compared with the field in the record last processed by the same CONDITION command.

This parameter is an alternative to the comparisontype and text parameter combination but can be specified only once in a CONDITION command.

The result of the comparison is either TRUE or FALSE.

TRUE
When the contents of the comparison field change from one record to the next.

FALSE
When the print server processes the data, if the comparison field lies outside the boundary of the current record, which might occur with variable-length records or with truncated trailing blanks, the current record is not used in future comparisons.

CHANGE is always false if used with the first WHEN subcommand of a series (no previous record to compare against). Whenever a new data map (one with a different name) is invoked, all the CHANGE comparisons are reset. Field values in the previous data map are not retained.

BEFORE
Specifies that the conditional action takes place before the current line or subpage is processed. This is the default.

AFTER
Specifies that the conditional action takes place after the current line or subpage is processed.

LINE
Specifies that the conditional action takes place either before or after the current line.
**SUBPAGE (Traditional only)**

Specifies that the conditional action takes place either before or after the current subpage. Between LINE and SUBPAGE, SUBPAGE is the default.

For a description of subpages, see "Logical Page" on page 8.

**PAGE (Record Format and XML only)**

Specifies that the conditional action takes place either before or after the current page. This is the default. Between LINE and PAGE, PAGE is the default.

**Note:** For CONDITION commands in a Record Format or XML page definition, the keyword SUBPAGE is acceptable but obsolete. Record Format and XML page definitions do not have subpages.

**NEWFORM**

Specifies that the only action to be taken is skipping to the front of a new form (sheet) and restarting the page format.

**Note:** This parameter is an alternative to using the COPYGROUP and PAGEFORMAT parameters, and is equivalent to specifying CURRENT for the COPYGROUP parameter and NULL for the PAGEFORMAT parameter. CURRENT NULL are the respective defaults for COPYGROUP and PAGEFORMAT parameters; therefore, NEWFORM is the default action.

**NEWSIDE**

Specifies that the only action to be taken is skipping to a new side (either the back of the current sheet or the front of a new sheet) and restarting the page format.

**Notes:**

1. This parameter is an alternative to using the COPYGROUP and PAGEFORMAT parameters, and is equivalent to specifying NULL for the COPYGROUP parameter and CURRENT for the PAGEFORMAT parameter.
2. Conditional processing does not result in unnecessary blank pages.

If the line currently being processed is the first line on a side, then:

- A COPYGROUP or NEWFORM action that takes effect BEFORE LINE does not force an additional new form.
- A PAGEFORMAT or NEWSIDE action that takes effect BEFORE LINE does not force an additional new side.

Similarly, additional sides or forms are not forced by BEFORE SUBPAGE if the line currently being processed is in the first subpage on a side or a form.

**copygroup options**

Specifies a copy group to be invoked if the condition is true.

**Note:** Any copy group action (except NULL) restarts the page format.

**{ CURRENT or = }**

Invoke the current copy group again. This results in ending printing on the current sheet and resuming on the front side of a new sheet. This is the default.

The page format is restarted, which means that the first input record to go on the new page is printed with the first PRINTLINE command of
the current page format. For example, data that is to be printed as subpage 4 on the sheet might be printed on subpage 1 on the new sheet.

**Note:** The character = can be used for CURRENT.

**FIRST**
Invokes the first copy group in the current form definition.

{ NULL or / }
Retains the current copy group, taking no action. The character / can be used for NULL.

**NEXT**
Invokes the next copy group in the current form definition.

**Note:** If NEXT is specified from the last copy group in the form definition, the first copy group in the form definition is used.

**COPYGROUP cgname**
Uses the named copy group that is defined in the current form definition. The name must contain 1 - 8 alphanumeric characters.

**pageformat options**
Specifies a page format to be invoked if the condition is true.

{ CURRENT or = }
Invokes the current page format again. This results in ending printing on the current sheet and resuming on the front side of a new sheet.

The page format is restarted, which means that the first input record to go on the new page is printed with the first PRINTLINE command of the current page format.

The character = can be used for CURRENT.

**FIRST**
Invokes the first page format in the current page definition.

{ NULL or / }
Retains the current page format, taking no action. The character / can be used for NULL. This is the default.

**NEXT**
Invokes the next page format in the current page definition.

**Note:** If NEXT is specified from the last page format in the page definition, the first page format in the page definition is used.

**PAGEFORMAT pfname**
Uses the named page format that is defined in the current page definition. The name must contain 1 - 8 alphanumeric characters.

**OTHERWISE**
Marks the start of a conditional action that is taken if all preceding WHEN comparisons have proved false. The syntax is the same as the WHEN subcommand, except that the comparison parameters (comparisontype text or CHANGE) are not used. See the WHEN parameters that start with “BEFORE” on page 277 for a description of the parameters.

If the OTHERWISE subcommand is not used within the sequence, no action is taken. This action is the same as if an OTHERWISE NULL NULL is entered.

**Note:** OTHERWISE is not started if any part of the comparison field that is specified by the combination of START and LENGTH is outside the boundaries of the data record.

### Short Form Subcommands

**Notes:**

1. These parameters (START or FLDNUM) have the same meaning as described on the long form of the CONDITION command except that when the parameters are coded here they override the value that is coded on the long form. When not coded here, their values are inherited from the associated long form of the CONDITION command. Long form and short form CONDITION commands are associated by the use of the same condname parameter.

2. No other parameters can be specified on the short form of the CONDITION command. They are inherited from the associated long form.

**START n**

Use this parameter to specify a new starting position for the text to be tested. If this parameter is not coded, the starting position is the same as specified or defaulted in the long form of this CONDITION.

**FLDNUM n**

Use this parameter to specify a new field number for the text to be tested. FLDNUM can be specified only if the LAYOUT or XLAYOUT is coded with a delimiter.

If this parameter is not coded, the starting position is the same as specified or defaulted in the long form of this CONDITION.

---

**DEFINE COLOR Command**

```plaintext
DEFINE colorname
```
Color name of a particular color model such as OCA, RGB, CMYK, HIGHLIGHT, or CIELAB. This name can be used anywhere a color of that model is allowed. For example, a defined color of any color model can be used as text color in the FIELD or PRINTLINE commands, but only a color that is defined as an OCA color can be used as an object placement area color. See the OBCOLOR subcommand in "PRINTLINE Command" on page 423.

**colorname**

Select a 1 - 10 character name. Use this name on the command to identify this color. For example:

```
DEFINE oldblue COLOR OCA brown;
PRINTLINE COLOR oldblue;
```

### Subcommands

**COLOR**

Specifies the color of print for this field that is supported in MO:DCA for the OCA, the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

**OCA ocacolor**

Choose one of the standard OCA colors or synonyms:

- BLUE
- RED
- MAGENTA (or PINK)
- GREEN
- CYAN (or TURQ)
- YELLOW
- BLACK
- BROWN
- MUSTARD
- DARKBLUE (or DBLUE)
- DARKGREEN (or DGREEN)
- DARKTURQ (DTURQ, or DCYAN, or DARKCYAN)
- ORANGE
- PURPLE
- GRAY
- NONE
- DEFAULT

**Note**: In some printer publications, the color turquoise (TURQ) is called cyan, and the color pink (PINK) is called magenta.

**RGB rvalue gvalue bvalue**

Three RGB integer values are used. The first (*rvalue*) represents a value for red, the second (*gvalue*) represents a value for green, and the third (*bvalue*) represents a value for blue. Each of the three integer values can be specified as a percentage 0 - 100.
**Note:** An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

**HIGHLIGHT hvalue COVERAGE cvalue BLACK bvalue**
Indicates the highlight color model. Highlight colors are device-dependent.

You can use an integer within the range of 0 - 65,535 for the *hvalue*.

**Note:** An *hvalue* of 0 indicates that no default value is defined; therefore, the default color of the presentation device is used.

**COVERAGE** indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 - 100 for the *cvalue*. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

**Note:** Fractional values are ignored. If COVERAGE is not specified, a value of 100 is used as a default.

**BLACK** indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 - 100 for the *bvalue*. The amount of black shading that is applied depends on the COVERAGE percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

**Note:** If BLACK is not specified, a value of 0 is used as a default.

**CMYK cvalue mvalue yvalue kvalue**
Defines the cyan/magenta/yellow/black color model. *cvalue* specifies the cyan value. *mvalue* specifies the magenta value. *yvalue* specifies the yellow value. *kvalue* specifies the black value. You can use an integer percentage within the range of 0 - 100 for any of the CMYK values.

**CIELAB Lvalue (-)c1value (-)c2value**
Defines the CIELAB model. Use a range of 0.00 - 100.00 with *Lvalue* to specify the luminance value. Use signed integers from -127 to 127 with *c1value* and *c2value* to specify the chrominance differences.

*Lvalue, c1value, c2value* must be specified in this order. No defaults exist for the subvalues.

**Note:** Do not specify both an OCA color with the COLOR subparameter and an extended color model on the same FIELD or PRINTLINE command. The output is device-dependent and might not be what you expect.

---

**DEFINE QTAG Command (XML only)**

```
DEFINE qtagname QTAG starttag ;
```

**DEFINE QTAG**
Defines a local identifier for a qualified tag that can be used later in the page definition on an XLAYOUT command. A QTAG is a sequence of one or more start-tag names, which taken together identify an XML data element. This identifier is the logical equivalent of the *record ID* on the LAYOUT command
for a record formatting page definition. But, instead of identifying an entire record as the LAYOUT command does, the QTAG identifies a single XML data element.

When used, the local identifier makes the coding of an XLAYOUT command easier by allowing the use of a locally defined name instead of the fully qualified set of start tags. It also makes the XLAYOUT command syntax similar to the LAYOUT command.

 qtagname

The internal name that is assigned to the fully qualified QTAG. This name can be used on the XLAYOUT command to identify the XML data item. This name is not case-sensitive. It can be up to 16 characters in length.

 starttag

An XML element name. This name must match exactly to the element name in the XML data. To preserve the case for the name, put it in quotation marks. Otherwise, the name is folded to uppercase. If necessary, the name is translated to the data type specified or defaulted by the UDTYPE subcommand on the PAGEDEF command. For example, if the page definition is coded on an EBCDIC operating system, but the UDTYPE specifies UTF8, PPFA converts the start tags from EBCDIC code page 500 to UTF-8.

See the "XLAYOUT Command (XML)" on page 446 for an example of using a defined QTAG with an XLAYOUT command.

**DOFONT Command**

```plaintext
DOFONT lname 'ttfname' X16'uuuuuuu...' HEIGHT 10 POINTS

HEIGHT n POINTS

HEIGHT IN CM MM PELS

RATIO percent

ROTATION 0 90 180 270

PRELOAD

UDType of the PAGEDEF

UDType EBCDIC CP 'T1V10500'

CP code-page-name

EBCDIC2 CP code-page-name

CP 'T1000819'

MICR

INLINE

ASCII CP code-page-name

UTF8 CP code-page-name

UTF16
```
The DOFONT command defines a Data Object font and specifies its attributes. Data Object fonts include TrueType and OpenType fonts. A font installer is used to install Data Object fonts and a Resource Access Table (RAT). The RAT contains a table which, when accessed with the full font name that is provided by the user, gives the file access name for the font. All names in the RAT are encoded in UTF-16. For more information, see Using OpenType Fonts in an AFP System.

To use Data Object fonts, do these:

- **Non-PPFA requirements:**
  - You need a printer and a print server (PSF or IPM) that supports Data Object fonts.
  - You must install a Resource Access Table (RAT) and the Data Object fonts that are used. For more information, see Using OpenType Fonts in an AFP System.
- **PPFA requirements:**
  - Define the font by using a DOFONT command.
  - Reference the font in one of the following two ways:
    - Reference the font with the PRINTLINE, LAYOUT, XLAYOUT, FIELD, or FIELD BARCODE commands by using the local name.
    - Use the EXTREF command in the appropriate PAGEFORMAT to create an external reference to any font that must be mapped but is not referenced in the first manner. For example, a BCOCA object can be presented in the page definition and that object can use a font that is not referenced by the page definition. The EXTREF command would allow the font to be mapped.

**DOFONT**

Defines a Data Object Font.

`lname`

Local name for the font. Specifies an unquoted alphanumeric name of 1 - 16 characters. The name must be unique within the page definition. `lname` is the name that is used in the EXTREF, PRINTLINE, LAYOUT, XLAYOUT, FIELD, or FIELD BARCODE commands by using FONT or DOFONT commands that reference the font.

**full font name**

The full font name of the Data Object font; for example, Times New Roman Bold.

`ttfname`

Specifies a quoted, case-sensitive name of the Data Object font to be used in the page definition. Names entered in this form are translated to UTF-16 for matching in the RAT. The name is 1 - 125 characters long and can contain blanks. It is entered in the operating system encoding (for example, ASCII or EBCDIC). The full font name is case-sensitive and must match exactly the full font name in the Data Object font, including blanks. Long font names must be entered as follows:

```
DOFONT Font1 'A very long named Helvetica Font whose name'
     ' will not fit on one line, and maybe '
     ' not even on two lines'
     Height 12 points;
```

Be sure that the blanks are not left out and the case (Upper or Lower) of the characters are correct.

`X'16'uuuuuuuu...'`

The full font name of the Data Object font in Unicode UTF-16BE (Big Endian) encoding. Enter the full Unicode font name in hex digits. Four
hex digits represent one Unicode code point if it isn't a surrogate. It takes eight if it is a surrogate. PPF-A only checks that the entered digits are a multiple of four. The total number of hex digits that are entered is restricted to 500. This allows a font name of up to 125 characters (if no surrogates exist). No translation is done on the name when entered in this format. The Unicode UTF-16BE is case-sensitive and must match exactly the full font name in the Data Object font, including blanks. Long font names must be entered as follows:

```plaintext
DOFONT  Font1  X16 '0041002000760065007200790020006C0066006E0067' '00200066006E006F006E0067' '00200048006500730061006C0074006900630061006E007400650066006900630061' Height 12 points;
```

Be sure that the blanks are included and the case (Upper or Lower) of the characters are correctly encoded.

### Subcommands

**HEIGHT**

```plaintext
HEIGHT—n—POINTS
```

Specifies the height of a Data Object font.

n  A number that specifies the height of the Data Object font. This number can be up to three decimal places.

**Note:** If HEIGHT is not specified, the default is 10 points.

**units**

One of the following standard units:

- **POINTS**  
  Each point is equivalent to 1/72 of an inch (default).

- **IN**  
  Inches

- **CM**  
  Centimeters

- **MM**  
  Millimeters

- **PELS**  
  Pels in the current Logical Units per inch. For example, in 240ths of an inch.

The default units are POINTS.

**RATIO**

```plaintext
RATIO—percent
```

Specifies the ratio of scaling the width relative to the height in a font.
percent
Represents the percent of the normal width of the character that is printed. For example, RATIO 50 yields a font with characters half as wide as normal.

**ROTATION**

```
  ROTATION 0
  90
  180
  270
```

Specifies the rotation of characters in degrees. The specified value is relative to the inline direction of the line to be printed. Valid rotations are 0, 90, 180, and 270. Zero is the default.

**PRELOAD**

If you want the font to be preloaded before starting the print job, specify it here. Preloaded fonts enhance print performance. The printer must support this function.

**UDType**

The UDType subcommand specifies the user's data type and optionally, the code page name for mapping the font.

If UDType is not coded on the DOFONT command, it defaults to the coded or default UDType of the page definition.

**Notes:**
1. Using a code page with a UDType specifies the code page that is used by the application to create the data.
2. To use multiple font mappings for a line in ASCII, UTF8, or UTF16 you must use the FIELD command, since automatic font switching for single and double byte text is only done for EBCDIC data.

**EBCDIC**
Single-byte EBCDIC.
**CP**  Code page name. This parameter is optional here and, if not coded, the default is single-byte EBCDIC code page T1V10500.

```
code-page-name
```

Enter a quoted or unquoted string for the code page name. If the string is unquoted, it can be up to 6 characters. The string is folded to uppercase and the two character prefix T1 is added. If the string is quoted, it can be up to 8 characters, retain the case, and no prefix is added.

**EBCDIC2**

Double byte EBCDIC.

**CP**  Code page name. This parameter is mandatory.

```
code-page-name
```

Enter a quoted or unquoted string for the code page name. If the string is unquoted, it can be up to 6 characters. The string is folded to uppercase and the two character prefix T1 is added. If the string is quoted it can be up to 8 characters, retain the case, and no prefix is added.

**ASCII**

Single-byte ASCII.

**CP**  Code page name. This parameter is optional here and, if not coded, the default is single-byte ASCII code page T1000819.

```
code-page-name
```

Enter a quoted or unquoted string for the code page name. If the string is unquoted, it can be up to 6 characters. The string is folded to uppercase and the two character prefix T1 is added. If the string is quoted, it can be up to 8 characters, retain the case, and no prefix is added.

**UTF8**

Unicode encoding form UTF-8.

**UTF16**

Unicode encoding form UTF-16.

**MICR**

Specifies that this font is to be used for MICR print. MICR print defines that the font is to be used for Magnetic Ink Character Recognition (MICR) printing. When MICR printing is requested, the font needs to be designed for use in MICR applications. MICR text is normally printed with a toner that is mixed with a magnetic material.

**INLINE**

Specifies that this font resource is to be found in an inline resource group when the line data is processed and no additional resource libraries are searched. Its use is intended to correspond to a data object font resource used.
with complex text that is contained in a PTOCA text object. This PTOCA text object is included by using the OBJECT command in PPFA.

Complex text languages provide different layouts for the presentation of text and its storage. Bidirectional (BIDI) languages present text normally from right to left; however, some text such as numbers and embedded Latin, Cyrillic, and Greek scripts, are written from left to right. These languages include Arabic, Urdu, Farsi, and Hebrew.

It is recommended that you place all TrueType/OpenType fonts that are used for complex text rendering in the print file resource group (inline). To ensure that only a font from the print file resource group is used by the presentation system, it is recommended that the INLINE parameter is used for such fonts. The user is responsible for including the font in an inline printfile resource group.

Data Object Font Examples

The following page definition shows these examples of font coding:

- Two AFP fonts are defined, myfont and font1.
- Four Data Object fonts are defined, fontU, font2, font3, and font4.
  - fontU is not referenced in the page definition, but is specified as referenced externally, because it is used in the GOCA object AmFlag.
  - Fonts myfont, font1, font2, font3, and font4 are referenced normally on a PRINTLINE command.
  - font2 uses the UDTYPE subcommand with a named code page.
  - font3 specifies a height and is preloaded.
  - font4 specifies its name in Unicode UTF-16BE (Traditional only).
  - font4 specifies its name in Unicode UTF-16 (Record Format and XML only).

```plaintext
Pagedef ttmpl replace yes;
  FONT myfont 'XZM32F';
  FONT font1 M32F;

  DOFONT fontU 'Unreferenced Font, Used in OBJECT AmFlag';
  DOFONT font2 'Times New Roman' UDTYPE EBCDIC CP 'T1V10037';
  DOFONT font3 'New Gothic Condensed' PRELOAD Height 12;
  DOFONT font4 X16 '004B0065006C007600650070006F0062006F00740068006500730074006900630061006E0074006900630061006E00730065006400760065' /*Helve */
                       '0074006900630061006E00730065006400760065' /*tica */
  DOFONT micr1 'Times New Roman' HEIGHT 12 MICR;
  OBJECT amflg OBXNAME 'AmFlag' OBTYPE goca OBKEEP;

PAGEFORMAT PF1;
  EXTREF fontU;
  Printline Font myfont;
  Printline Font font1 ;
  Printline Font font2 ;
  Printline Font font3 ;
  Printline Font font4 OBJECT amflg;
  Printline;
  FIELD Start 21 Length 16 FONT micr1;

  DOFONT tnr 'Times New Roman WT J' Height 12 UDTYPE EBCDIC CP 'TIV10500';
  DOFONT tnr 'Times New Roman WT J' Height 12 UDTYPE EBCDIC2 CP 'T10300';
  DOFONT tnr 'Times New Roman WT J' Height 12 UDTYPE ASCII;
  DOFONT tnr 'Times New Roman WT J' Height 12 UDTYPE UTF8;
  DOFONT tnr16 'Times New Roman WT J' Height 12 UDTYPE UTF16;
```

288 Page Printer Formatting Aid: User's Guide
You use the DRAWGRAPHIC - BOX command to draw boxes on the page with GOCA (Graphics Object Content Architecture) objects.

**Note:** GOCA boxes require specific microcode in your printer.

You can draw a box of varying attributes and colors at either the current line position or a specified position. DRAWGRAPHIC can be used with the COLOR parameter and DEFINE COLOR to shade a box with a percentage of black or other colors.

**Subcommands**

**GRAPHID**

```
GRAPHID-00
GRAPHID-nn
```

Specified number is used to later identify as the box or set of boxes to be closed by the ENDGRAPHIC. The default is 00.

**POSITION**

```
POSITION - LPOS - NEXT
```

Horizontal and vertical position for first box. This position is relative to the LAYOUT command’s position statement or the current position.

LPOS and CPOS refer to Layout Position and Current Position. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or - value, the position moves that amount from the LAYOUT position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

**BOXSIZE**
Specify the horizontal and optionally, vertical dimensions of the box. The first parameter is required and specifies the horizontal width of the box, which is a fixed size. The second parameter is optional and if given, specifies the fixed vertical depth of the box. If the second parameter is omitted, the box is a variable size or floating box. For a floating box, the depth of the box is determined when the box is closed with an ENDGRAPHIC command.

**ROUNDED**

Size of the rounded corner length is determined by the following parameters:

**MEDIUM**
Medium corner length - equates to a radius of 20 pels at 240 pels/inch or 120 pels at 1440 pels/inch.

**SMALL**
Small corner length - equates to a radius of 10 pels at 240 pels/inch or 60 pels at 1440 pels/inch.

**LARGE**
Large corner length - equates to a radius of 30 pels at 240 pels/inch or 180 pels at 1440 pels/inch.

**MAX**
Maximum corner length gives an arc with a radius that extends half the length of the shortest box side. If boxes are rounded MAX, they cannot be open-ended.

**LINEWT**

Specify either one of the following keywords or the number of lineweights to be used (1 lineweight = 0.01 inches). Specify 0 if you want invisible borders (type and color are then ignored).

**LIGHT**
The same as LINEWT 0.01 inches.

**MEDIUM**
The same as LINEWT 0.02 inches.
**BOLD**

The same as LINEWT 0.03 inches.

**LINETYPE**

Specify one of the following keywords for the border type:

- SOLID
- DOTTED
- SHORTDASH
- DASHDOT
- DBLDOT (double dot)
- LONGDASH
- DSHDBLDOT (dash double dot)

**COLOR**

Color to be used for the box border. The color name must be either one of the pre-defined OCA keywords or the color name from the DEFINE COLOR command.

**COPY**

Repeat the same box at regular intervals either across or down the page. Total number of boxes is one more than the value specified on this parameter.

**Restriction:** If boxes are repeated in the DOWN direction, they cannot be open-ended.
Spacing between the boxes can be specified directly. The default is to have no space between the boxes. If spaces do not exist between the boxes, the common border is shared and not duplicated.

FILL

Allows the option of filling a box with a pre-defined GOCA pattern and optionally specifying a color. The numbering of the boxes is done in the order they are defined within this one command, such as 1, 2, 3. Filling follows the last fill wins rule.

The NOFILL keyword fills ALL boxes with one fill pattern. Specify NOFILL on one box to remove that box's pattern.

Figure 128. Spaced Boxes (not to scale).

Figure 129. Boxes Spaced 0 (not to scale).
For an example of the various GOCA-supported fill patterns, see Figure 167 on page 565.

The NOFILL keyword can be used when a series of boxes are specified as filled and one or more of them are to be left empty. In the example, boxes 1, 2, 4, and 5 are filled with solid blue and box 3 is empty:

```
LAYOUT ...
  Drawgraphic BOX boxsize 1 in .2 in copy down 4
  Linetype solid color green
  FILL ALL SOLID Color Blue
  FILL Box 3 NOFILL;
```

**ALL**

All boxes are filled.

**BOX**

$n$ boxes are numbered starting at 1 for the initial box in this command, and increasing by using the COPY parameter.

**RENDER**

```
RENDER
  PERCEPTUAL
  SATURATION
  RELCM
  ABSCM
```

Subcommand on the DRAWGRAPHIC command that is used to specify the rendering intent (RI) for an object within a page definition. RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.

**Note:** See Chapter 8, “AFP Color Management,” on page 163 for more information about using the RENDER subcommand.

**rendering intent parameter**

Specify the rendering intent for the defined graphic (GOCA) object.

**PERCEPTUAL**

Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

**SATURATION**

Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

**RELCM**

Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.
**ABSCM**

ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

**CMR**

Specify a Color management resource (CMR) and its process mode for a graphics object within the page definition.

**Note:** See [Chapter 8, “AFP Color Management,” on page 163](#) for more information about using the CMR subcommand.

**cmr-lname**

The CMR local name. This name must be defined with a DEFINE CMRNAME command.

**Note:** This parameter must immediately follow the CMR keyword.

**processing mode parameter**

Specify the processing mode for the CMR.

**AUDIT**

Process this CMR as an audit CMR.

**INSTR**

Process this CMR as an instruction CMR.

**LINK**

Process this CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs.

**Examples:** The following code examples show how to define CMRs and rendering intent for graphics objects. Rendering intent and a CMR are defined for Record Format and XML page definitions, which are the only two page definition types for which DRAWGRAPHIC commands are legal.

```plaintext
DEFINE mycmr CMRNAME ...

PAGEDEF cmr11L REPLACE yes;
  FONT f1;
  LAYOUT '11';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
    RENDER relcm CMR myCMR audit;

PAGEDEF cmr11X REPLACE yes;
  FONT f1 TYPE ebcndic;
  XLAYOUT QTAG 'x1';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
    RENDER relcm CMR myCMR audit;
```

---

*Chapter 11. Page Definition Command Reference* 295
You use the DRAWGRAPHIC - LINE command to:

- Draw lines on the page with GOCA (Graphic Character Global Identifier) objects.
**Note:** GOCA lines require specific microcode in your printer.

- Create either one straight line or a series of straight lines from either the current line position or a specified position.

**Subcommands**

**GRAPHID**

```
GRAPHID 00
GRAPHID nn
```

Specifies a number that is used to later identify the graphic line to be closed by the ENDGRAPHIC. The default is 00.

**POSITION**

```
POSITION - LPOS - NEXT
LPOS (+) horiz
LPOS (-) horiz
IN MM
CM
POINTS
PELS
LPOS (+) vert
LPOS (-) vert
IN MM
CM
POINTS
PELS
```

Horizontal and vertical position for the start of the first line. This position is relative to either the Layout Position parameter or the current position.

LPOS and CPOS refer to Layout Position and Current Position. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or - value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

**ACROSS, DOWN, TO**
Specify the line length in either the ACROSS or DOWN directions. If ACROSS is specified, the line length must also be specified. If DOWN is specified and the \textit{n units} value is not entered, the line continues until either a logical page eject is started or an ENDGRAPHIC is found.

Specify the horizontal and vertical ending positions for the line with TO. Used for lines that are point-to-point. The TO position is specified relative to the POSITION parameter values in this command.

\textbf{LINEWT}

Specify either one of the following keywords or the number of lineweights to be used (1 lineweight= 0.01 inches).

\textbf{LIGHT} \\
The same as LINEWT 0.01 inches.

\textbf{MEDIUM} \\
The same as LINEWT 0.02 inches.

\textbf{BOLD} \\
The same as LINEWT 0.03 inches.

\textbf{LINETYPE}
Specify one of the following keywords for the line type:
- SOLID
- DOTTED
- SHORTDASH
- DASHDOT
- DBLDOT (double dot)
- LONGDASH
- DSHDBLDOT (dash double dot)

**COLOR**
Color to be used for the line. The color name must be either one of the pre-defined OCA keywords or the color name from the DEFINE COLOR command.

**COPY**
Repeat the same line at regular intervals either across or down the page. Total number of lines is one more than the value specified on this parameter.

**RENDER**
Subcommand on the DRAWGRAPHIC command that is used to specify the rendering intent (RI) for an object within a page definition. RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.

**Note:** See Chapter 8, “AFP Color Management,” on page 163 for more information about using the RENDER subcommand.

**rendering intent parameter**
Specify the rendering intent for the defined graphic (GOCA) object.
PERCEPTUAL
Perceptual rendering intent. It can be abbreviated as PERCP. With this
rendering intent, gamut mapping is vendor-specific, and colors are
adjusted to give a pleasing appearance. This intent is typically used to
render continuous-tone images.

SATURATION
Saturation rendering intent. It can be abbreviated as SATUR. With this
rendering intent, gamut mapping is vendor-specific, and colors are
adjusted to emphasize saturation. This intent results in vivid colors
and is typically used for business graphics.

RELCM
Media-relative colorimetric rendering intent. In-gamut colors are
rendered accurately, and out-of-gamut colors are mapped to the nearest
value within the gamut. Colors are rendered in regard to the source
white point and are adjusted for the media white point. Therefore,
colors that are printed on two different media with different white
points do not match colorimetrically, but might match visually. This
intent is typically used for vector graphics.

ABSCM
ICC-absolute colorimetric rendering intent. In-gamut colors are
rendered accurately, and out-of-gamut colors are mapped to the nearest
value within the gamut. Colors are rendered only in regard to the
source white point and are not adjusted for the media white point.
Therefore, colors that are printed on two different media with different
white points must match colorimetrically, but might not match visually.
This intent is typically used for logos.

 Specify a Color management resource (CMR) and its process mode for a
graphics object within the page definition.

Note: See Chapter 8, “AFP Color Management,” on page 163 for more
information about using the CMR subcommand.

**cmr-lname**
The CMR local name. This name must be defined with a DEFINE
CMRNAME command.

Note: This parameter must immediately follow the CMR keyword.

**processing mode parameter**
Specify the processing mode for the CMR.

**AUDIT**
Process this CMR as an audit CMR.

**INSTR**
Process this CMR as an instruction CMR.
LINK
Process this CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs.

Examples: The following code examples show how to define CMRs and rendering intent for graphics objects. Rendering intent and a CMR are defined for Record Format and XML page definitions, which are the only two page definition types for which DRAWGRAPHIC commands are legal.

```
DEFINE mycmr CMRNAME ...

PAGEDEF cmr11l REPLACE yes;
  FONT f1;
  LAYOUT 'll';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
  RENDER relcm CMR myCMR audit;

PAGEDEF cmr11x REPLACE yes;
  FONT f1 TYPE ebcdic;
  XLAYOUT QTAG 'x1';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
  RENDER relcm CMR myCMR audit;
```

---

**DRAWGRAPHIC - CIRCLE Command (Record Format and XML only)**

---

**POSITION - LPOS - NEXT**

- **POSITION**
  - **LPOS**
    - **(+)** horiz
    - **(-)** LPOS
  - **(+)** vert LPOS
  - **(-)** CPOS

- **NEXT**
  - **LPOS**
    - **(+)** vert LPOS
    - **(-)** vert LPOS

- **RADIUS**
  - **(+)** LPOS

---

**IN**
- **LINEWT**
  - **MEDIUM**
- **LINETYPE**
  - **SOLID**

**COLOR**
- **colorname**
- **DOTTED**
- **SHORTDASH**
- **DASHDOT**
- **DBLDOT**
- **LONGDASH**
- **DSHDBLDOT**
You use the DRAWGRAPHIC - CIRCLE command to:

- Draw circles on the page with GOCA (Graphics Object Content Architecture) objects.

**Note:** GOCA circles require specific microcode in your printer.

- Create a circle at either a specified radial distance from the last line that is printed or a specified position.

DRAWGRAPHIC can be used with the COLOR parameter and DEFINE COLOR to shade a circle with a percentage of black or other colors.

## Subcommands

**POSITION**
Horizontal and vertical position of the center of the first circle. This position value is relative to either the Layout Position parameter or the current position.

LPOS and CPOS refer to Layout Position and Current Position. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or - value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

**RADIUS**

Specify the circle radius. (The radius is measured from the center of the circle to the middle of the line width.)

**LINEWT**

Specify either one of the following keywords or the number of lineweights to be used (1 lineweight = 0.01 inches). Specify 0 if you want invisible borders (type and color are then ignored).

**LIGHT**

The same as LINEWT 0.01 inches.

**MEDIUM**

The same as LINEWT 0.02 inches.

**BOLD**

The same as LINEWT 0.03 inches.
LINETYPE

Specify one of the following keywords for the line type:
- SOLID
- DOTTED
- SHORTDASH
- DASHDOT
- DBLDOT (double dot)
- LONGDASH
- DSHDBLDOT (dash double dot)

COLOR
Color to be used for the circle border. The color name must be one of the pre-defined OCA keywords or the color name from the DEFINE COLOR command.

COPY

Repeat the same circle at regular intervals either across or down the page. Repeating ACROSS or DOWN with the DIAMETER indication means that the circles are placed to join at one point with the center positions of each being one diameter width apart. See following figures for a pictorial view of repeating circles.

Total number of circles is one more than the value specified on this parameter.

Figure 130. Repeating circles with 0.45 inches spacing (not to scale).
Figure 131. Repeating circles with DIAMETER spacing (not to scale).

FILL

Allows the option of filling a circle with a pre-defined GOCA pattern and optionally specifying a color. Circles are numbered in the order they are defined within this command, such as 1, 2, 3. Filling follows the rule that the last fill wins.

Using the NOFILL keyword fills ALL circles with one fill pattern. Specify NOFILL on one circle to remove that circle's pattern.

For an example of the various GOCA-supported fill patterns, see Figure 167 on page 565.

RENDER

Subcommand on the DRAWGRAPHIC command to specify the rendering intent (RI) for an object within a page definition. RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.
Note: See Chapter 8, “AFP Color Management,” on page 163 for more information about using the RENDER subcommand.

Rendering intent parameter
Specify the rendering intent for the defined graphic (GOCA) object.

**PERCEPTUAL**
Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

**SATURATION**
Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

**RELCM**
Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

**ABSCM**
ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

CMR

Specify a Color management resource (CMR) and its process mode for a graphics object within the page definition.

Note: See Chapter 8, “AFP Color Management,” on page 163 for more information about using the CMR subcommand.

**cmr-lname**
The CMR local name. This name must be defined with a DEFINE CMRNAME command.

Note: This parameter must immediately follow the CMR keyword.

**processing mode parameter**
Specify the processing mode for the CMR.
AUDIT
Process this CMR as an audit CMR.

INSTR
Process this CMR as an instruction CMR.

LINK
Process this CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs.

Examples: The following code examples show how to define CMRs and rendering intent for graphics objects. Rendering intent and a CMR are defined for Record Format and XML page definitions, which are the only two page definition types for which DRA GRAPHIC commands are legal.

```
DEFINE mycmr CMRNAME ... ;

PAGEDEF cmr11L REPLACE yes;
  FONT f1;
  LAYOUT '11';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
  RENDER relcm CMR myCMR audit;

PAGEDEF cmr11X REPLACE yes;
  FONT f1 TYPE ebcdis;
  XLAYOUT QTAG 'x1';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
  RENDER relcm CMR myCMR audit;
```

DRA GRAPHIC - ELLIPSE Command (Record Format and XML only)
You use the DRAWGRAPHIC - ELLIPSE command to:

- Draw ellipses on the page by generating GOCA (Graphics Object Content Architecture) structure fields.

**Note:** GOCA lines require specific microcode in your printer.

- Create an ellipse with a number of positions that show the major and minor axes at a specified distance from the last line printed.

The DRAWGRAPHIC can be used with the COLOR parameter and DEFINE COLOR to shade an ellipse with a percentage of black or other colors.

**Subcommands**

**POSITION**
Horizontal and vertical position of the ellipse.

LPOS and CPOS refer to Layout Position and Current Position. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or - value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

**AXIS1**

The first pair of \( n \) units specifies the location of one point on the ellipse that is specified in relation to the POSITION parameter on this command. This location is specified as if the POSITION parameter is now at (0,0) on a coordinate system. The \( x \) and \( y \) movements are either in the positive or negative direction from the center point at \((0,0)\). For a picture of how AXIS1 is used, see point R,Q in Figure 132 on page 311.

**AXIS2**

The second pair of \( n \) units specifies the location of second point on the ellipse that is specified in relation to the POSITION parameter on this command. This location is specified as if the POSITION parameter is now at \((0,0)\) on a coordinate system. The \( x \) and \( y \) movements are either in the positive or negative direction from the center point at \((0,0)\). For a picture of how AXIS2 is used, see point P,S in Figure 132 on page 311.

**LINEWT**
Specify either a keyword or the number of lineweights to be used (1 lineweight = 0.01 inches). Specify 0 if you want invisible borders (type and color are then ignored).

**LIGHT**
The same as LINEWT 0.01 inches.

**MEDIUM**
The same as LINEWT 0.02 inches.

**BOLD**
The same as LINEWT 0.03 inches.

**LINETYPE**
Specify one of the following keywords for the line type:
- **SOLID**
- **DOTTED**
- **SHORTDASH**
- **DASHDOT**
- **DBLDOT** (double dot)
- **LONGDASH**
- **DSHDBLDOT** (dash double dot)

**COLOR**
Color to be used for the ellipse border. Specify either one of the pre-defined OCA keywords or the color name from the DEFINE COLOR command.

**FILL**
Allows the option of filling an ellipse with a pre-defined GOCA pattern and optionally specifying a filling color. For an example of the various GOCA-supported fill patterns, see Figure 167 on page 565.
The dot in the center of the ellipse shows the POSITION parameter. The asterisk shows the major axis position and star shows the minor axis position.

Rendering intent parameter

Specify the rendering intent for the defined graphic (GOCA) object.

PERCEPTUAL
Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

SATURATION
Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

RELCM
Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

ABSCM
ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

Note: See Chapter 8, “AFP Color Management,” on page 163 for more information about using the RENDER subcommand.
Specify a Color management resource (CMR) and its process mode for a graphics object within the page definition.

**Note:** See [Chapter 8, “AFP Color Management,” on page 163](#) for more information about using the CMR subcommand.

**cmr-lname**

The CMR local name. This name must be defined with a DEFINE CMRNAME command.

**Note:** This parameter must immediately follow the CMR keyword.

**processing mode parameter**

Specify the processing mode for the CMR.

**AUDIT**

Process this CMR as an audit CMR.

**INSTR**

Process this CMR as an instruction CMR.

**LINK**

Process this CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs.

**Examples:** The following code examples show how to define CMRs and rendering intent for graphics objects. Rendering intent and a CMR are defined for Record Format and XML page definitions, which are the only two page definition types for which DRAWGRAPHIC commands are legal.

```assembly
DEFINE mycmr  CMRNAME ...

PAGEDEF cmr11L REPLACE yes;
  FONT f1;
  LAYOUT 'l1';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
    RENDER relcm CMR myCMR audit;

PAGEDEF cmr11X REPLACE yes;
  FONT f1 TYPE ebc;dc;
  XLAYOUT QTAG 'x1';
  DRAWGRAPHIC BOX BOXSIZE 1 in 2 in
    RENDER relcm CMR myCMR audit;
```

**ENDGRAPHIC Command (Record Format and XML only)**
You use the ENDGRAPHIC command to end all active graphics with a matching graphic ID. An active graphic is one that is started but not ended; for example, a vertical line or a box with no vertical size.

**Subcommands**

**GRAPHID**

ID must match one previously defined in a DRAWGRAPHIC command. If no GRAPHID is specified, all DRAWGRAPHIC commands that have no GRAPHID are closed (for example, GRAPHID 00).

**NEXT**

Specifies the layout is to be positioned down (on the logical page) one line (as defined in the LINESP subcommand of the last SETUNITS command) from the previous field. The LINESP subcommand of the SETUNITS command establishes the distance from one line to the next.

**LPOS CPOS**

LPOS and CPOS refer to Layout Position and Current Position. If LPOS is used alone, the position is used exactly at the same position as is specified on the LAYOUT command. If it is used with a + or - value, the position moves that amount from the Layout position. The same is true for Current position except that the position is taken from the previous FIELD or DRAWGRAPHIC command.

**vert**

This value is relative to the Layout position. If not specified, the graphics are closed one line spacing from the Layout position.

**ENDSUBPAGE Command (Traditional Only)**

The ENDSUBPAGE command is used to identify the end of a subpage for conditional processing.

You can specify the ENDSUBPAGE command at any point in a page definition command stream where a PRINTLINE or LAYOUT command can occur. However, you must not enter the ENDSUBPAGE command between a PRINTLINE or LAYOUT command and its associated FIELD or CONDITION command.
If an ENDSUBPAGE command is not specified, the entire page format is treated as one subpage.

**EXTREF Command**

The EXTREF command specifies resources that are to be mapped in the page. It is a way in PPFA to map objects that wouldn’t otherwise be mapped. If an object contains another mapped object, the contained object must be mapped, but PPFA does not automatically map that object.

For example, if you presented a GOCA object that contained a mapped font, a CMR, another object, or a combination, those resources are not mapped in the page. The EXTREF command can be used to map this required resource.

**Note:** Even though a font is mapped automatically by PPFA when it is used internally in the page definition on a PRINTLINE, LAYOUT, XLAYOUT, or FIELD command, a font that is used in an included object, such as a GOCA object, is not known to PPFA and consequently not mapped. In that case, the user must define that font with either a FONT or DOFONT command and map it with the EXTREF command.

You use the EXTREF command to map resources that are used in included objects that are not known to PPFA and therefore not mapped.

**Subcommands**

**FONT**

Specify a font to be mapped. This parameter is the default if FONT, OB2CMR, or OB2R are not specified.

\( \text{Iname} \)

Local name for a font. Specifies an unquoted alphanumeric name of 1 - 16 characters that is used in the page definition. The name must be unique within the page definition. \text{Iname} is the name that is used in the FONT or DOFONT commands that define the font.

**OB2CMR**

\[ \text{OB2CMR} \]

Specify a CMR to be mapped.
Specify a Color management resource (CMR) and its process mode for a data object specified within an included object. CMRs are secondary objects when used at this level. An object that is specified here is mapped with object scope.

**Note:** See Chapter 8, “AFP Color Management,” on page 163 for more information about using the CMR subcommand.

`iname`

The CMR local name. This name must be defined with a DEFINE CMR command.

**processing mode parameter**

Specify the processing mode for the CMR.

**AUDIT**

CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

**INSTR**

CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer that uses a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer’s website.

**LINK**

This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

**OB2R**

```
OB2R

<table>
<thead>
<tr>
<th>OB2R</th>
<th>OB2XNAME</th>
<th>x2name</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2name</td>
<td>'i2name'</td>
<td></td>
</tr>
<tr>
<td>'i2name'</td>
<td>'i2name'</td>
<td></td>
</tr>
<tr>
<td>C'i2name'</td>
<td>'x2name'</td>
<td>C'i2name'</td>
</tr>
<tr>
<td>E'i2name'</td>
<td>'x2name'</td>
<td>E'i2name'</td>
</tr>
<tr>
<td>A'i2name'</td>
<td>'x2name'</td>
<td>A'i2name'</td>
</tr>
<tr>
<td>X'hhhh'</td>
<td>'x2name'</td>
<td>X'hhhh'</td>
</tr>
<tr>
<td>U8'x2name'</td>
<td>'x2name'</td>
<td>U8'x2name'</td>
</tr>
<tr>
<td>U16'x2name'</td>
<td>'x2name'</td>
<td>U16'x2name'</td>
</tr>
<tr>
<td>x8'hhhh'</td>
<td>'x2name'</td>
<td>x8'hhhh'</td>
</tr>
<tr>
<td>x16'hhhh'</td>
<td>'x2name'</td>
<td>x16'hhhh'</td>
</tr>
</tbody>
</table>
```
Specify a secondary object to be mapped.

If an included object contains a reference to one or more secondary objects, you must identify them. Specify the internal name for the secondary resource as specified in the included resource. If the internal name contains special characters such as periods or blanks, then quotation marks must surround the name.

\( \text{i2name} \)
- Unquoted name up to 250 characters long is folded to uppercase and translated into EBCDIC if necessary.
- 'i2name'
  - Quoted name up to 250 characters long is accepted as-is with no case folding or translation.
- C'i2name'
  - Quoted name with a C for Character is treated the same as a quoted name of up to 250 characters. No folding or translation is done.
- E'i2name'
  - Quoted name with an E for EBCDIC entered with up to 250 characters is accepted as-is if on an EBCDIC operating system or translated to EBCDIC if on an ASCII operating system. The translation is made with no case folding.
- A'i2name'
  - Quoted name with an A for ASCII entered with up to 250 single-byte characters is accepted as-is if on an ASCII operating system or translated to ASCII if on an EBCDIC operating system. The translation is made with no case folding.
- X'hhhh'
  - Quoted name with an X for Hexadecimal entered with up to 500 hexadecimal characters. The characters are translated to hexadecimal, but no assumptions of data type are made.

\( \text{OB2XNAME x2name} \)
- Specifies the external name for a secondary resource object. The name can be up to 250 characters. If the name contains special characters or blanks, then quotation marks must surround the name.
- x2name
  - Unquoted name up to 250 characters long is folded to uppercase and translated into EBCDIC if necessary.
- 'x2name'
  - Quoted name up to 250 characters long is accepted as-is with no case folding or translation.
- C'x2name'
  - Quoted name with a C for Character is treated the same as a quoted name up to 250 characters. No folding or translation is done.
- E'x2name'
  - Quoted name with an E for EBCDIC entered with up to 250 single-byte characters is accepted as-is if on an EBCDIC operating system or translated to EBCDIC if on an ASCII operating system. The translation is made with no case folding.
- A'x2name'
  - Quoted name with an A for ASCII entered with up to 250 single-byte
characters is accepted as-is on an ASCII operating system or translated to ASCII if on an EBCDIC operating system. The translation is made with no case folding.

X'hhhh'
Quoted name with an X for Hexadecimal entered with up to 500 hexadecimal characters. The characters are translated to hexadecimal, but no assumption of data type is made.

U8'x2name'
Quoted name with a U8 for UTF-8 entered with up to 250 single-byte characters is translated to UTF-8.

X8'hhhh'
Quoted name with an X8 for UTF-8 HEX entered with up to 500 single-byte hexadecimal characters is translated to hexadecimal and assumed to be data type UTF-8. A multiple of two hexadecimal characters must be entered.

U16'x2name'
Quoted name with a U16 for UTF-16 entered with up to 125 single-byte characters is translated to UTF-16.

X16'hhhh'
Quoted name with an X16 for UTF-16 HEX entered with up to 500 single-byte hexadecimal characters is translated to hexadecimal and assumed to be data type UTF-16. A multiple of four hexadecimal characters entered.

0B2ID n | type-name

Component type identifier for secondary resource that uses an object type number as specified in Object type list adjustments. Use an object type number from the Component-ID column or a type name from the Type name of the following table:

Table 12. Object Types that can be referenced as Secondary Resources

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OID Type-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFRO</td>
<td>26</td>
<td>PDF Resource Object (new)</td>
</tr>
<tr>
<td>RESCLRPRO</td>
<td>46</td>
<td>Resident Color Profile Resource Object</td>
</tr>
<tr>
<td>IOCAFS45RO</td>
<td>47</td>
<td>IOCA FS45 Resource Object Tile (new)</td>
</tr>
</tbody>
</table>

Example: In the following example, the fonts ocAf, fontA, and fontABI and the CMR rvtc are mapped in the Object Environment Group (OEG) of an object that is being included in the page.

Without the EXTREF commands, only the font varb would be mapped because only it is being called out in the PPFA source code. Also, without the EXTREF command, the CMR rvtc is not mapped.

Notes:
1. The fonts that are coded in the EXTREF commands must also be defined in a FONT command.
2. If you code rtvc with a CMR command (as in the commented out CMR
command), it is mapped but is active for the entire page; however, you want it
to be active only for the object in whose OEG it is mapped.
PAGEDEF cmr42 replace yes;
FONT varb gt10
;
/*Variable data
FONT ocaf CS N40090 CP 000395; /* mapped in OEG
DOFONT fontA ’Arial’ Height 12;
DOFONT fontABI ’Arial Bold Italic’
UDTYPE EBCDIC CP ’T1V10500’;

*/
*/

DEFINE rvtc CMRNAME
’RevVideoTC001.000@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@’
’@@@@@@’
SETUNITS LINESP .25 in ;

/* Line spacing

*/

PAGEFORMAT rept1
TOPMARGIN .25 in;
EXTREF ocaf;
EXTREF fontA;
EXTREF fontABI;
EXTREF OB2CMR rvtc instr;
/*

CMR rvtc instr;
**/
LAYOUT ’startline’ BODY newpage POSITION .5 in SAME FONT varb;
LAYOUT ’plaindata’ BODY
POSITION .5 in NEXT FONT varb;

FIELD Command
FIELD Command (Traditional only)

►► FIELD

TEXT

▼

C
X
Dn
G
K
LENGTH

'text'

►

L(m)
n

START n
POSITION

CURRENT or *

CURRENT or *

DIRECTION

ACROSS

DIRECTION

DOWN
BACK
UP

►

►
POSITION

x-pos
–
CURRENT or *

y-pos
–
NEXT
CURRENT or *

►

►
FONT

name1

SUPPRESSION name
, name2

Chapter 11. Page Definition Command Reference

319


Other BARCODE Parameters:

- Common BARCODE Parameters
- Common 2D BARCODE Parameters
- Concatenated BARCODE Parameters

Common BARCODE Parameters:

- MOD \(_n\)
- HRI
- HRIFONT
- SSASTERISK
- WIDTH \(_n\)
- HEIGHT \(_n\)
- MODWIDTH

Other BARCODE Parameters:

- COLOR
- RGB
- HIGHLIGHT
- CMYK
- CIELAB
- BARCODE
- BCCOLOR
- SUPPBLANKS
- RATIO
Common 2D BARCODE Parameters:

Data Matrix 2D Parameters:

MaxiCode 2D Parameters:
The FIELD command identifies a field in a data record or supplies a field of constant text, and positions where the field is on the page. More than one position on the page can be specified.

FIELD commands:
- Are subordinate to a PRINTLINE command (Traditional), LAYOUT command (Record Format), or LAYOUT subcommand (XML)
- Must follow a PRINTLINE command (Traditional) or a LAYOUT command (Record Format)
- Must contain either a LENGTH subcommand or a TEXT subcommand (Traditional only)

The FONT, DIRECTION, and COLOR subcommands do not have fixed defaults. If any of the subcommands are omitted, the value for the omitted subcommand is obtained from corresponding subcommand in the PRINTLINE command (Traditional), LAYOUT command (Record Format), or LAYOUT subcommand (XML).

Subcommands

START

Specifies the starting byte in the data record for the field you want.

\[ n \]

Specifies the number of bytes from the first data byte in the record to be used as the starting point of the field. The first data byte position of an input record is 1.

Note: The carriage-control character, table-reference character, and record ID are not considered data.

* Denotes the next byte after the field identified in the previous FIELD command, excluding FIELD commands with constant TEXT.
+ n
Adds the value of n to the * byte position.

- n
Subtracts the value of n from the * byte position.

If START is omitted and LENGTH is specified, then START * is assumed.

LENGTH n
Specifies the number (n) of bytes to process from the data record, beginning
with the position specified in START.

Record Format: When the maximum length of the field is determined, the
print server truncates all of the fields not containing data.

TEXT

```
| TEXT | Dn | ▼text ▼ |
```

Specifies the constant text that is to be printed in the output. A maximum of
65,535 bytes of text can be provided in one page format.

Note: This text is considered constant in that the same text is printed each
time. In regard to the CONSTANT command within a form definition, this text
is considered variable because the text prints only where variable data is
allowed to print.

Dn  Specifies the number of times the text is to be duplicated (use a decimal
number). The maximum times the text is repeated varies depending on the
size of the text. The default is 1.

text type
Specifies the type of text.

C  Indicates that the text contains single-byte code characters, which
include all Roman alphabetic characters (for example, the ones used for
English). Any valid character code can be specified, including blanks.
This is the default.

X  Indicates that the text contains hexadecimal codes (in groups of two
hexadecimal codes) that specify values from X'00' - X'FE'.

G  Indicates that the text contains double-byte code characters (for
example, kanji characters).
Characters in type G text must start with shift-out (SO X'0E') and end
with shift-in (SI X'0F') characters within opening and closing
apostrophes (X'7D' for EBCDIC operating systems and X'27 ' for ASCII
operating systems).

K  Indicates that the text contains kanji numbers that are enclosed in
apostrophes. Kanji numbers are separated by commas:
K' 321,400'

Valid double-byte character set (DBCS) codes are from X'41' - X'FE' for each
byte. Code X'4040' (blank) is the only exception.
Valid: X'4040', X'4141', X'41FE', X'FE41', X'FEFE'

Invalid: X'2040', X'413E', X'4100' X'7F00', X'FE3E'

L\((m)\)
Specifies the length of text (use a decimal number in parentheses). When the actual length of the text is different from \(m\), the \(m\) specification is accepted. That is, the text is either padded with blanks to the right or truncated.

'\text'
Specifies the text.

Examples:
- When TEXT 2C(3)'AB' is specified, 'AB AB ' is generated. The blanks are generated because of the (3) specification.
- TEXT 2C(1)'AB' generates 'AA', truncating the Bs.

PAGENUM \(n\) (Record Format and XML)

Although parameters are specified as optional, at least one must be specified. Page numbers can be set to start with the value specified as \(n\); otherwise, they follow the specification that is made in the PAGEDEF or PAGEFORMAT command.

The POSITION parameters that are specified with the PAGENUM parameter reflect the position of the page number only.

If you do not want a page number to be printed, either do not use this parameter or specify NOPRINT.

The RESET parameter is only used when you want to reset the page number that is to be used with this page.

Note: You must define a font that specifies the font type to be used for printing page numbers.

FLDNUM (Record Format and XML)

This keyword must only be used if the DELIMITER field was used in the LAYOUT command. Fields cannot be counted without specifying delimiters in the database.

To allow for the identification of a part of a field that is field delimited, you can specify the starting position (from the delimiter), and optionally the length of the part of the field you want to use. The LENGTH default is to use the entire remainder of the field from the start position to the ending delimiter.

RECID (Record Format only)
You use this keyword to access characters in the first $n$ characters of a record. This area is reserved for the record identifier, and all other field starts and lengths are calculated after this area. These starts and lengths reference only the area within the record ID.

If no record length is specified, the remaining bytes of the $n$-byte field are assumed.

**STAG (XML only)**

You use this keyword to access characters in the START tag. It also includes the < and > delimiters; position 1 is always the < delimiter.

If no record length is specified, the remaining bytes of the START tag are assumed. If no START is specified, 1 is assumed.

LENGTH * means that the remainder of the field is used for the length.

**ATTR (XML only)**

You use this keyword to access attribute values from the data. Multiple attribute fields can access the same attribute, which allows subsets of the value to be printed.

If no record length is specified, the remaining bytes of the attribute are assumed. If not START is specified, 1 is assumed.

`aname`

The attribute name. To preserve the case, enter the name in quotation marks. The name is converted to the data type you specify, by using UDTYPE on the page definition, or it is defaulted.

START $n$

The starting position of the attribute to extract the data. If this parameter is omitted, position 1 is assumed.

LENGTH $n$

The length of the attribute to be placed. If this parameter is omitted or LENGTH * is coded, the rest of the field is assumed for the length.
Specifies the starting position of the field in the printout.

**x-pos**

Do not mix x-pos specifications with CURRENT or * except in ACROSS fields.

- Specifies that the x value is negative.

x Specifies the horizontal offset for the starting print position that is relative to the *printline starting position*. The choices are IN, MM, CM, POINTS, or PELS.

The default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

The PELS measurement equals one L-unit or 1/240 of an inch, depending on whether the PELSPERINCH parameter is specified previously.

**APOS**

Specifies that the x pos parameter that follows is absolute. The x pos parameter is mandatory and must be positive.

**CPOS**

Specifies that the x pos parameter that follows is relative to the current position. This parameter can be negative.

**LPOS**

Specifies that the x pos parameter that follows is relative to the XLAYOUT position. This parameter can be negative.
CURRENT

Specifies that the inline offset (relative to the field's direction) is the end of the previous field. For the first field, use the PRINTLINE offset. This is the default.

Note: The meaning of CURRENT differs from the meaning of the PRINTLINE command parameter (Traditional) or a LAYOUT command parameter (Record Format) SAME.

* Alternative for CURRENT.

y-pos

Do not mix y-pos specifications with CURRENT or * except in ACROSS fields.

- Specifies that the y value is negative.

y

Specifies the vertical offset for the starting print position that is relative to the printline starting position. The choices are IN, MM, CM, POINTS, or PELS.

The default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

NEXT

Specifies a field that is positioned down one line in the baseline direction (as defined in the SETUNITS command LINESP subcommand) from the previous field.

Use NEXT only in ACROSS fields.

CURRENT

Specifies that the baseline offset (relative to the field's direction) is the same as the previous field. That is, the baseline position does not change. For the first field, use the PRINTLINE (Traditional) or a LAYOUT (Record Format) offset. This is the default.

* Alternative for CURRENT.

FONT

Defines the font to be used for the field.

name1

Specifies the local name of a font that is used to print the data. This font must be defined in a previous FONT or DOFONT command in this page definition.

If Shift-Out, Shift-In (SOSI) processing is used, name1 must be the single-byte font.

name2

Specify only when you are using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the field. name2 must be the double-byte font.

Note: name2 is only valid with EBCDIC data.
Notes:
1. If this subcommand is not specified, the font that is specified in the preceding PRINTLINE command (Traditional) or a LAYOUT command (Record Format) is used. If neither is specified, the print server assigns a font.
2. Record Format only: For ASCII, UTF8, or UTF16, the entire PRINTLINE command must be one font. To use multiple font mappings for a line in ASCII, UTF8, or UTF16, you must use the FIELD command.

ALIGN LEFT | RIGHT (Record Format and XML only)

The data in this field is left or right-aligned to the x position that is specified in the horizontal POSITION parameter.

DIRECTION

Specifies the print direction of the field, relative to the upper-left corner as you view the logical page. If this subcommand is omitted, the direction that is specified in the governing PRINTLINE command is used.

ACROSS
The page is printed with the characters added from left to right on the page, and the lines are added from the top to the bottom.

DOWN
The page is printed with the characters added from top to bottom on the page, and the lines are added are from the right to the left.

BACK
The page is printed with the characters added from right to left on the page, and the lines are added from the bottom to the top.

UP
The page is printed with the characters added from bottom to top on the page, and the lines are added from the left to the right.

If DIRECTION is not zero, then the position is relative to the rotated origin of the page.

Note: Not all printers can print in all directions. See your printer documentation for more information.

SUPPRESSION

Specifies that this text field can be suppressed (not valid for bar codes).
name

Specifies the name of a field to be suppressed.

Printing of this field is suppressed if this name is identified by a SUPPRESSION command within the form definition.

The same name can be used in one or more fields to suppress these fields as a group.

COLOR

<table>
<thead>
<tr>
<th>COLOR—colormap</th>
<th>RGB—rvalue—gvalue—bvalue</th>
<th>HIGHLIGHT—hvalue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COVERAGEx—cvalue</td>
<td>BLACK—bvalue</td>
</tr>
<tr>
<td>CMYK—cvalue—mvalue—yvalue—kvalue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIELAB—lvalue</td>
<td>(—)clvalue</td>
<td>(—)c2value</td>
</tr>
</tbody>
</table>

Specifies an OCA or defined color for the text of this field. This subcommand is recognized only by printers that support multiple-color printing. See your printer publication for more information.

colormap

Values for colormap can be a defined color (see “DEFINE COLOR Command” on page 280), or an OCA colormap. Values for OCA colormaps are:

BLUE
RED
MAGENTA (or PINK)
GREEN
CYAN (or TURQ)
YELLOW
BLACK
BROWN
MUSTARD
DARKBLUE (or DBLUE)
DARKGREEN (or DGREEN)
DARKTURQ (DTURQ, or DCYAN, or DARKCYAN)
ORANGE
PURPLE
GRAY
NONE
DEFAULT

The color choices depend on the printer.

Note: In some printer publications, the color turquoise (TURQ) is called cyan, and the color pink (PINK) is called magenta.

color model

Specifies the color of print for this field that is supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model. This example shows the color model with the FIELD command:

```
FIELD START 1 LENGTH 5
    COLOR BLUE ;
FIELD START 1 LENGTH 1
```
RGB rvalue gvalue bvalue

Three RGB integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values can be specified as a percentage 0 - 100.

Note: An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

HIGHLIGHT hvalue COVERAGE cvalue BLACK bvalue

Indicates the highlight color model. Highlight colors are device-dependent.

You can use an integer within the range of 0 - 65535 for the hvalue.

Note: An hvalue of 0 indicates that no default value is defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 - 100 for the cvalue. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If COVERAGE is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 - 100 for the bvalue. The amount of black shading that is applied depends on the COVERAGE percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If BLACK is not specified, a value of 0 is used as a default.

CMYK cvalue mvalue yvalue kvalue
Defines the cyan/magenta/yellow/black color model. Cvalue specifies the cyan value. Mvalue specifies the magenta value. Yvalue specifies the yellow value. Kvalue specifies the black value. You can use an integer percentage within the range of 0 to 100 for any of the CMYK values.

**CIELAB** Lvalue (-)c1value (-)c2value

Defines the CIELAB model. Use a range of 0.00 - 100.00 with Lvalue to specify the luminance value. Use signed integers from -127 to 127 with c1value and c2value to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. No defaults exist for the subvalues.

**Note:** Do not specify both an OCA color with the COLOR subparameter and an extended color model on the same FIELD or PRINTLINE command. The output is device-dependent and might not be what you expect.

**BARCODE**

Specifies a bar code in a page definition. The following are valid bar code type-names:
- CODE39 (same as 1)
- MSI (same as 2)
- UPCA (same as 3)
- UPCE (same as 5)
- UPC2SUPP (same as 6)
- UPC5SUPP (same as 7)
- EAN8 (same as 8)
- EAN13 (same as 9)
- IND2OF5 (same as 10)
- MAT2OF5 (same as 11)
- ITL2OF5 (same as 12)
- CDB2OF7 (same as 13)
- CODE128 (same as 17)
- EAN2SUP (same as 22)
- EAN5SUP (same as 23)
- POSTNET (same as 24)
- RM4SCC (same as 26)
- JPOSTAL (same as 27)
- 2DMATRIX (same as 28)
- 2DMAXI (same as 29)
- 2DPDF417 (same as 30)
- APOSTAL (same as 31)
- 2DQRCODE (same as 32)
- CODE93 (same as 33)
- US4STATE (same as 34)
• REDTAG (same as 35)
• DATABAR (same as 36)

The bar code name can be 1 - 8 characters long. See your printer documentation for more information about bar code support. Ensure that the bar code fits on the page or you errors occur at print time.

Read your printer hardware documentation before you use bar codes. The documentation indicates which bar code types, modifiers, MODWIDTH, element heights, and ratio values are valid for the printer.

PPFA does minimal verification of the bar code values. If you use the MOD, HEIGHT, MODWIDTH, and RATIO parameters, ensure that the values you specify are valid for your printer.

For printer optimization, specify BARCODE name options in the first instance of a specific type of bar code. If this type is used again, position it as usual with START, LENGTH, and POSITION, but specify the bar code information by using only BARCODE same-name-as-previously. The BARCODE subcommand is recognized only by printers that support BCoca bar code printing; see your printer documentation for more information.

**Bar code concatenation**

The concatenated bar code function allows the user to collect bar code data from different fields, records, or both to be concatenated in generating a bar code object.

For example, the hyphen in a nine-digit ZIP code, `aaaaa-bbbb`, is not a valid character in a POSTNET bar code. The concatenated bar code function allows a user to specify a FIELD for the `aaaaa` and a FIELD for the `bbbb` and concatenate them together into one bar code. For 2D bar codes, which can contain many bytes of data, multiple records of data can be concatenated together into one bar code.

The defining bar code fully defines a bar code, specifying any necessary placement or descriptive parameters. A defining bar code can start the collection of bar code data if it specifies a BARCODE name and the BCDSYMB keyword with a symbol name.

The continuation bar code defines bar code data that is to be added to a bar code data collection started with a defining bar code with BCDSYMB coded. This bar code uses the defining BARCODE name and BCDSYMB name but does not specify the TYPE or BARCODE parameters because it uses the information from the defining bar code.

**Example 1**

POSTNET 9-digit bar code split in two fields `aaaaa-bbbb`:

```plaintext
PAGEDEF BC1P REPLACE YES;
FONT afont GT10;
PAGEFORMAT example1;
PRINTLINE POSITION 4 in 1.5 in;
FIELD START 1 LENGTH 15 POSITION 0 IN 0 IN;
FIELD START 16 LENGTH 20 POSITION 0 IN NEXT;
FIELD START 36 LENGTH 15 POSITION 0 IN NEXT;
FIELD START 51 LENGTH 5 POSITION 0 IN 0.70 IN
BARCODE zip54 TYPE POSTNET MOD 1 BCDSYMB datacoll;
FIELD START 57 LENGTH 4 BARCODE zip54 BCDSYMB datacoll;
```

**Example 2**

2D bar code with data bytes across multiple records:
Notes:

1. A bar code cannot be positioned on or outside of the logical page.

   For example, the following page definition results in a position of 0 0, which is on the logical page boundary. The following example results in a printer error:

   ```
   PAGEDEF XMPL1 REPLACE YES;
   SETUNITS 1 IN 1 IN LINESP 6 LPI;
   FONT FNT1 CR10;
   FONT FNT2 CR10;
   PAGEFORMAT IBMSSN WIDTH 9.5 HEIGHT 4.0 /* PORTRAIT */
   DIRECTION ACROSS;
   PRINTLINE
   FONT FNT1 REPEAT 1 CHANNEL 1 POSITION 0 0;
   FIELD START 671 LENGTH 13 POSITION 0 0
   BARCODE 3OF9
   TYPE 1 MODWIDTH 17
   SSASTERISK ON;
   ```

2. If you want to suppress blanks, use the SUPPBLANKS parameter.

3. SUPPBLANKS for bar code data cause the entire field to be ignored if it is all blanks.

4. Bar code Symbol Size

   The height of the bar code symbol is controlled by the bar code symbology definition and by various BCOCNA parameters. The width of the symbol is typically dependent on the amount of data to be encoded and by choices that are made in various BCOCNA parameters.

   **Linear Symbologies**

   The element-height and height-multiplier parameters specify the height of the symbol. For some bar code types, these parameters also include the height of the human-readable interpretation (HRI). See the description of the element-height parameter in the Bar Code Object Content Architecture Reference for a description of the height for specific linear symbols. Some bar code symbologies (Australia Post Bar Code, Japan Postal Bar Code, POSTNET, RM4SCC, and REDTAG) explicitly specify the bar code symbol height; in this case, the element-height and height-multiplier parameters are ignored.

   **Two-Dimensional Matrix Symbologies**

   The MaxiCode symbology specifies a fixed physical size, nominally 28.14 mm wide by 26.91 mm high; the module-width, element-height, and height-multiplier parameters are ignored for MaxiCode values.

   Data Matrix symbols are rectangular and are made up of a pattern of light and dark squares (called modules). The size of each module is specified in the module-width parameter and the number of rows and columns of these modules is controlled by the
desired-number-of-rows and desired-row-size parameters and the amount of data to be encoded. The element-height and height-multiplier parameters are ignored for Data Matrix symbols.

QR Code symbols are square and are made up of a pattern of light and dark squares (called modules). The size of each module is specified in the module-width parameter; the number of rows and columns of these modules is controlled by the version parameter, the error correction level that is selected, and the amount of data to be encoded. The element-height and height-multiplier parameters are ignored for QR Code symbols.

Two-Dimensional Stacked Symbologies
PDF417 symbols are rectangular and are made up of a pattern of light and dark rectangles (called modules). The size of each module is specified in the module-width, element-height, and height-multiplier parameters and the number of rows and columns of these modules is controlled by the data-symbols and rows parameters and the amount of data to be encoded. A PDF417 symbol must contain at least three rows.

For more information about bar codes, see Appendix D, “More About Bar Code Parameters,” on page 477 and Bar Code Object Content Architecture Reference.

name
Specifies a specific bar code name to be included in a page definition. The name is required if BARCODE is used to reference or continue the bar code later, as is done for bar code concatenation.

TYPE \{ n | type-name \}
Specifies the type of bar code symbol to be generated.

Note: If a type indicates “(same as n)”, you can substitute the specified number for the character name.
The following bar code types are supported:

type-name
Specifies a specific bar code type name to be included in a page definition.

CODE39 (same as 1)

MSI (same as 2)
Specifies a bar code type of modified Plessey code.

UPCA (same as 3)
Specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version A.

UPCE (same as 5)
Specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version E.

UPC2SUPP (same as 6)
Specifies a bar code type of Universal Product Code (United States) two-digit Supplemental (periodicals).

UPC5SUPP (same as 7)
Specifies a bar code type of Universal Product Code (United States) five-digit Supplemental (paperbacks).
EAN8 (same as 8)  
Specifies a bar code type of European Article Numbering 8 (includes Japanese Article Numbering-short).

EAN13 (same as 9)  
Specifies a bar code type of European Article Numbering 13 (includes Japanese Article Numbering-standard).

IND2OF5 (same as 10)  
Specifies a bar code type of Industrial 2-of-5.

MAT2OF5 (same as 11)  
Specifies a bar code type of Matrix 2-of-5.

ITL2OF5 (same as 12)  
 Specifies a bar code type of Interleaved 2-of-5, Automatic Identification Manufacturers Uniform Symbol Specification-I 2/5.

CDB2OF7 (same as 13)  

CODE128 (same as 17)  

Note: A subset of CODE128 exists that is called EAN128. These EAN128 bar codes can be produced with PPFA by specifying CODE128 for the bar code type in the PAGEDEF and including the extra parts of the bar code in the data. The UCC-128 bar code format is:

```
startcode FNC1 ai nnnnnnnnnnnnnnnnnn m c stopchar
```

The string of ns represents the bar code data. The start code, stop character, and 'c' value are generated by the printer microcode for BCoca bar codes. The FNC1 is a hexadecimal 8F character. The ai is an application identifier and needs to be defined for use by each EAN128 application. The m is a modulo 10 check digit that must be calculated by the application and included in the bar code data.

Not all printers generate the EAN128 bar codes; thus, you might need to verify that the bar code that is produced in this manner is readable by your bar code scanner.

For more information about the EAN128 bar codes, see the [AFP Consortium web page](http://www.afpcinc.org).

EAN2SUP (same as 22)  
Specifies a bar code type of European Article Numbering, Two-digit Supplemental.

EAN5SUB (same as 23)  
Specifies a bar code type of European Article Numbering, Five-digit Supplemental.

POSTNET (same as 24)  
Specifies a bar code type of POSTal Numeric Encoding Technique
(United States Postal Service), and defines specific values for the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields.

Note: POSTNET MOD 4 is normally called PLANET bar code. POSTNET MOD 4 is supported by PFF and some AFP printers.

RM4SCC (same as 26)
Specifies a 4-state customer code that is defined by the Royal Mail Postal Service of England for bar coding postal code information.

JPOSTAL (same as 27)
A complete Japan Postal Bar Code symbol consists of a set of distinct bars and spaces for each character followed by a modulo 19 checksum character and enclosed by a unique start character, stop character and quiet zones.

2DMATRIX (same as 28)
Specifies a Data Matrix two-dimensional bar code. Two-dimensional matrix symbologies (sometimes called area symbologies) allow large amounts of information to be encoded in a two-dimensional matrix. These symbologies are typically rectangular and require a quiet zone around all four sides; for example, the Data Matrix symbology requires a quiet zone at least one module wide around the symbol. Two-dimensional matrix symbologies use extensive data compaction and error correction codes, allowing large amounts of character or binary data to be encoded.

2DMAXI (same as 29)
Specifies a MaxiCode two-dimensional stacked bar code. Two-dimensional stacked symbologies allow large amounts of information to be encoded by effectively stacking short one-dimensional symbols in a row/column arrangement. This arrangement reduces the amount of space that is typically used by conventional linear bar code symbols and allows for a large variety of rectangular bar code shapes.

2DPDF417 (same as 30)
Specifies a PDF417 two-dimensional stacked bar code. Two-dimensional stacked symbologies allow large amounts of information to be encoded by effectively stacking short one-dimensional symbols in a row/column arrangement. This arrangement reduces the amount of space that is typically used by conventional linear bar code symbols and allows for a large variety of rectangular bar code shapes.

APOSTAL (same as 31)
Specifies the bar code type as defined by the Australian Postal Service.

2DQRCODE (same as 32)
 Specifies a QR Code two-dimensional bar code. QR Code consists of a matrix of dark and light squares (modules). The matrix is also square and 40 sizes exist that range from a matrix of 21 by 21 modules and increase by steps of 4 up to a matrix of 177 by 177 modules. Thus, up to 7089 numeric characters, 4296 alphabetic
characters, 2953 8-bit characters, or 1817 Kanji characters can be contained on a single symbol, and up to 16 symbols can be logically linked together.

Since squares (modules) are square, the size of a module is determined by the MODWIDTH parameter only. The HEIGHT and RATIO parameters are not used.

**CODE93 (same as 33)**

Specifies a bar code type as defined by the AIM Uniform Symbology Specification - Code 93. This linear bar code is similar to Code 39, but more complex.

**US4STATE (same as 34 or US4ST)**

Specifies a United States Postal Service (USPS) Four-State bar code. This parameter can be abbreviated as US4ST. This bar code is also known as the Intelligent Mail Bar Code.

The USPS Four-State bar code symbol has a fixed size; therefore, the HEIGHT and RATIO parameters are not applicable and ignored. This bar code symbol allows a MODWIDTH parameter with two sizes SMALL and OPTIMAL. If you specify any other MODWIDTH, PPFA issues a warning message (RC=4), defaults to OPTIMAL, and continues generating the page definition. MODWIDTH SMALL prints a symbol approximately 2.575 inches wide and MODWIDTH OPTIMAL prints a symbol approximately 2.9 inches wide.

The input data is all numeric and consists of five data fields. The first four are fixed length; the fifth is variable length. The five fields are:

1. Application ID (2 digits) - the second digit must be 0 - 4 so that the valid values are 00 - 04, 01 - 14, and so on, through 90 - 94.
2. Special Services (3 digits) - assigned by the USPS. Valid values are 000 - 999.
3. Customer Identifier (6 digits) - assigned by the USPS. Valid values are 000000 - 999999.
4. Sequence Number (9 digits) - assigned by the mailer. Valid values are 000000000 - 999999999.
5. Delivery Point ZIP Code (0, 5, 9, or 11 digits) - see the following MOD parameter for valid values.

USPS Four-State modifiers (MOD) are defined as follows:

- **X'00'** Present a USPS Four-State bar code symbol with no Delivery Point ZIP Code. The input data for this bar code symbol must be 20 numeric digits.
- **X'01'** Present a USPS Four-State bar code symbol with a 5-digit Delivery Point ZIP Code. The input data for this bar code symbol must be 25 numeric digits. The valid values for the Delivery Point ZIP code are 00000 - 99999.
- **X'02'** Present a USPS Four-State bar code symbol with a 9-digit Delivery Point ZIP Code. The input data for this bar code symbol must be 29 numeric digits. The valid values for the Delivery Point ZIP code are 000000000 - 999999999.
- **X'03'** Present a USPS Four-State bar code symbol with an 11-digit Delivery Point ZIP Code. The input data for this
bar code symbol must be 31 numeric digits. The valid values for the Delivery Point ZIP code are 00000000000 - 99999999999.

**Note:** You can print HRI with this symbol but it is not currently (Oct 2004) defined by the USPS. Therefore, PPFA defaults the HRI parameter to HRI OFF for this symbol. The USPS says that they plan to define HRI for some Special Services. “Track and Confirm” is an example of a Special Service that USPS does not currently define HRI but might in the future.

**REDTAG (same as 35)**
Specifies a 4-state bar code type that is defined by the Royal Mail Postal Service of England as RED TAG.

**DATABAR (same as 36)**
Specifies a bar code type of GS1 DataBar.

**Other BARCODE Parameters:**

<table>
<thead>
<tr>
<th>Common BARCODE Parameters</th>
<th>Common 2D BARCODE Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenated BARCODE Parameters</td>
<td></td>
</tr>
</tbody>
</table>

**Common BARCODE Parameters**

<table>
<thead>
<tr>
<th>MOD — n</th>
<th>HRI — ON</th>
<th>SSASTERISK — ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVE</td>
<td>HRIFONT— fontname</td>
<td>ABOVE</td>
</tr>
<tr>
<td>BELOW</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>ONLY</td>
<td></td>
<td>ONLY</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WIDTH — n</th>
<th>HEIGHT — n</th>
<th>MODWIDTH — OPTIMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MM</td>
<td>OPTIMAL</td>
</tr>
<tr>
<td>CM</td>
<td>CM</td>
<td>SMALL</td>
</tr>
<tr>
<td>POINTS</td>
<td>POINTS</td>
<td></td>
</tr>
<tr>
<td>PELS</td>
<td>PELS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BCCOLOR — colorname</th>
<th>SUPPBLANKS</th>
<th>RATIO — n</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB— rvalue— gvalue— bvalue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGHLIGHT — hvalue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CMYK— cvalue— mvalue— yvalue— kvalue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIELAB— lvalue— (-) — cvluae— (-) — c2value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MOD

Specifies more processing information about the bar code symbol to be generated (for example, MOD specifies whether a check-digit \(^6\) must be generated for the bar code symbol).

\( n \) The meaning of \( n \) differs between the types. For more information, see Table 24 on page 486

If MOD is not specified, the MOD value defaults as follows, depending on the bar code type specified:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>MOD</th>
<th>TYPE</th>
<th>MOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>36</td>
<td>0</td>
</tr>
</tbody>
</table>

HRI

Specifies the human-readable interpretation (text characters) to be generated and placed above or below the bar code symbol, as directed.

**ON** Specifies that HRI must be generated at the default location for the bar code type.

**ABOVE** Specifies that HRI must be placed above the bar code symbol.

---

6. Check digits are a method of verifying data integrity during the bar code reading process.
**BELOW**
Specifies that HRI must be placed below the bar code symbol.

**OFF**
Specifies that HRI must not be generated.

**ONLY**
Specifies that only the HRI is to be printed. No bar code symbol is to be generated. The POSITION parameters on the FIELD command specify the placement position for the first character of the HRI.

**Note:** Not all bar code printers accept the request to suppress printing the bar code symbol.

**Notes:**
1. If HRI is requested, and HRI font isn’t, the printer default font is used to render the HRI, instead of the font specified on the FIELD FONT subcommand.
2. HRI is not supported by any of the 2D bar codes.

**HRIFONT** `fontname`
Specifies the local name of a font that is used in printing the HRI for the bar code. This font must first be defined in a previous font command in the page definition.

**SSASTERISK**

<table>
<thead>
<tr>
<th>SSASTERISK</th>
<th>ON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>

Specifies whether an asterisk is to be generated as the HRI for CODE39 bar code start and stop characters.

**Note:** SSASTERISK is ignored by all bar code types except CODE39.

**ON** Specifies that start and stop characters must be generated in the HRI.

**OFF** Specifies that start and stop characters must not be generated in the HRI.

**WIDTH**

<table>
<thead>
<tr>
<th>WIDTH—n</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
</tr>
<tr>
<td>MM</td>
</tr>
<tr>
<td>CM</td>
</tr>
<tr>
<td>POINTS</td>
</tr>
<tr>
<td>PELS</td>
</tr>
</tbody>
</table>

Specifies the width of the whole bar code symbol.

`n` Specifies the width value of the whole bar code symbol. The `n` value can be up to three decimal places.
**HEIGHT**

```
  HEIGHT n
  \ IN
  \ MM
  \ CM
  \ POINTS
  \ PELS
```

Specifies the height of bar code element. For UPC and EAN bar codes, the total height includes the bar code and the HRI characters.

If HEIGHT is not specified, the printer default height is used.

**Note:** HEIGHT is ignored by bar code types that explicitly specify the element heights (for example, POSTNET or RM4SCC).

$n$ Specifies the height of the bar code. The $n$ value can be up to three decimal places.

**unit**

Specifies a unit of measurement for the HEIGHT parameter. The choices are IN, MM, CM, POINTS, or PELS.

**Notes:**

1. If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

2. Height for the 2D bar code. PDF417 specifies the height of a bar or row (not the total height of the symbol).

**MODWIDTH**

```
  MODWIDTH OPTIMAL
  MODWIDTH n
  \ OPTIMAL
  \ SMALL
```

Specifies the width of the smallest defined bar code element by using mils (thousandths of an inch). The range of values is 1 - 254. If MODWIDTH is not specified, the printer default MODWIDTH is used; the printer default yields the optimum scannable symbol.

$n$ Specifies the width of each module by using thousandths of an inch (1/1000) as the unit of measurement.

**OPTIMAL**

Specifies that the printer chooses the optimal module width. This value is recommended. It is the default value when MODWIDTH is not coded.

**SMALL**

Specifies that the PPFA chooses a module width that produces the smallest symbol that meets the symbology tolerances.

**Note:** Because this symbol is at the lower boundary of the symbology-defined tolerance range, external conditions, such as
printer contrast setting, toner consistency, and paper absorbency, might cause this symbol to scan improperly.

Example:

```plaintext
PAGEDEF 4SXM1  REPLACE YES;
  FONT FN1;

  PRINTLINE ;
     FIELD START 1 LENGTH 20 BARCODE BC1 TYPE US4ST;

  PRINTLINE ;
     FIELD START 01 LENGTH 20 BARCODE bc2 TYPE US4STATE MOD 0
       MODWIDTH OPTIMAL;
  PRINTLINE ;
     FIELD START 41 LENGTH 25 BARCODE bc3 TYPE US4STATE MOD 1
       MODWIDTH SMALL ;
  PRINTLINE ;
     FIELD START 66 LENGTH 29 BARCODE bc4 TYPE US4STATE MOD 2
       MODWIDTH SMALL ;
  PRINTLINE ;
     FIELD START 66 LENGTH 31 BARCODE bc5 TYPE US4STATE MOD 3
       MODWIDTH SMALL ;
  PRINTLINE ;
     FIELD START 66 LENGTH 31 BARCODE bc6 TYPE US4STATE MOD 3
       MODWIDTH 15 ;
```

In the example:
- FIELD BARCODE commands exist for the new “Four State” barcode with default Modifier, and explicit Modifiers each with the proper field length.
- FIELD BARCODE commands exist that use the new MODWIDTH parameters SMALL and OPTIMAL.
- One example BARCODE command exists that uses an explicit MODWIDTH parameter that results in an informational message and a MODWIDTH of OPTIMAL.

**BCOLOR**

```
<table>
<thead>
<tr>
<th>BCOLOR</th>
<th>colormname</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>rvalue-gvalue-bvalue</td>
</tr>
<tr>
<td>HIGHLIGHT</td>
<td>hvalue</td>
</tr>
<tr>
<td>CMYK</td>
<td>cvalue-mvalue-yvalue-kvalue</td>
</tr>
<tr>
<td>CIELAB</td>
<td>lvalue-(e)cvalue-(e)cvalue-(e)c2value</td>
</tr>
</tbody>
</table>
```

Specifies an OCA color or a defined color to be used in printing the barcode and its HRI. Defined colors are specified with the DEFINE COLOR command.

**colorname**

Values for color names are:
- A defined color (defined by the DEFINE COLOR command)
- NONE
- DEFAULT
- BLACK
- BLUE
- BROWN
- GREEN
- PINK
• RED
• TURQ (turquoise)
• YELLOW
• ORANGE
• PURPLE
• MUSTARD
• GRAY
• DARKBLUE
• DARKGREEN
• DARKTURQ (dark turquoise)

The color choices depend on the printer. NONE is the color of the medium. DEFAULT is the printer default color.

color model

Specifies the color of print for this field that is supported in MODCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

Example: In the following code example, four bar codes are defined that use color.
1. The first uses a predefined non-OCA color.
2. The second uses an OCA color that is not predefined.
3. The third uses a predefined OCA color.
4. The fourth uses a CMYK color model directly.

/*--------------------------------------------------------*/
/* CMRX13 - Full Color on Bar Code */
/* */
/* */
/* */
/* */
/* Traditional pagedef */
/* */
*/*************************************************************/
Pagedef cmx14P replace yes;

DEFINE ocbblue COLOR OCA blue ;
DEFINE cmyk3yel COLOR CMYK 50 30 30 30 ;

FONT fte egt12 TYPE ebcdic;

PAGEFORMAT pf1;
PRINTLINE;
FIELD START 50 LENGTH 8 RGB 30 25 25
BARCODE AUST1 TYPE APOSTAL MOD 2
   BCCOLOR cmyk3yel /* PRE-DEFINED NON-OCA COLOR */
   HEIGHT .5 IN;
FIELD START 5 LENGTH 5 BARCODE BCCO TYPE POSTNET
   BCCOLOR RED /* OCA COLOR */
   HEIGHT .5 IN;
FIELD START 5 LENGTH 5 BARCODE BCCO1 TYPE POSTNET
   BCCOLOR ocbblue /* Defined OCA COLOR */
   HEIGHT .5 IN;
FIELD START 5 LENGTH 5 BARCODE BCCO2 TYPE POSTNET
   CMYK 50 30 30 30 /* direct cmyk color */
   HEIGHT .5 IN;

/*--------------------------------------------------------*/
/* Record Fmt pagedef */
/* */
*/*************************************************************/
Pagedef cmx14L replace yes;

DEFINE ocbblue COLOR OCA blue ;
Define rgbred COLOR RGB 30 25 25 ;
Suppress the trailing blanks in the data field that is used to generate the bar code.

When the page definition selects any of the EAN, UPC or POSTNET bar code types and modifiers and also requests that trailing blanks be
truncated for the bar code field, the print server examines the resulting data length and choose the correct bar code type and modifier for the bar code object created.

**Note:** If the data length does not match any of the bar code type and modifier combinations, the print server uses the original bar code type and modifier that is requested to build the bar code object.

**RATIO**

\[ \text{RATIO} \quad n \]

Specifies the ratio between the width of the wide and the narrow bar code elements. The range of values is 100 - 500, but you must specify a value appropriate for your printer and bar code type or you errors occur at print time.

If RATIO is not specified, the printer default ratio is used.

\[ n \] The RATIO is specified as a percent value. For example, form \( nnn \).

For example, 200 represents a ratio of 2 to 1; 250 represents a ratio of 2.5 to 1. For most bar code symbols, the RATIO value is 200 - 300. For bar code types that explicitly specify the module width (for example, POSTNET and RM4SCC, this field is ignored. If RATIO is not specified, the default ratio for the bar code symbol is used.

**CMR**

\[ \text{CMR} \quad \text{cmr-lname} \]

Specify a Color management resource (CMR) and its process mode for a bar code object within the page definition.

**Note:** See [Chapter 8, “AFP Color Management,” on page 163](#) for more information about using the CMR subcommand.

\[ \text{cmr-lname} \] The CMR local name. This name must be defined with a DEFINE CMRNAME command.

**Note:** This parameter must immediately follow the CMR keyword.

**processing mode parameter**

Specify the processing mode for the CMR.

**AUDIT**

Process this CMR as an audit CMR.

**INSTR**

Process this CMR as an instruction CMR.
LINK

Process this CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs.

Example: In the following code example, 2 bar codes are defined with CMRs specified. The bar codes are defined for traditional, record format and XML page definitions.

Note: The DEFINE CMRNAMEs for mycmr and dark1 are used in each page definition but defined only once. Page definitions that are compiled together can define a local CMR name only once. This requirement is because a DEFINE CMRNAME definition is global for all page definitions and form definitions in the same set of source code.

```
DEFINE mycmr CMRNAME ... ;
DEFINE dark1 CMRNAME ... ;
```

/* Traditional Pagedef */
PAGEDEF cmr10P REPLACE yes;
   PRINTLINE;
   FIELD Start 1 Length 20
   BARCODE TYPE code39 MOD 1
       CMR myCMR audit;
   FIELD Start 21 Length 40
   BARCODE TYPE code39 MOD 1
       CMR dark1 instr;

/* Record Layout Pagedef */
PAGEDEF cmr10L REPLACE yes;
   Font f1;
   LAYOUT 'II';
   FIELD Start 1 Length 20
       BARCODE TYPE code39 MOD 1
       CMR myCMR audit;
   FIELD Start 21 Length 40
       BARCODE TYPE code39 MOD 1
       CMR dark1 instr;

/* XML Pagedef */
PAGEDEF cmr10X REPLACE yes;
   Font f1 TYPE ebc_dic;
   XLAYOUT QTAG 'x1';
   FIELD Start 1 Length 20
       BARCODE TYPE code39 MOD 1
       CMR myCMR audit;
   FIELD Start 21 Length 40
       BARCODE TYPE code39 MOD 1
       CMR dark1 instr;
```

Common 2D BARCODE Parameters
Bar code parameters are common for all for two-dimensional bar code types.

**Note:** See the Bar Code Object Content Architecture (BCOCA) Reference and Appendix D, “More About Bar Code Parameters,” on page 477 for more details on these extra parameters.

**escape sequence processing**
Specifies whether to process escape sequences in the data.

**Note:** If the EBCDIC to ASCII flag is set (E2A), all characters are converted ASCII first so that the EBCDIC backslash characters (X'E0') are converted to ASCII (X'5C') before the escape sequence handling is applied.

**ESC**
Escape Sequence Handling. This is the default if neither is coded. When this parameter is coded or defaulted, each backslash character within the bar code data is treated as an escape character according to the particular bar code symbology specification.

**NOESC**
Ignore Escape Sequences. When this parameter is coded, each backslash character within the bar code data is treated as a normal data character.

**Note:** In this case, no code page switching can occur within the data.

**EBCDIC to ASCII translation**
Determine whether to translate the data.

**Note:** Only QRCODE uses the E2A code page parameters.

**E2A**
EBCDIC to ASCII translation for all two-dimensional bar codes.
- For Data Matrix and MaxiCode, the printer converts each byte of the data from EBCDIC code page 500 to ASCII code page 819.
- For PDF417, the printer converts each byte of the bar code data and each byte of the Macro PDF417 control block data from a subset of EBCDIC code page 500 into ASCII. This translation covers 181 code points, which includes alphanumerics and many symbols. The code points that are not covered by the translation do not occur in EBCDIC and are mapped, by the printer, to the X'7F' (127) code point. Do not use the following EBCDIC code points for PDF417:
QR CODE - The default coding for QR CODE is ECI 000020, which is equivalent to the IBM ASCII code page 897. When translation is required, you must enter the code page to use for translation. Three choices exist. Each choice causes the printer to translate from the code page into ASCII code page 897 before the data is used to build the bar code symbol:

- EBCDIC code page 500 (International #5). Only 128 bytes of this code page can be translated into ECI 000020. These code points are specified in "QR Code Special-Function Parameters" on page 331.
- EBCDIC code page 290 (Japanese Katakana Extended).
- EBCDIC code page 1027 (Japanese Latin Extended).

The first three values are used when the input data is encoded with a single-byte EBCDIC code page. The parameter identifies the EBCDIC code page that encodes single-byte EBCDIC bar code data:

**CP500**
Code page 500 (International #5) Only 128 of the characters within ECI 000020 can be specified in code page 500. The code page 500 characters that can be translated are shown in the Bar Code Object Content Architecture Reference.

**CP290**
Code page 290 (Japanese Katakana Extended)

**CP1027**
Code page 1027 (Japanese Latin Extended)

The following parameters are used when the input data is SOSI. Each parameter identifies a specific conversion from EBCDIC SOSI input data to a specific mixed-byte ASCII encoding:

**CV1390To943**
Translates EBCDIC data CCSID 1390 code points to ASCII Shift-JIS CCSID 943 code points.

**CV1399To943**
Translates EBCDIC data CCSID 1399 code points to ASCII Shift-JIS CCSID 943 code points.

**CV1390To932**
Translates EBCDIC data CCSID 1390 code points to ASCII Shift-JIS CCSID 932 code points.

**CV1399To932**
Translates EBCDIC data CCSID 1399 code points to ASCII Shift-JIS CCSID 932 code points.
CV1390To942
Translates EBCDIC data CCSID 1390 code points to ASCII Shift-JIS CCSID 942 code points.

CV1399To942
Translates EBCDIC data CCSID 1399 code points to ASCII Shift-JIS CCSID 942 code points.

Note: CCSID definitions:

**CCSID 932**
Japanese PC Data Mixed including 1880 UDC.

**CCSID 942**
Japanese PC Data Mixed including 1880 UDC, Extended SBCS.

**CCSID 943**
Japanese PC Data Mixed for Open environment (Multi-vendor code): 6878 JIS X 0208-1990 chars, 386 IBM selected DBCS chars, 1880 UDC (X'F040' to X'F9FC')

**CCSID 1390**
Extended Japanese Katakana-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS and DBCS euro)

**CCSID 1399**
Extended Japanese Latin-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS and DBCS euro)

**NOE2A**
No translation. This is the default if neither is coded. This parameter is used for all two-dimensional bar codes. No translation is done by the printer or PPFA. The bar code data is assumed to be the default coding as defined in the AIM Uniform Symbology Specification.

**Data Matrix 2D parameters**

These parameters are for Data Matrix 2D bar codes:
**SIZE**

The size of the two-dimensional bar code. The number of rows and columns (row size) in the symbol. The allowable values for rows and columns are specified in Table 28 on page 519. If size is not coded, the size is marked as unspecified and the appropriate number of rows and columns are used based on the amount of data.

- **unspecified**
  - Unspecified size. The SIZE parameter isn’t coded. The appropriate number of rows and columns are used based on the amount of data that is presented.

- **rows BY cols**
  - The number of rows that are wanted including the finder pattern, and the number of columns (or modules) in each row that are wanted including the finder pattern. The keyword BY is optional. The rows and columns must be one of the allowed combinations in Table 28 on page 519.

**SEQUENCE**

Structured append sequence indicator. Some two-dimensional bar codes can be logically linked together to encode large amounts of data. The logically linked symbols can be presented on the same or different media and are logically recombined after they are scanned. PPFA checks the numbers for obvious errors and the proper number range. For example, SEQUENCE 5 OF 3 is wrong.

- **sqn**
  - Structured-append sequence indicator. This parameter is an integer whose acceptable range of values depends on the bar code type. The range for this parameter is 1 - 16.

- **OF**
  - Optional parameter for readability.

- **tot**
  - Total number of structured-append symbols. This parameter is an integer whose acceptable range of values depends on the bar code type. The range of this parameter is 2 - 16.

- **ID uidHi uidLo**
  - The high and low-order bytes of a unique file identification for a set of structured-append symbols. Each is a unique number 1 - 254 and identifies this set of symbols. The actual File ID is computed by 256 times uidHi plus uidLo.
Special functions that can be used only with a Data Matrix symbol. If not coded, the default is USERDEF (user-defined symbol).

**FNC1UCC**

UCC/EAN FNC1 alternate data type identifier. An FNC1 is added in the first data position (or fifth position of a structured append symbol) to indicate that this bar code symbol conforms to the UCC/EAN application identifier standard format.

**FNC1IND**

Industry FNC1 alternate data type identifier. An FNC1 is added in the second data position (or sixth data position of a structured append symbol) to indicate that this bar code symbol conforms to a particular industry standard format.

**RDRPROG**

Used when the symbol contains a message that is used to program the bar code reader. In this case, the bar code symbol cannot be a part of a structured append sequence.

**MAC5**

This provides instructions to the bar code reader to insert an industry-specific header and trailer around the symbol data. The bar code symbol contains a 05 Macro codeword. The bar code symbol cannot be a part of a structured append sequence.

**MAC6**

Same as MAC5 except the bar code symbol contains a 06 Macro codeword. The bar code symbol cannot be a part of a structured append sequence.

**ENCODE**

Data Matrix bar code encodation scheme.

*type*

Specifies the Data Matrix bar code encodation scheme. These encodation schemes are supported:

**DEFAULT**

Uses a device-specific method of selecting and switching among encodation schemes. If you are unsure of which encodation scheme to use, device default is a good choice.

**ASCII**

Encodation scheme that produces 4 bits per data character for double digit numerics, 8 bits per data character for ASCII values 0 - 127, and 16 bits per data character for Extended ASCII values 128 - 255.

**C40**

Encodation scheme that is used when the input data is primarily uppercase alphanumeric.
Text
Encodation scheme that is used when the input data is primarily lowercase alphanumeric.

X12
Encodation scheme that is used when the input data is defined with the ANSI X12 EDI data set.

EDIFACT
Encodation scheme that is used when the input data is ASCII values 32 - 94.

BASE256
Encodation scheme that is used when the data is binary (for example image or non-text data).

AUTO
Starts with ASCII encodation and switches between encodation schemes as needed to produce the minimum symbol data characters.

USERDEF
None of the supported schemes. This is a user-defined data symbol with no Header or Trailer instructions to the reader.

MaxiCode 2D Parameters

These parameters are for MaxiCode 2D bar codes:

MODE

Symbol mode (used for MaxiCode two-dimensional bar code only). If not coded, the default is Standard Symbol Mode 4.

2 Structured Carrier Message - numeric postal code.
3 Structured Carrier Message - alphanumeric postal code.
4 Standard symbol (default).
5 Not supported.
6 The bar code data is used to program the bar code reader system.

SEQUENCE
Structured append sequence indicator. Some two-dimensional bar codes can be logically linked together to encode large amounts of data. The logically linked symbols can be presented on the same or different media and are logically recombiner after they are scanned. PPFA checks the numbers for obvious errors and the proper number range. For example, SEQUENCE 5 OF 3 is wrong.

\[sqn\]
Structured-append sequence indicator. This parameter is an integer whose acceptable range of values depends on the bar code type. The range of this parameter is 1 - 8.

OF Optional parameter for readability.

tot Total number of structured-append symbols. This parameter is an integer whose acceptable range of values depends on the bar code type. The range for this parameter is 2 - 8.

\[zipper\]
Print a zipper pattern and contrast block (use for MaxiCode two-dimensional bar code only).

\[NOZIPPER\]
Does not print a zipper pattern.

\[ZIPPER\]
Prints a zipper pattern.

\[PDF417 2D Parameters\]

These parameters are for PDF417 2D bar codes:

\[SIZE\]
The size of the two-dimensional bar code. The number of rows and number of columns (number of data symbol characters per row). These numbers do not include the start patterns or left and right row indicators. The allowable values for rows are 3 - 90, the
allowable values for cols are 1 - 30, but their product cannot exceed 928. If size is not coded, the default is MIN number of rows and 10 cols (characters per row).

rows
The number of rows that are wanted.

MIN
Instructs the printer to use the minimum number of rows necessary to print the symbol.

cols
The number of data symbol characters in a row that are wanted.

SECLEV

SECLEV-0
SECLEV-st

This parameter specifies the security level that is wanted for the symbol as a value 0 - 8. Each higher security level causes more error correction codewords to be added to the symbol (used for PDF417 two-dimensional bar code only). If not coded, the default is Security level 0.

MACRO

qstring

PDF417 Macro data. The total length of macro text is limited to 28,000 bytes.7

qstring
A quoted string. The string does not extend across records, but you can code multiple quoted strings. Code the MACRO keyword only once.

QRcode 2D Parameters

SIZE-MIN
SIZE-rows-MIN
SEQUENCE-seq-tot-OF-PARITY-X'dd'

7. This limit is imposed by the data stream architecture. The total number of bytes allowed in a structured field is 32,000. Macro data must be shared with triplets, bar code data (which can be up to 2710 bytes), and other overhead.
These parameters are for QRCODE 2D bar codes:

**SIZE**

The size (in rows and columns) that is wanted for the QRCODE bar code. This symbol is square so both rows and columns are the same. The allowable values for rows and columns are 21 - 177 in increments of 4. These are also specified in Table 30 on page 536. The size in rows by cols is 21 - 177 in increments of 4.

**rows**

The number of rows and columns that is wanted.

**MIN**

Instructs the printer to use the minimum number of rows necessary to print the symbol.

**SEQUENCE**

QR bar codes can be logically linked together to encode large amounts of data. The logically linked symbols can be presented on the same or different media and are logically recombined after they are scanned. PPFA checks the numbers for obvious errors and the proper number range. For example, SEQUENCE 5 OF 3 is wrong.

**seq**

Structured-append sequence indicator. This parameter is an integer whose acceptable range of values is 1 - 16.

**OF**

Optional parameter for readability.

**tot**

Total number of structured-append sequences indicator. This parameter is an integer whose acceptable range of values is 2 - 16.

**PARITY X'dd'*

Structured append parity data. This parameter is used for the QR Code 2D bar code only when it has linked structured-append symbols. The parameter specifies the parity byte for the entire collection of linked structured-append symbols. The parity byte is the same for each symbol in the collection and is obtained by doing an “exclusive or” function.
on all of the bytes of the ASCII data in all symbols of the collection. If this symbol is not structured-append symbol, the parity parameter is ignored.

\[ X'\text{dd}' \]

The parity data byte. It must be entered as two hexadecimal digits (\(X'0' - X'F'\)). As for all hexadecimal digits in PPF, the digits must be uppercase if they are \(X'\text{A}' - X'\text{F}'\).

**ECLEV**

![ECLEV Diagram]

Error Correction Level. It specifies the level of error correction to be used for the symbol. Each higher level of error correction causes more error correction code words to be added to the symbol; therefore, fewer code words are left for the data. See the particular bar code symbology specification for more information. Four different levels of Reed-Solomon error correction can be selected:

- **L** Level L allows recovery of 7% of symbol code words.
- **M** Level M allows recovery of 15% of symbol code words.
- **Q** Level Q allows recovery of 25% of symbol code words.
- **H** Level H allows recovery of 30% of symbol code words.

### special functions

![Special Functions Diagram]

Special functions that can be used with QR Code 2D symbols. If not coded, the default is USERDEF (user-defined symbol).

**FNC1UCC**

UCC/EAN FNC1 alternate data type identifier. The symbol indicates that this QR Code symbol conforms to the UCC/EAN application identifiers standard.

**FNC1IND**

Industry FNC1 alternate data type identifier. The symbol indicates that this QR Code symbol conforms to the specific industry or application specifications previously agreed with AIM International. When this standard is selected, an application indicator must be specified.

**AI 'ai'**

Application indicator for Industry FNC1. This value is coded as a single uppercase or lowercase alphabetic character, or a 2-digit number. It must be enclosed in
single quotation marks. This parameter is required for QR Code bar codes when FNC1IND is coded.

**USERDEF**

None of the special functions. This value is a user-defined symbol with either no significance or “user-defined” significance that is assigned to all FNC1 characters that appear in the symbol.

**QRCODE Barcode Example:**

```
PAGENDEF QNXmp Replace Yes;

PRINTLINE;
FIELD START 1 LENGTH 4400 BARCODE bc3p TYPE 2DQRCODE
BCXPARMS E2A CP500
    noesc
    SIZE 025
    ECLEV M
    SEQUENCE 1 of 7
    PARITY x'7A'
    FNC1IND AI 'a'

```

In the example:

- A QRCODE 2D bar code is placed. The data is encoded in EBCDIC with code page 500. The bar code is 25 by 25 squares and the error correction level is M, which allows recovery of 15% of symbol code words.
- This is the first of seven symbols that are to be linked together by the bar code reader application program. The parity-data value for all symbols is a hexadecimal X'7A'. Parity must be the same for all the linked symbols and is obtained by doing an “exclusive or” function on all of the bytes of the ASCII value of all the original input data.
- The QRCODE symbol conforms to industry specifications for application indicator a.

**Concatenated BARCODE Parameters**

```
<table>
<thead>
<tr>
<th>BCDSYM</th>
<th>symname</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCDSEQ</td>
<td>seq#</td>
</tr>
<tr>
<td>BCDNEW</td>
<td></td>
</tr>
</tbody>
</table>
```

**BCDSYM symname**

Specifies bar code data collection symbol and names a bar code data collection. All bar code data that is described with FIELD command and bar codes with this name is collected for printing the bar code.

Defines an alpha-numeric name up to 16 characters long. This name must be unique within a page format.

**BCDSEQ seq#**

Specifies bar code data sequence number. Allows for sequencing bar code data. Bar code data that is collected with unique sequence numbers is collected in ascending order of its sequence number. Data with the same sequence number is collected in the order its FIELD commands are processed.
Note: BCDSEQ is optional but if it coded, all bar codes with the same BCDSYMB symbol name must code BCDSEQ.

**BCDNEW**

Specifies to start a new bar code symbol for collected bar code data when this record is reused. If BCDSEQ is used to sequence the collection data, BCDNEW is placed on the FIELD command for the first record that is encountered in the data. This record might not be the first sequentially. In general, BCDNEW is placed on the record for the first piece of data that is encountered in the bar code data collection. If BCDSYMB is not specified, BCDNEW is ignored.

Note: This parameter has no effect on a PRINTLINE FIELD BARCODE command as the BCDNEW function does not apply to traditional line data.

**FONT Command**

**Traditional and Record Format**

![Diagram of FONT command]
The FONT command is used to identify the fonts that are to be specified with the following commands:

**Traditional**
- PRINTLINE, FIELD, and TRCREF commands.

**Record Layout**
- LAYOUT and FIELD commands.

**XML**
- XLAYOUT and FIELD commands.

A maximum of 127 font names for each page definition can be identified.

**Note:** Naming a font with the FONT command does not, by itself, affect your output. You must specify the font in one of the listed commands for the font to become effective. You must name at least one font in a Record Format or XML page definition.

FONT commands immediately follow the PAGEDEF command. A separate FONT command is required:
- For each font used within a page definition
- For each rotation of the same font

**Note:** For Traditional, see the "TRCREF Command (Traditional)' on page 445 for the exception.

### FONT

```plaintext
FONT

<table>
<thead>
<tr>
<th>lname</th>
<th>cfname</th>
</tr>
</thead>
</table>
```

Identifies the fonts to be specified in the preceding commands.

**lname**

Local name for the font. Specifies an unquoted alphanumeric name of 1 - 16 characters (local name) of the font to be used in the page definition. The name must conform to the token rules and must be unique within the page definition.

**lname** is used with the following commands:

**Traditional**
- PRINTLINE, FIELD, and TRCREF commands.

**Record Layout**
- LAYOUT and FIELD commands.

**XML**
- XLAYOUT and FIELD commands.

** lname** is optional if **cfname** is specified.

**cfname**

Coded font name. Specifies an alphanumeric name of 1 - 6 characters (user-access name) of the coded font to be used in the page definition. Specify this name without the Xn prefix.

**'cfname'**

Quoted full user-access name. Specifies a quoted alphanumeric name of 1 - 8 characters of the coded font to be used in the page definition. The name
can contain blanks and special characters. No uppercase folding or prefix is added to the name. The 'cfname' variable is intended for outline fonts and allows them to be selected without overriding the HEIGHT specified in the CFI structured field in the coded font. Enter the full outline font name as a quoted name and do not enter the HEIGHT parameter. For example, if you enter the following command, the outline font XZM32F is used with no overriding HEIGHT parameters:

```
FONT myfont 'XZM32F'
```

Notes:

1. The quoted name of the font name is primarily intended for outline fonts. If you use a quoted name for a raster font, you must be sure that you have the name corresponding to the correct rotation of the font.
2. If you use the quoted name of the font name, you must also enter an INAME (local name); sometimes called an alias name.
3. You can still specify the HEIGHT command if you want and override the coded font height.

### Subcommands

**SBCS or DBCS**

- **SBCS**
- **DBCS**

Specifies single-byte or double-byte fonts.

#### SBCS

Specifies that the font is a single-byte character set. This value is the default.

#### DBCS

Specifies that the font is a double-byte character set.

**HEIGHT n**

```
HEIGHT n
```

Specifies the height of the outline font.
POINTS
Each point is equal to 1/72 of an inch.

IN Inches
CM Centimeters
MM Millimeters

PELS
Pels in the current Logical Units per inch. For example, in 240ths of an inch.

TYPE (Traditional and Record Format)

The TYPE subcommand indicates the type of font that is used.

EBCDIC
This parameter is normally used for fonts on OS390–based systems. This value is the default.

ASCII
This parameter is normally used for fonts on workstation–based systems.

UNICODE
This parameter is used with Unicode type fonts.

TYPE (XML only)

The TYPE subcommand indicates the type of font that is used. This parameter is required for fonts in an XML page definition.

EBCDIC
This parameter is normally used for fonts on OS390–based systems.

ASCII
This parameter is normally used for fonts on workstation–based systems.

UNICODE
This parameter is used with Unicode type fonts (fixed 2-byte UNICODE without surrogates).

Note: TYPE indicates what type of font is being used, OS390 or workstation, for the printing of PRINTLINE, LAYOUT or XLAYOUT commands. UDTYPE (on the PAGEDEF) is the user’s data type (EBCDIC, ASCII, UTF8, UTF16) that is being placed with the font. The font TYPE and user data UDTYPE must match but certain combinations of TYPE and UDTYPE are permitted.
• Data is UTF-8 and Font is ASCII or UNICODE
• Data is UTF-16 and Font is UNICODE
Otherwise, message AKQ271E results.
RATIO—percent

Specifies the ratio of scaling the width relative to the height in an outline font.

percent

Represents the percent of the normal width of the character that is printed. For example, specifying RATIO 50 yields a font with characters half as wide as normal, and specifying RATIO 200 yields a font with characters twice as wide (200% as wide) as normal. If RATIO is specified, you must also specify the HEIGHT.

ROTATION

0

90

180

270

Specifies the rotation of characters in degrees. The specified value is relative to the inline direction of a printline or field. Valid rotations are 0°, 90°, 180°, or 270°; 0° is the default.

RESOLUTION

RES or RESOLUTION

The raster-pattern resolution units in pels per inch:

240  240 pels per inch
300  300 pels per inch

METTECH or METRICTECHNOLOGY

The metric technology that is used for this raster font:

| FIXED     | Fixed-metric technology |
| RELATIVE  | Relative-metric technology |

Notes:

1. The resolution and metric technology subcommands allow rigorous font specifications for use with font fidelity. See the font fidelity subcommand FONTFID on the FORMDEF command.
2. For a description of metric technologies, see:
   - Intelligent Printer Data Stream Reference
   - Font Object Content Architecture Reference
3. RESOLUTION can be abbreviated as RES; METRICTECHNOLOGY can be abbreviated as METTECH.

In the following example, the form definition xmp01 specifies font fidelity and the page definition specifies a font that has 240 pels per inch resolution and
fixed-metric technology. If a font with exactly those characteristics is not accessible by the printer, an error occurs and processing stops.

```
FORMDEF xmp01
  FONTFID YES ;

PAGEDEF xmp01 replace yes ;
  FONT xx2 res 240 mettech fixed ;
  PRINTLINE font xx2 ;
```

**Chapter 11. Page Definition Command Reference**
The LAYOUT command is used to format a data record. The LAYOUT command is associated with the line of data record by using a record ID that appears both on the LAYOUT command and in the first $n$ bytes of the data record (normally $n$ is 10 bytes, but can be specified as 1 - 250 bytes). The 'record ID' is entered in quotation marks and must match the 'record ID' within the data exactly, byte for byte. The ID is padded with blanks if the field is entered with less than $n$ bytes.

The LAYOUT command is used in a different type of page definition, a Record Format page definition. The LAYOUT command is analogous to the PRINTLINE and XLAYOUT commands in Traditional page definitions and XML page definitions.

Notes:
1. The LAYOUT command defines a Record Format page definition and cannot be mixed with PRINTLINE commands, which define Traditional page definitions, or XLAYOUT commands, which define XML page definitions.
2. Typically the record ID is 10 bytes long; however, it can be specified as 1 - 250 bytes by using the RECIDLEN subcommand on either the PAGEDEF or PAGEFORMAT commands.
Subcommands

DEFAULT

This ‘record ID’ is used only when the layout type is either PAGEHEADER or PAGETRAILER and no name is needed.

‘record ID’
The ‘record ID’ is a quoted name up to 250 characters long that is accepted as is with no case folding or translation.

C ‘record ID’
The C ‘record ID’ is a quoted name with a C for Character that is treated the same as a quoted name up to 250 characters. No folding or translation is done.

A ‘record ID’
The A ‘record ID’ is a quoted name with an A for ASCII entered with up to 250 single-byte characters that are accepted as-is if on an ASCII operating system or translated to ASCII if on an EBCDIC operating system. The translation is made with no case folding.

E ‘record ID’
The E ‘record ID’ is a quoted name with an E for EBCDIC entered with up to 250 single-byte characters that are accepted as-is if on an EBCDIC operating system or translated to EBCDIC if on an ASCII operating system. The translation is made with no case folding.

X ‘hhhh’
The X ‘hhhh’ is a quoted name with an X for Hexadecimal entered with up to 500 hexadecimal characters. The characters are translated to hexadecimal, but no assumption of data type is made.

Note: All X’00’ and X’FF’ values are considered reserved and must not be used to specify the hexadecimal name value.

U8 ‘record ID’
The U8 ‘record ID’ is a quoted name with a U8 for UTF-8 entered with up to 250 single-byte characters that are translated to UTF-8.

U16 ‘record ID’
The U16 ‘record ID’ is a quoted name with a U16 for UTF-16 entered with up to 125 single-byte characters that are translated to UTF-16.
The BODY layout type is used for most of data in the user's database, normally printed line by line. This is the default.

**GROUP**

The GROUP parameter indicates that the existing group header must be saved and used for subsequent pages. If this parameter is not set when processing starts on a BODY layout, the active group header record is discarded and not reprinted on subsequent pages.

**PAGEHEADER**

This layout type specifies a header that is to be printed on each new page. The baseline position of this layout is normally in the top margin, but can be anywhere on a logical page. If RELATIVE is specified, the position is considered to be relative to the page origin. Usually contains customer's name, address, account number, and so forth. Only one default PAGEHEADER layout can be specified in a PAGEFORMAT and no input record data can be specified in a default layout.

**GRPHEADER**

This layout type specifies a header that is to be printed at the beginning of a group of data. If a logical page eject occurs before the group of data ends, the header is printed after the top margin on each new page until the group ends. The baseline position of this layout can be specified as RELATIVE. It can include column headings.

**XSPACE**

XSPACE indicates the amount of extra space from the position of the layout to the bottom of the group header area. This allows the user to identify the amount of extra space in excess of one text line that is used by the header so that the baseline moves down and the following group data is not placed on top of the header area. This space is not calculated by PPFA and must be explicitly defined by the user. See Figure 133 on page 369 (shaded space shows group header area):
When a Group Header record is processed and is still active when it leaves the PAGEFORMAT, the group header record is saved by the presentation services program. Whenever the same PAGEFORMAT is reinvoked, the saved group header record is presented again if the first body record after reinvoking the PAGEFORMAT selects a Body record that has the Group Indicator on.

**PAGETRAILER**
This layout type specifies a trailer that is to be printed on each new page. The baseline position of this layout is normally in the bottom margin, but can be located anywhere on a logical page and can be specified as RELATIVE. Only one default PAGETRAILER layout can be specified in a PAGEFORMAT and no input record data is processed with a default layout. It can contain the name of the form or a footnote.

**NEWPAGE**
This parameter indicates that a new page must be started with this layout name. If it is a header or trailer layout, the print position is moved to the start of a new page before this header or trailer becomes the active header or trailer.

**DELIMITER**
The delimiter is a one or two-byte code that is specified in either character or hex indicates a delimiting character within the customer’s database and is used to separate fields. PPFA translates the character data to the data type specified by the UDType subcommand on the PAGEDEF command. Hex characters must be entered in uppercase within the quotation marks and are not translated.

**Notes:**
1. Delimiters that are specified after the Record ID are ignored.
2. You cannot mix delimited and non-delimited fields on the same LAYOUT command.
3. A single-byte delimiter character is ignored when EBCDIC double byte text (SOSI) is processed.

**PRINTDATA**
Specifies whether the line of data that is associated with the current LAYOUT must be printed. The PRINTDATA subcommand is useful when the data stream is interspersed with lines of comments, blank lines, or lines without data that is not meant to be printed.

**YES**
- Specifies that the data for the current LAYOUT is printed. YES is the default.

**NO**
- Specifies that the data for the current LAYOUT is not printed.

**POSITION**

Used in positioning FIELD, DRAWGRAPHIC, and ENDGRAPHIC text and graphics. If RELATIVE is specified or POSITION is not specified, the baseline of the POSITION is relative to the previous LAYOUT position.

1. For PAGEHEADER RCD: The baseline position can be anywhere on a logical page, but cannot be specified as Relative.
2. For PAGETRAILER, GROUPHEADER and BODY RCDs: The baseline position can be anywhere on a logical page and can be specified as RELATIVE.

Specifies the starting position of the layout in the printout.

**horizontal position**

**x-pos**
- Specifies the horizontal offset from the left side of the logical page. The value is a number with up to three decimal places. The valid options for x-pos are described in the SETUNITS command for the horizontal value.

**LEFTMARGIN**
- Specifies that this line starts at the position that is specified as the horizontal (x) value in the previous LEFTMARGIN subcommand within the page definition.

**SAME**
- Specifies this line starts at the same horizontal offset position as the previously coded LAYOUT. If applied to the first LAYOUT of a logical page, the horizontal position is 0, which is the default.

= Alternative for SAME.
RELATIVE
Specifies that the following vertical position value is to be processed as a relative value. The LAYOUT is positioned relative to the last LAYOUT placed on the page.

Notes:
1. If both TOP and RELATIVE are requested for the y-pos value, the RELATIVE request is ignored.
2. When PPFA uses RELATIVE positioning, it does not flag off-the-page conditions for the position of a LAYOUT or for any overlays, segments, or objects placed relative to that LAYOUT. LAYOUTs that fall outside the bounds of the logical page are flagged by the print server at run time.
3. When you are specifying RELATIVE, use the minus sign to indicate any negative values for the LAYOUT vertical position; you can use the plus sign to indicate positive values. If no sign is used, a positive value is assumed.
4. The DIRECTION for a relative LAYOUT must be ACROSS. Fields that are associated with a relative LAYOUT must have the same DIRECTION as the LAYOUT and must match the PAGEFORMAT DIRECTION.
5. If RELATIVE is specified with SAME or = as the y-pos value, the relative value in the LAYOUT is +0.
6. RELATIVE positioning is allowed on a LAYOUT command only if the LAYOUT and all its associated FIELD commands are formatted to print in the same direction as the PAGEFORMAT. That is, the DIRECTION parameter in the LAYOUT and any associated FIELD commands must specify (or default to) ACROSS. The DIRECTION in the PAGEFORMAT or PAGEDEF command is any allowable value: ACROSS, DOWN, BACK, or UP.

vertical position

y-pos
Specifies the vertical offset from the top side of the logical page. The value options for y-pos are described in the SETUNITS command for the vertical value.

TOPMARGIN
Specifies that the LAYOUT is placed in the position that is specified as the vertical (y) value in the TOPMARGIN subcommand within the page definition.

NEXT
Specifies the layout is to be positioned down (on the logical page) one line (as defined in the LINESP subcommand of the last SETUNITS command) from the previews LAYOUT. The LINESP subcommand of the SETUNITS command establishes the distance from one line to the next.

When NEXT is specified for the first LAYOUT of a logical page, the starting position of the line is one line down from the top of the logical page, as defined by the TOPMARGIN subcommand.

Note: The down direction is determined by the direction of the logical page (as specified in the page format), not the LAYOUT direction. Therefore, NEXT is mainly useful in ACROSS LAYOUTs.
SAME
Specifies this LAYOUT starts at the same vertical position as the previous LAYOUT.
= Alternative for SAME.

DIRECTION

ACROSS
DOWN
BACK
UP

Specifies the print direction of the line relative to the upper-left corner as you view the logical page. Not all printers can print in all print directions. For more information about your printer, see your printer documentation.

If DIRECTION is not specified, the direction that is specified in the PAGEFORMAT command is used. Observe that this direction is additive to the direction specified in the PAGEFORMAT command. See “PAGEFORMAT Command” on page 415.

ACROSS
The layout direction is rotated 0 degrees relative to the direction specified in the PAGEFORMAT (the layouts are oriented in the same direction as the page).

DOWN
The layout direction is rotated 90 degrees relative to the direction specified in the PAGEFORMAT.

BACK
The layout direction is rotated 180 degrees relative to the direction specified in the PAGEFORMAT.

UP
The layout direction is rotated 270 degrees relative to the direction specified in the PAGEFORMAT.

ENDSPACE

IN
MM
CM
POINTS
PELS

If the remaining body space is less than the value specified, ENDSPACE causes a logical page eject to be done. This can be used, for example, on a GRPHEADER layout to ensure that a group header does not print at the end of a page without the first data record of the group. ENDSPACE does not include the space within the bottom margin (specified on the PAGEDEF or PAGEFORMAT command). This indicator is ignored on a PAGEHEADER or PAGETRAILER layout.

COLOR
Specifies an OCA or defined color for the text of this field. This subcommand is recognized only by printers that support multiple-color printing. See your printer publication for information about the colors that can printed.

**colorname**

Values for colorname can be a defined color (see \textit{DEFINE COLOR Command} on page 280), or an OCA colorname. Values for OCA colornames are:

- NONE
- DEFAULT
- BLACK
- BLUE
- BROWN
- GREEN
- RED
- PINK (or MAGENTA)
- TURQ (or CYAN)
- YELLOW
- DARKBLUE (or DBLUE)
- ORANGE
- PURPLE
- MUSTARD
- GRAY
- DARKGREEN (or DGREEN)
- DARKTURQ (DTURQ, or DCYAN, or DARKCYAN)

The color choices depend on the printer.

If you do not enter one of these colors, the default color for that printer is used. NONE is the color of the medium; DEFAULT is the printer default color.

**Note:** In some printer manuals, the color turquoise (TURQ) is called \textit{cyan}, and the color pink (PINK) is called \textit{magenta}.

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

**color models**

Specifies the color of print for this field that is supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.
RGB \textit{rvalue gvalue bvalue}

Three RGB integer values are used. The first (\textit{rvalue}) represents a value for red, the second (\textit{gvalue}) represents a value for green, and the third (\textit{bvalue}) represents a value for blue. Each of the three integer values can be specified as a percentage 0 - 100.

\textbf{Note:} An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

**HIGHLIGHT \textit{hvalue COVERAGE cvalue BLACK bvalue}**

Indicates the highlight color model. Highlight colors are device-dependent.

You can use an integer within the range of 0 - 65535 for the \textit{hvalue}.

\textbf{Note:} An \textit{hvalue} of 0 indicates that no default value is defined; therefore, the default color of the presentation device is used.

\textit{COVERAGE} indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 - 100 for the \textit{cvalue}. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

\textbf{Note:} Fractional values are ignored. If \textit{COVERAGE} is not specified, a value of 100 is used as a default.

\textit{BLACK} indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 - 100 for the \textit{bvalue}. The amount of black shading that is applied depends on the \textit{COVERAGE} percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

\textbf{Note:} If \textit{BLACK} is not specified, a value of 0 is used as a default.

**CMYK \textit{cvalue mvalue yvalue kvalue}**

Defines the cyan/magenta/yellow/black color model. \textit{cvalue} specifies the cyan value. \textit{mvalue} specifies the magenta value. \textit{yvalue} specifies the yellow value. \textit{kvalue} specifies the black value. You can use an integer percentage within the range of 0 - 100 for any of the CMYK values.

**CIELAB \textit{Lvalue (-)c1value (-)c2value}**

Defines the CIELAB model. Use a range of 0.00 to 100.00 with \textit{Lvalue} to specify the luminance value. Use signed integers from -127 to 127 with \textit{c1value} and \textit{c2value} to specify the chrominance differences.

\textit{Lvalue}, \textit{c1value}, \textit{c2value} must be specified in this order. No defaults exist for the subvalues.

\textbf{Notes:}

1. Do not specify both an OCA color with the COLOR subparameter and an extended color model on the same FIELD or PRINTLINE command. The output is device-dependent and might not be what you expect.

2. Do not specify two extended COLOR subcommands on the same FIELD or PRINTLINE command.
**FONT**<i>name1</i>, <i>name2</i>

Defines the font to be used for the layout.

<i>name1</i>

Specifies the name of a font that is used to print the data. This font must be defined in a previous FONT command in the page definition.

If Shift-Out, Shift-In (SOSI) processing is used, <i>name1</i> must be the single-byte font.

<i>name2</i>

Specify only when you are using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the layout. <i>name2</i> must be the double-byte font.

**Notes:**

1. If this subcommand is not specified in the print data, the print server uses the font indicated. Otherwise, the print server selects a default font.

2. <i>name2</i> is only valid with EBCDIC data.

**OBJECT**

Specifies the name of an object that is to be positioned and oriented relative to the location specified in the LAYOUT command in which the OBJECT subcommand was named. The OBJECT, as identified by the <i>lname</i> parameter, must be defined by an OBJECT command.

**Note:** Multiple page/image objects that are used without specifying a page with OBPAGE default to using the first page in the object.

You can place multiple objects on the same LAYOUT command and you can place the same object multiple times. Each placement must have its own set of placement parameters, as follows:

<i>lname</i>

Specifies the local name of an object that is up to 16 alphanumeric characters in length. The <i>lname</i> is used to match the LAYOUT OBJECT subcommand to its definition from the OBJECT command. An object must be defined with this local name by the OBJECT command.

<i>relX relY</i>

Specifies the number of units (such as inches or mm) that are added to the position of the current LAYOUT to position the top-left corner of the object. The values for the horizontal and vertical positioning are limited by the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.
Each position specification can be a positive or negative number with up to three decimal places. The units that are specified can be one of these: IN, MM, CM, POINTS, or PELS.

**VARIABLE**

Indicates that the actual name of the object is read from the data record. The Variable-Name-Locator field specifies where in the data to get the name.

**Notes:**
1. Any object that is to be included in this manner must be defined in the PAGEDEF by using the OBJECT command. Defining objects enhances performance.
2. If you specify VARIABLE for the OBJECT name and don’t want to print the name, then you must have at least one field command, or code PRINTDATA NO on the LAYOUT command.

**VARIABLE Parameters**

- **START** $n$
  The starting position in the data record to get the overlay name. The first data byte position of the input record is 1. If START is not coded, 1 is assumed.

- **LENGTH** $n$
  Length of field. Specifies the number ($n$) of bytes to process from the data record, beginning with the position specified in START. The maximum length is 8.

- **FLDNUM** $n$
  The field number. This is the same as in the FIELD command. The overlay name is taken from the $n$ field of the input data record. START $n$ and LENGTH $n$ describe which portion of the $n$ field is used. If omitted, the entire field is used to form the overlay name.

- **RECID**
  Gets the name from the record ID. This is the same as in the FIELD command. Use START $n$ and LENGTH $n$ to use only a portion of the record ID, or leave them out to use the entire record field.

- **OBTYPE**
<table>
<thead>
<tr>
<th>VARIABLE OBTYPE</th>
<th>PSEG</th>
<th>IOCA</th>
<th>BCOCA</th>
<th>GOCA</th>
<th>PTOCA</th>
<th>OTHER OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>comp-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>type-name</td>
</tr>
</tbody>
</table>

Used to specify the type of the object. Observe that each of the object types restricts the type of mapping option that is allowed in the placement of the object (OBMAP on the OBJECT subcommand on the PRINTLINE command).

**PSEG**

Specifies a page segment object, as described in the *Mixed Object Document Content Architecture Reference*. All mapping types (OBMAP) are allowed by PPFA; however, the print server issues an error if any of the objects that are contained in the page segment are not compatible with the coded OBMAP parameter.

**GOCA**

Specifies a graphic object, as described in the *Graphics Object Content Architecture Reference*. You use GOCA to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**BCOCA**

Specifies a bar code object, as described in the *Bar Code Object Content Architecture Reference*. You use BCOCA to specify only the LEFT parameter on the OBMAP subcommand.

**IOCA**

Specifies an image object, as described in the *Image Object Content Architecture Reference*. You use IOCA to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**PTOCA**

Specifies a presentation text object with Object Environment Group (OEG) as described in the *Presentation Text Object Content Architecture Reference* and the *Mixed Object Document Content Architecture Reference*. You use the PTOCA object type to specify the LEFT parameter in the OBMAP subcommand.

**OTHER**

Specifies other object data. The object data to be included is a paginated presentation object with a format that might or might not be defined by an AFP architecture. When you specify OTHER, you must also specify the OBID parameter. You use OTHER to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**OBID**

Specifies either a component identifier or a type name from Table 13 on page 378. The OBID is translated into an Encoded OID and matched to the OID inside the object; they must match.

- **comp-id**
  - Specifies the component identifier.
**type-name**

The name that is chosen by PPFA as an alternative to coding a component identifier.

**Table 13. Non-OCA Objects supported by IOB**

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OBID Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPS</td>
<td>13</td>
<td>Encapsulated PostScript</td>
</tr>
<tr>
<td>TIFF or TIF</td>
<td>14</td>
<td>Tag Image File Format</td>
</tr>
<tr>
<td>WINDIB</td>
<td>17</td>
<td>Device Dependent Bit Map [DIB], Windows Version</td>
</tr>
<tr>
<td>OS2DIB</td>
<td>18</td>
<td>Device Dependent Bit Map [DIB], PM Version</td>
</tr>
<tr>
<td>PCX</td>
<td>19</td>
<td>Paint Brush Picture File Format</td>
</tr>
<tr>
<td>GIF</td>
<td>22</td>
<td>Graphics Interchange Format</td>
</tr>
<tr>
<td>JFIF, JPEG, or JPG</td>
<td>23</td>
<td>AFPC (AFP Consortium) JPEG Subset</td>
</tr>
<tr>
<td>PDFSPO</td>
<td>25</td>
<td>PDF Single Page Object</td>
</tr>
<tr>
<td>PCLPO</td>
<td>34</td>
<td>PCL Page Object</td>
</tr>
<tr>
<td>EPSTR</td>
<td>48</td>
<td>EPS with Transparency</td>
</tr>
<tr>
<td>PDFSPOTR</td>
<td>49</td>
<td>PDF Single Page Object with Transparency</td>
</tr>
<tr>
<td>TIFF</td>
<td>61</td>
<td>TIFF Multiple Image File</td>
</tr>
<tr>
<td>MTIFF</td>
<td>62</td>
<td>TIFF Multiple Image without Transparency File</td>
</tr>
<tr>
<td>MTIFFNT</td>
<td>63</td>
<td>PDF Multiple Page File</td>
</tr>
<tr>
<td>MPDF</td>
<td>64</td>
<td>PDF Multiple Page with Transparency File</td>
</tr>
<tr>
<td>PNG</td>
<td>65</td>
<td>PNG File Format</td>
</tr>
<tr>
<td>AFPCTIFF</td>
<td>66</td>
<td>AFPC TIFF Subset</td>
</tr>
</tbody>
</table>

**Table 14. Object Types that can be referenced as Secondary Resources**

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OID Type-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFRO</td>
<td>26</td>
<td>PDF Resource Object (new)</td>
</tr>
<tr>
<td>RESCLRPRO</td>
<td>46</td>
<td>Resident Color Profile Resource Object</td>
</tr>
<tr>
<td>IOCAFS45RO</td>
<td>47</td>
<td>IOCA FS45 Resource Object Tile (new)</td>
</tr>
</tbody>
</table>

**Other OBJECT parameters**
OBRESOLUTION x y IN CM

OBSIZE

OBJECT

OBSIZE wd unit

USEOBJ

OBCOLOR colorname

OBPAGE n

OBROTATE

y-pos

OBCVPOS

OBCVPOS USEOBJ

OBROTATE 0

180

270

180

270

OBMAP

Chapter 11. Page Definition Command Reference 379
Specifies mapping options. The OBMAP parameter defines the mapping of the object to the object placement area. If OBMAP is not coded, the mapping option within the object is used. If the object does not contain a mapping option, then the print server sets it to the created default for the container type.

Each object type (OBTYPE on the OBJECT command) dictates the allowable mapping options for that type. When it can, PPFA issues a message when these rules are violated. However, for an object type of page segment (OBTYPE=PSEG), PPFA does not know what types of objects are contained in it; therefore, PPFA cannot enforce the restrictions. See "OBJECT Command" on page 385 for a description of the restrictions.

LEFT

Specifies that the object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the relative-xpos, relative-ypos, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is not trimmed and might cause an exception condition by the presentation system.

TRIM

Specifies position and trim. The object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the relative-xpos, relative-ypos, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is trimmed.

FIT

Specifies scale to fit; this value is the default if the OBMAP parameter is not coded. The object is to be scaled to fit within the object placement area, as defined by the OBSIZE parameter. The center of the object is placed in the center of the object placement area and the object is scaled up or down to fit the block. Scaling in the horizontal and vertical directions is symmetrical. The FIT parameter ensures that all of the data in the object is presented in the object placement area at the largest possible size. The object is not trimmed.

CENTER

Specifies that the center of the object is positioned at the center of the object placement area. Any portion of the object that falls outside the object placement area is trimmed.

REPEAT

Specifies that the origin of the data object is positioned with the origin of the object placement area. The object is then replicated in the X and Y directions. If the last replicated data does not fit in the object area, it is trimmed to fit.
**FILL**

Specifies that the center of the data object is positioned coincident with the center of the object placement area. The data object is then scaled so that it totally fills the object placement area in both the X and Y directions. This might require that the object is asymmetrically scaled by different scale factors in the X and Y directions.

**OBCHPOS**

- **OBJECT**
  - **OBCHPOS—USEOBJ**
  - **OBCHPOS—x-pos**

Specifies the horizontal offset of the object contents within the object placement area as a number.

- **x-pos**
  - Specifies a positive or negative number. The valid options for x-pos are described in the SETUNITS command for the horizontal value.

- **USEOBJ**
  - Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

**OBCVPOS**

- **OBJECT**
  - **OBCVPOS—USEOBJ**
  - **OBCVPOS—y-pos**

Specifies the vertical offset of the object contents within the object placement area, as defined by the OBSIZE parameter. If OBCVPOS is not specified, it defaults to USEOBJ and uses the value set in the object. If no value is set in the object, the value defaults to 0. The OBCHPOS parameter is used only in LEFT and TRIM mapping of the object into the object placement area.

- **y-pos**
  - Specifies a positive or negative number. The valid options for y-pos are described in the SETUNITS command for the vertical value.

- **USEOBJ**
  - Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

**OBROTATE \{0\ | 90\ | 180\ | 270\}**

- **OBJECT**
  - **OBROTATE—0**
  - **OBROTATE—90**
  - **OBROTATE—180**
  - **OBROTATE—270**

Specifies the object rotation in regard to the current LND’s coordinate system.
OBCOLOR

Specifies the color to be used as the default color or initial color for the object placement area. The OBCOLOR parameter is used only for objects of the PSEG, GOCA, BCOCA, and IOCA type. If the object type is OTHER, this parameter is ignored.

colormame

Values for colormame can be a defined color (see “DEFINE COLOR Command” on page 280) or one of these OCA color spaces:

- NONE
- DEFAULT
- BLACK
- BLUE
- BROWN
- GREEN
- RED
- PINK (or MAGENTA)
- TURQ (or CYAN)
- YELLOW
- DARKBLUE (or DBLUE)
- ORANGE
- PURPLE
- MUSTARD
- GRAY
- DARKGREEN (or DGREEN)
- DARKTURQ (DTURQ, or DCYAN, or DARKCYAN)

In the following example, the page definition pd1 defines an object with an external name of PSEGxyz, of object type PSEG. The object has an internal name of xyzintname. The internal name identifies the object for the LAYOUT OBJECT subcommand when the object is placed. Observe that case is not significant on either the internal nor the external names.

```
PAGDEF pd1 Replace Yes
   COMMENT 'this is my program';
   FONT XF1 ;

   OBJECT xyzIntName
      OBXNAME PSEGxyz
      OBTYPE PSEG ;

   PAGEFORMAT pf1;
      LAYOUT 'abc' POSITION 2 in 1 in;
      OBJECT xyzintname 1.1 in 2.1 in
      OBSIZE 3 in 5 in
      OBMAP FILL
      OBCOLOR BLUE ;
```

The LAYOUT in PAGEFORMAT pf1 places the object on the page 1.1 inches to the left and 2.1 inches below the current LAYOUT position. It also maps the object into the object area with the FILL parameter, which centers the object in the object area and totally fills the area, possibly with different scaling factors in the X and Y
directions. It has an area size of 3 by 5 inches, and overrides the default presentation space color to BLUE.

**OBPAGE**

```
OBPAGE - n
```

Specifies the page number of a multipage object or file to be presented. $n$ is the page number. A number 1 - 999999999 (9 digits) is valid.

**OBRESOLUTION**

```
OBRESOLUTION - x - y - IN
```

Specifies the resolution and unit of measurement of an image. If the resolution is already specified inside the image, this information is ignored by the printer. Use this subcommand for images that do not or might not contain their resolution. Specify resolution of an image so that the printer can print the image correctly.

To specify object resolution, you must have a printer and a print server (PSF or IPM) that support this capability.

If not specified, the default is to assume that the image resolution is the same as the printer. If the image does not print at the size you expect, use OBRESOLUTION to identify the image's resolution. With the resolution information, the printer is then able to print the image at the expected size.

$x$ Specifies the number to be used for the horizontal resolution of an image. Specify an integer value in the range of 1 - 3276.

$y$ Specifies the number to be used for the vertical resolution of an image. Specify an integer value in the range of 1 - 3276.

unit Specifies a unit of measurement. The choices are:

- **IN** Inch
- **CM** Centimeter

**Example:** In the following code example, the OBJECT subcommand is used to define a JFIF object (which can be specified as JPG). This object has a resolution of 300 pels per inch in both the $x$ and $y$ directions.

```
Pagedef obres2 replace yes;

PRINTLINE OBJECT VAR .4 .5 start 2 length 6
OBTYPE OTHER OBID JPG
OBRESOLUTION 300 300 IN;
```

**OVERLAY**
Specifies the name of an overlay that is to be positioned relative to the location specified in the LAYOUT command in which the OVERLAY subcommand was named. The PAGEFORMAT OVERLAY command can contain the named overlays. The maximum number of overlays that are specified for a PAGEFORMAT including the LAYOUT OVERLAY subcommand is 254.

Specifies the electronic overlay that is to be used with this subgroup.

**Xname**

Specifies the user-access name as defined in the OVERLAY command.

**Notes:**

1. PPFA checks for duplication of local names. If a duplication exists, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

**relX relY**

Specifies the number of units (such as inches or mm) that are added to the position of the layout to position the top-left corner of the overlay. The values for horizontal and vertical can be (+) or (-). The maximum value is + or - 32760 L-units. For example:

- OVERLAY NAME1 2 in 1 in
- OVERLAY NAME2 5 mm 1 mm

**Note:** Any offset coded in the overlay itself is added to this offset.

**VARIABLE**

Indicates that the actual name of the overlay, including the O1 prefix, is read from the data record. The Variable-Name-Locator field specifies where in the data to get the name.

**Notes:**

1. Any overlay that is to be included in this manner must be defined in the PAGEFORMAT by using the OVERLAY command. Any overlay that is included but not defined causes a runtime print error for a missing MPO structured field, for example APS263I.
2. If you specify VARIABLE for the OVERLAY name and don't want to print the name, then you must have at least one field command, or code PRINTDATA NO on the LAYOUT command.

**VARIABLE Parameters**

See [VARIABLE Parameters] for the syntax. The field number and record ID are only used for Record Layout and XML Page definitions.

**OVROTATE {0|90|180|270}**

Specifies the rotation of the placed overlay in regard to the x-axis of the page.

See "FORMDEF Command" on page 234 for an OVROTATE example, which is presented in the FORMDEF description.
### SEGMENT Command

Specifies the name of a segment that is to be positioned relative to the location specified in the LAYOUT command in which the SEGMENT subcommand was named. The PAGEFORMAT SEGMENT command can contain the named segments. The maximum number of segments that are specified for a PAGEFORMAT including the LAYOUT SEGMENT subcommand is 127.

Specifies the page segment that is to be used with this subgroup.

#### Xname

Specifies the user-access name as defined in the SEGMENT command.

#### Notes:

1. PPFA checks for duplication of local names. If a duplication exists, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

#### relX relY

Specifies the number of units (such as inches or mm) that are added to the position of the layout to position the top-left corner of the page segment. The values for horizontal and vertical can be (+) or (-). The maximum value is + or - 32760 L-units. For example:

- SEGMENT MYSEG1 2 in 1 in
- SEGMENT MYSEG1 5 mm 1 mm

#### VARIABLE

Indicates that the actual name of the segment, including the S1 prefix, is read from the data record. The Variable-Name-Locator field specifies where in the data to get the name.

#### Note:

If you specify VARIABLE for the SEGMENT name and don’t want to print the name, then you must have at least one field command, or code PRINTDATA NO on the LAYOUT command.

#### VARIABLE Parameters

See [VARIABLE Parameters](#) for the syntax. The field number and record ID are only used for Record Layout and XML Page definitions.

---

### OBJECT Command

<table>
<thead>
<tr>
<th>OBJECT—Iname—OBXNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
<tr>
<td>x-name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBTYPE</th>
<th>PSEG</th>
<th>100CA</th>
<th>800CA</th>
<th>600CA</th>
<th>500CA</th>
<th>400CA</th>
<th>300CA</th>
<th>200CA</th>
<th>100CA</th>
<th>100CA</th>
<th>000CA</th>
<th>OTHERS</th>
<th>OBID</th>
<th>component-id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Chapter II. Page Definition Command Reference 385
You use the OBJECT command to define an external object to PPFA. Then, you can use the PRINTLINE command (Traditional) or the LAYOUT command (Record Format and XML) with the OBJECT subcommand to place the defined object on a page.

You can use one PRINTLINE command (Traditional), LAYOUT command (Record Format), or XLAYOUT command (XML) to place one or many defined objects multiple times with different placement parameters on each placement. On the PRINTLINE OBJECT subcommand, enter information about the positioning, rotation, color, object size, page number, and mapping instructions. All positioning
is relative to the print line coordinate system. The \textit{name} appears on both the
OBJECT command and on the PRINTLINE OBJECT subcommand (Traditional), the
LAYOUT OBJECT command (Record Format), and the XLAYOUT OBJECT
command (XML), and is used similar to the way overlays and page segments are
defined and placed (or printed).

Notes:
1. The \textit{name} is case insensitive but the \textit{name} of the OBJECT command and of the
PRINTLINE OBJECT subcommand (Traditional), the LAYOUT OBJECT
command (Record Format), and the XLAYOUT OBJECT command (XML) must
match exactly.
2. This function requires both the print server and printer support. Check your
print server and printer documentation.
3. Fonts that are used by the OBJECT must be mapped. You can use the EXTREF
command to map a font.
4. CMRs used by the OBJECT must be mapped. You can use the EXTREF
OB2CMR command to map a CMR.

\textbf{OBJECT \textit{name}}

Identifies the object and also is used to match a PRINTLINE OBJECT
subcommand (Traditional), the LAYOUT OBJECT command (Record Format),
and the XLAYOUT OBJECT command (XML). The \textit{name} can be no more than
16 alphanumeric characters.

\textbf{Subcommands}

\textbf{OBXNAME}

\begin{verbatim}
  OBXNAME -- x-name
  -'x-name'
  -C'x-name'
  -E'x-name'
  -A'x-name'
  -X 'hhhh'
  -UB'x-name'
  -XB 'hhhh'
  -UB16'x-name'
  -XB16 'hhhh'
\end{verbatim}

Specifies the external user access name of the resource object. The name
indicates where on the file system the object is located.

Notes:
1. As a file system name, it must adhere to the rules of the operating system
where the object is located. For example, the \textit{x-name} in z/OS is the member
name of the object in the object library, so it must be 8 characters or less
and in uppercase EBCDIC code page 500.
2. All the translations assume code page International #5 (code page 500) for
EBCDIC and LATIN1 ISO/ANSI 8-bit (code page 819) for ASCII.

\textit{x-name}

Unquoted name up to 250 characters long is folded to uppercase and
translated into EBCDIC if necessary.
'x-name'
Quoted name up to 250 characters long is accepted as-is with no case folding or translation.

'C'x-name'
Quoted name with a “C” for Character is treated the same as a quoted name up to 250 characters. No folding or translation is done.

'A'x-name'
Quoted name with an “A” for ASCII entered with up to 250 single-byte characters is accepted as-is if on an ASCII operating system or converted to ASCII if on an EBCDIC operating system. The conversion is made with no case folding.

'E'x-name'
Quoted name with an E for EBCDIC entered with up to 250 single-byte characters is accepted as-is if on an EBCDIC operating system or converted to EBCDIC if on an ASCII operating system. The conversion is made with no case folding.

'X'hhhh'
Quoted name with an X for hexadecimal is entered with up to 500 hexadecimal characters. The characters are converted to hexadecimal, but no assumption of data type is made.

'U8'x-name'
Quoted name with an U8 for UTF-8 entered with up to 250 single-byte characters is translated to UTF-8.

'X8'hhhh'
Quoted name with an X8 for UTF-8 HEX entered with up to 500 single-byte hexadecimal characters is translated to hexadecimal and assumed to be data type UTF-8. A multiple of two hexadecimal characters must be entered.

'U16'x-name'
Quoted name with a U16 for UTF-16 entered with up to 125 single-byte characters is translated to UTF-16.

'X16'hhhh'
Quoted name with an X16 for UTF-16 HEX is entered with up to 500 single-byte hexadecimal characters. The characters are translated to hexadecimal and assumed to be data type UTF-16. A multiple of four hexadecimal characters must be entered.

To have portability across older versions of print servers in multiple system environments, it is recommended that resource object names use EBCDIC encoding with code page 500.

**OBTYPER**

```
OBTYPE
   | PSEG
   | JOC
   | BTOCA
   | GOCA
   | PTOCA
   | OTHER
   | OBID
   | component-id
   | type-name
```
Used to specify the type of the object. Observe that each of the object types restricts the type of mapping option that is allowed in the placement of the object (OBMAP on the OBJECT subcommand on the PRINTLINE command (Traditional) or the LAYOUT command (Record Format and XML)).

**PSEG**

Specifies a page segment object, as described in the *Mixed Object Document Content Architecture Reference*. All mapping types (OBMAP) are allowed by PPFA; however, the print server issues an error if any of the objects that are contained in the page segment is not compatible with the coded OBMAP parameter.

**GOCA**

Specifies a graphics object, as described in the *Graphics Object Content Architecture Reference*. You use GOCA to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**BCOCA**

Specifies a bar code object, as described in the *Bar Code Object Content Architecture Reference*. You use BCOCA to specify only the LEFT parameter on the OBMAP subcommand.

**IOCA**

Specifies an image object, as described in the *Image Object Content Architecture Reference*. You use the IOCA object type to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**PTOCA**

Specifies a presentation text object with Object Environment Group (OEG) as described in the *Presentation Text Object Content Architecture Reference* and the *Mixed Object Document Content Architecture Reference*. You use the PTOCA object type to specify the LEFT parameter in the OBMAP subcommand.

**OTHER**

Specifies other object data. The object data that is included is a paginated presentation object with a format that might be defined by an AFP presentation architecture. When you specify OTHER, you must also specify the OBID parameter. You use the OTHER object type to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**OBID**

Specifies either a component identifier or a type name from Table 15. The OBID is translated into an Encoded OID and matched to the OID inside the object; they must match.

<table>
<thead>
<tr>
<th>component-id</th>
<th>Specifies the component identifier.</th>
</tr>
</thead>
<tbody>
<tr>
<td>type-name</td>
<td><em>Type-name</em> is a name that is chosen by PPFA as an alternative to coding a component identifier.</td>
</tr>
</tbody>
</table>

*Table 15. Non-OCA Objects supported by IOB.*

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OBID Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPS</td>
<td>13</td>
<td>Encapsulated PostScript</td>
</tr>
<tr>
<td>TIFF or TIF</td>
<td>14</td>
<td>Tag Image File Format</td>
</tr>
<tr>
<td>WINDIB</td>
<td>17</td>
<td>Device Dependent Bit Map [DIB], Windows Version</td>
</tr>
</tbody>
</table>
Table 15. Non-OCA Objects supported by IOB. (continued)

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OBID Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS2DIB</td>
<td>18</td>
<td>Device Dependent Bit Map [DIB], PM Version</td>
</tr>
<tr>
<td>PCX</td>
<td>19</td>
<td>Paintbrush Picture File Format</td>
</tr>
<tr>
<td>GIF</td>
<td>22</td>
<td>Graphics Interchange Format</td>
</tr>
<tr>
<td>JFIF, JPEG, or JPG</td>
<td>23</td>
<td>AFPC (AFP Consortium) JPEG Subset</td>
</tr>
<tr>
<td>PDFSPO</td>
<td>25</td>
<td>PDF Single Page Object</td>
</tr>
<tr>
<td>PCLPO</td>
<td>34</td>
<td>PCL Page Object</td>
</tr>
<tr>
<td>EPSTR</td>
<td>48</td>
<td>EPS with Transparency</td>
</tr>
<tr>
<td>PDFSPOTR</td>
<td>49</td>
<td>PDF Single Page Object with Transparency</td>
</tr>
<tr>
<td>MTIFF</td>
<td>61</td>
<td>TIFF Multiple Image File</td>
</tr>
<tr>
<td>MTIFFNT</td>
<td>62</td>
<td>TIFF Multiple Image without Transparency File</td>
</tr>
<tr>
<td>MPDF</td>
<td>63</td>
<td>PDF Multiple Page File</td>
</tr>
<tr>
<td>MPDFT</td>
<td>64</td>
<td>PDF Multiple Page with Transparency File</td>
</tr>
<tr>
<td>PNG</td>
<td>65</td>
<td>PNG File Format</td>
</tr>
<tr>
<td>AFPCTIFF</td>
<td>66</td>
<td>AFPC TIFF subset</td>
</tr>
</tbody>
</table>

Table 16. Object Types that can be referenced as Secondary Resources

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OID Type-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFRO</td>
<td>26</td>
<td>PDF Resource Object (new)</td>
</tr>
<tr>
<td>RESCLRPRO</td>
<td>46</td>
<td>Resident Color Profile Resource Object</td>
</tr>
<tr>
<td>IOCAFS45RO</td>
<td>47</td>
<td>IOCA FS45 Resource Object Tile (new)</td>
</tr>
</tbody>
</table>

REN DER

Subcommand on the OBJECT command to specify the rendering intent (RI) for an object within a page definition.

RI is used to modify the final appearance of color data and is defined by the International Color Consortium (ICC). For more information about RI, see the current level of the ICC Specification.

Not all object types have rendering intent; in that case, rendering intent is ignored. The following list shows object types that can be specified as parameters on the OBTYPE subcommand, and the resulting rendering intent object type:
- PSEG - rendering intent that is specified on a PSEG is used for all object types
- IOCA - supported
- BCOCA - not supported. Rendering intent for BCOCA objects is fixed as media-relative colorimetric (RELCM).
- GOCA - supported
- PTOCA - supported
• OTHER - supported

**rendering intent parameter**

Specify the rendering intent for the preceding object type.

**PERCEPTUAL**

Perceptual rendering intent. It can be abbreviated as PERCP. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to give a pleasing appearance. This intent is typically used to render continuous-tone images.

**SATURATION**

Saturation rendering intent. It can be abbreviated as SATUR. With this rendering intent, gamut mapping is vendor-specific, and colors are adjusted to emphasize saturation. This intent results in vivid colors and is typically used for business graphics.

**RELCM**

Media-relative colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered in regard to the source white point and are adjusted for the media white point. Therefore, colors that are printed on two different media with different white points do not match colorimetrically, but might match visually. This intent is typically used for vector graphics.

**ABSCM**

ICC-absolute colorimetric rendering intent. In-gamut colors are rendered accurately, and out-of-gamut colors are mapped to the nearest value within the gamut. Colors are rendered only in regard to the source white point and are not adjusted for the media white point. Therefore, colors that are printed on two different media with different white points must match colorimetrically, but might not match visually. This intent is typically used for logos.

**CMYKSWOP | CMYKEURO**

Indicates the color profile if it is required by the object.

**OBNOKEEP**

This object name is not included in a Map Data Resource structured field so that the object is loadable each time the object is placed on the page.

**OBKEEP**

This object is included in a Map Data Resource at the beginning of the
PAGEDEF making a hard object at the beginning of the page and then available throughout without reloading.

**Note:** Only objects with OBTYPE IOCA and OTHER can be kept; otherwise, OBKEEP is ignored.

**NOPRELOAD**

Do not preload this object.

**PRELOAD**

Preload this object before processing the print job.

**NOPRERIP**

Do not PreRip this object.

**PRERIP**

Prepare an object for printing by rasterizing it with its presentation parameters — object area size, object mapping option, object content offset, and object rotation in regard to the media leading edge.

**PreRip Parameters**

These parameters are used to specify the exact rasterization of the object, its size, offset, mapping, and rotations.

**Notes:**

1. To specify multiple pages, rip sizes, mappings, and offsets for the same object, code multiple object commands.

2. Mapping an object (also known as KEEPing) enhances throughput by allowing the printer to download an object once and use it on subsequent pages of the same print job or possibly on subsequent print jobs. Secondary objects are always mapped.

3. Preloading an object consists of loading the object into the printer memory before the print job is started. This enhances throughput because it removes the downloading time from print-time to page build time.

4. Preripping further enhances throughput because it allows the object and its secondary objects to be rasterized (RIPped) at the proper size and rotation when they are preloaded. When a primary object is preripped, all its secondary objects are also preloaded and preripped.

**RIPSIZE**

```
RIPSIZE  USEOBJ
  wd  unit
  hg  unit
```

Specifies the size of the object placement area. When no RIPSIZE is specified, the default is the size that is specified in the object. If no size is specified in the object, the size of the page is used. The page width is specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.
USEOBJ

Specifies that the size measurements that are specified in the object are to be used. If no size is specified in the object, the size of the page is used, which is the length and width as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

\textit{wd} Specifies the width of an object placement area as a number with up to three decimal places. The allowable width can vary with the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

\textit{hg} Specifies the height of an object placement area as a number with up to three decimal places. The allowable height can vary with the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

\textit{unit}

Specifies a unit of measurement for the width parameter. The choices are: IN, MM, CM, POINTS, or PELS.

\textbf{Note:} If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

RIPMAP

\begin{center}
\begin{tabular}{|c|}
\hline
RIPMAP \hspace{1cm} \textit{LEFT} \\
\textit{TRIM} \\
\textit{FIT} \\
\textit{CENTER} \\
\textit{FILL} \\
\hline
\end{tabular}
\end{center}

Specifies mapping options. The RIPMAP parameter defines the mapping of the object to the object placement area. If RIPMAP is not coded, the mapping option within the object is used. If the object does not contain mapping option, then the print server sets it to the created default for container type. Each object type (OBTYPE on the OBJECT command) specifies the allowable mapping options for that type. See the OBJECT OBTYPE parameter for a description of the restrictions.

\textbf{FIT}

Specifies scale to fit. This is the default value of the RIPMAP parameter is not coded. The object is to be scaled to fit within the object placement area, as defined by the RIPSIZE parameter. The center of the object is placed in the center of the object placement area and the object is scaled up or down to fit the block. Scaling in the horizontal and vertical directions is symmetrical. The FIT parameter ensures that all of the data in the object is presented in the object placement area at the largest possible size. The object is not trimmed.

\textbf{FILL}

Specifies that the center of the data object is positioned
coincident with the center of the object placement area. The data object is then scaled so that it totally fills the object placement area in both the X and Y directions. This might require that the object is asymmetrically scaled by different scale factors in the X and Y directions.

**LEFT**

Specifies that the object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the `relative-xpos`, `relative-ypos`, and `RIPOFFSET` parameters. Any portion of the object that falls outside the object placement area as defined by the RIPSIZE parameter is not trimmed and might cause an exception condition by the presentation system. This mapping type is invalid with an IOCA object.

**TRIM**

Specifies position and trim. The object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the `relative-xpos`, `relative-ypos`, and `RIPOFFSET` parameters. Any portion of the object that falls outside the object placement area as defined by the RIPSIZE parameter is trimmed.

**CENTER**

Specifies that the center of the object is positioned at the center of the object placement area. Any portion of the object that falls outside the object placement area is trimmed.

**RIPOFFSET**

![RIPOFFSET Diagram]

Object Content Offset - Specifies the horizontal and vertical offset of the object contents within the object placement area, as defined by the RIPSIZE parameter. If RIPOFFSET is not specified, the object is preprocessed and cached at its full size. The content offset specified at Include time is then used to place and possibly trim the object to the object area, with an associated performance penalty.

The RIPOFFSET parameter is used only in LEFT and TRIM mapping of the object into the object placement area.

**rel x**

Specifies the offset along the X-axis of the object area coordinates system. This can be a positive or negative number.

**rel y**

Specifies the offset along the Y-axis of the object area coordinates system. This can be a positive or negative number.

**unit**

Specifies a unit of measurement for the width parameter. The choices are: IN, MM, CM, POINTS, or PELS.
**Note:** If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

RIPROTATE { 0 | 90 | 180 | 270 }

Specifies the object rotation in regard to the leading edge of the media. Up to four rotations can be specified.

**Note:** Many factors, such as media selection, media side, media loading media orientation, page orientation, and object area rotation affect the orientation of an object in regard to the media leading edge. Therefore, proper specification of this parameter might require visual inspection of physical output.

RIPPAGE

Specifies the page number of a multipage object or file to be pre-RIPped. \( n \) is the page number. A number 1 - 999999999 (9 digits) is valid. ALL specifies to pre-RIP all objects in a multipage object.

OB2RESOURCE

If the primary object contains a reference to one or more secondary objects, you must identify them. Specify the internal name for the secondary resource as specified in the primary resource. If the internal name contains special characters such as periods or blanks, then quotation marks must surround the name.

\( i2name \)

Unquoted name up to 250 characters long is folded to uppercase and translated into EBCDIC if necessary.
'i2name'
Quoted name up to 250 characters long is accepted as-is with no case folding or translation.

C'i2name'
Quoted name with a “C” for Character is treated the same as a quoted name of up to 250 characters. No folding or translation is done.

A'i2name'
Quoted name with an “A” for ASCII entered with up to 250 single-byte characters is accepted as-is if on an ASCII operating system or translated to ASCII if on an EBCDIC operating system. The translation is made with not case folding.

E'i2name'
Quoted name with an “E” for EBCDIC entered with up to 250 single-byte characters is accepted as-is if on an EBCDIC operating system or translated to EBCDIC if on an ASCII operating system. The translation is made with not case folding.

X'hhhh'
Quoted name with an X for Hexadecimal entered with up to 500 hexadecimal characters. The characters are translated to hexadecimal, but no assumption of data type is made.

OB2XNAME

Specifies the external name for a secondary resource object. The name can be up to 250 characters. If the name contains special characters or blanks, it must be enclosed in blanks.

Note: A file system name must adhere to the rules of the operating system where the object is located. This might further restrict the following sizes.

x2name
Unquoted name up to 250 characters long is folded to uppercase and translated into EBCDIC if necessary.

'x2name'
Unquoted name up to 250 characters long is accepted as-is with no case folding or translation.

C'x2name'
Quoted name with a C for Character is treated the same as a quoted name up to 250 characters. No folding or translation is done.

A'x2name'
Quoted name with an A for ASCII entered with up to 250 single-byte
characters is accepted as-is if on an ASCII operating system or translated to
ASCII if on an EBCDIC operating system. The translation is made with no
case folding.

\texttt{E'x2name'}

Quoted name with an E for EBCDIC entered with up to 250 single-byte
characters is accepted as-is if on an EBCDIC operating system or translated
to EBCDIC if on an ASCII operating system. The translation is made with
no case folding.

\texttt{X'hhhh'}

Quoted name with an X for Hexadecimal entered with up to 500
hexadecimal characters. The characters is translated to hexadecimal, but no
assumption of data type is made.

\texttt{U8'x2name'}

Quoted name with an U8 for UTF-8 entered with up to 250 single-byte
characters is translated to UTF-8.

\texttt{X8'hhhh'}

Quoted name with an X8 for UTF-8 HEX entered with up to 500
single-byte hexadecimal characters is translated to hexadecimal and
assumed to be data type UTF-8. A multiple of two hexadecimal characters
must be entered.

\texttt{U16'x2name'}

Quoted name with an U16 for UTF-16 entered with up to 125 single-byte
characters is translated to UTF-16.

\texttt{X16'hhhh'}

Quoted name with an X16 for UTF-16 HEX entered with up to 500
single-byte hexadecimal characters is translated to hexadecimal and
assumed to be data type UTF-16. A multiple of four hexadecimal characters
must be entered.

All specified secondary resources are kept. See OBKEEP for more information.

\texttt{OB2ID n | type-name}

Component type identifier for secondary resource. Use an object type number
as specified in Object type list adjustments. Use an object type number from
the Component-ID column or a type name from the Type-Name column of
Table 16 on page 390.

\texttt{NOPRELOAD | PRELOAD}

All specified secondary resources are kept. If you want the secondary object to
be preloaded before the running of this job, specify it here.

\texttt{OB2CMR}
Specify a Color management resource (CMR) and its process mode for a data object within the PAGEDEF. CMRs are secondary objects when used at this level. Multiple OB2CMR subcommands are allowed on the OBJECT command.

Note: See Chapter 8, “AFP Color Management,” on page 163 for more information about using the CMR subcommand.

cmr-lname
The CMR local name. This name must be defined with a DEFINE CMRNAME command.

processing mode parameter
Specify the processing mode for the CMR.

AUDIT
CMRs with the audit processing mode refer to processing that is already applied to a resource. In most cases, audit CMRs describe input data and are similar to ICC input profiles.

The audit processing mode is used primarily with color conversion CMRs. In audit processing mode, those CMRs indicate which ICC profile must be applied to convert the data into the Profile Connection Space (PCS).

INSTR
CMRs with the instruction processing mode refer to processing that is done to prepare the resource for a specific printer by using a certain paper or another device. Generally, instruction CMRs refer to output data and are similar to ICC output profiles.

The instruction processing mode is used with color conversion, tone transfer curve, and halftone CMRs. In instruction processing mode, these CMRs indicate how the system must convert a resource so it prints correctly on the target printer. The manufacturer of your printer must provide ICC profiles or various CMRs that you can use. Those ICC profiles and CMRs might be installed in the printer controller, included with the printer on a CD, or available for download from the manufacturer's website.

LINK
This CMR defines a direct color conversion from an input color space to a device output color space; process the CMR as a link CMR. This processing mode is only valid for device link (DL) CMRs. The PPFA command RENDER is not used with device link (DL) CMRs as such CMRs specify the intended rendering intent internally. This function requires print server (PSF) and printer support, which is in addition to the original CMR support.

NOPRELOAD|PRELOAD
All specified secondary resources are kept. If you want the CMR object to be preloaded before the running of this job, specify it here.

Example: In the following code example, an object with CMR is defined. The following LAYOUT commands place the object on the page. The CMR name is
defined and referenced by the CMR local name. See the DEFINE CMRNAME command for examples and instructions on defining CMR names.

```plaintext
PAGEDEF cmr89 replace yes;
    FONT varb gt10 ; /*Variable data */
    SETUNITS LINESP .25 in ; /* Line spacing */

DEFINE srgb CMRNAME
    'sRGBicc_CC001.000@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@'
        '000000';

Object oc1 obxname 'Flowers_with_sRGB_profile'
    obtype other obid 23 OBKEEP
    ob2cmr srgb audit

    PAGEFORMAT rept1 TOPMARGIN 1 in BOTMARGIN 2 in;
    LAYOUT 'startpage' BODY NEWPAGE POSITION 1 in NEXT
        font varb
    object oc1 0 in 3 in obsize 6.5 in 8.5 in;
    LAYOUT 'basicline' BODY POSITION SAME NEXT font varb;
```

OBRESOLUTION

```
      OBRESOLUTION x-y IN
            CM
```

Specifies the resolution and unit of measurement of an image. If the resolution is already specified inside the image, this information is ignored by the printer. Use this subcommand for images that do not or might not contain their resolution. Specify resolution of an image so that the printer can print the image correctly.

To specify object resolution, you must have a printer and a print server (PSF or IPM) that support this capability.

If not specified, the default is to assume that the image resolution is the same as the printer. If the image does not print at the size you expect, use OBRESOLUTION to identify the image's resolution. With the resolution information, the printer then able print the image at the expected size.

- `x` Specifies the number to be used for the horizontal resolution of an image. Specify an integer value in the range of 1 - 3276.
- `y` Specifies the number to be used for the vertical resolution of an image. Specify an integer value in the range of 1 - 3276.
- `unit` Specifies a unit of measurement. The choices are:
  - **IN** Inch
  - **CM** Centimeter

In the following code example, the OBJECT command is used to define two JFIF objects. One is pre-ripped and the other is not. One has a resolution of 300 pels per inch in both the `x` and `y` directions. The other has a resolution of 600 pels per inch in both the `x` and `y` directions.

```plaintext
SETUNITS 2 in 2 in;
Pagedef obxres

OBJECT obres1 OBXNAME xpseg23 OBTYPE other OBID JFIF
    OBRESOLUTION 300 300 IN;

OBJECT obres2 OBXNAME xpseg24 OBTYPE other OBID JFIF
    OBRESOLUTION 600 600 IN;
```
Examples

Example 1

In the following page definition, several examples of long names are shown, which is for illustration only.

- **OBU8** — The primary object name is specified in U8 format, which means that it is specified as a character string and translated to UTF-8 encoding.

  The secondary object that is defined on object OBU8 is referenced in the primary object with an identifier that is the equivalent of hexadecimal X'ABF8'. The external name for that object is specified in C format, which means that the name is accepted as it with no translation or folding of case.

- **OBU16** — This object name is specified in U16 format, which means that it is specified as a character string and translated to UTF-16 encoding.

- **OBX16** — This object name is specified in X16 format, which means that it is specified as a hexadecimal string that represents the UTF-16 encoding. It is not translated. The only check that PPFA makes is that its length is a multiple of 4.

```
PAGEDEF LNNG2P REPLACE YES;
  FONT FN1 GT10;
  OBJECT obU8 OBXNAME u8'A Long Object Name in UFT8'
    'Which is continued on a second line'
    'And could also be continued on a subsequent line'
    OBTYPE IOCA OBKEEP PRELOAD
    OB2RESOURCE X'ABF8' OB2XNAME C'A plain old Character'
    'type Secondary Object name which will be used as typed'
    'in the code page of the User' OB2ID PDFRO PRELOAD;

  OBJECT OBU16 OBXNAME U16'abcdef4'
    OBTYPE IOCA OBKEEP PRELOAD;

  OBJECT obx16 OBXNAME X16'006100620063'
    '0064006500660034'
    OBTYPE IOCA OBKEEP PRELOAD;

printline object obU8 FONT fn1;
printline object obU16;
printline object obX16;
```

Example 2

In the following page definition, an IOCA object is defined and placed. The object is to be mapped, preloaded, and preripped in 3 orientations (0, 90, 270). Object area size and offset mapping are specified. TRIM mapping specifies that the object is to be placed in the upper-left corner of the object area, as defined by the PRINTLINE position and RIPOFFSET parameters, and, if necessary, trimmed to the object area size, as defined by RIPSIZE.

```
PAGDEF RipXml Replace Yes;
  OBJECT ripit OBXNAME FS45pic OBTYPE IOCA
    OBKEEP PRELOAD
    PRERIP
      RIPSIZE 3.0 in 4.0 in
      RIPMAP trim
```
Example 3

In the following page definition, multiple page objects are defined and placed. To specify the page number, see the descriptions of the parameters RIPAGE and OBPAGE.

```plaintext
/*---------------------------------------------*/
/* MULTIEX2 multipage objects - Printline */
/* Examples */
/* */
/*---------------------------------------------*/
setunits 1 in 1 linesp 6 lpi;
pagedef multx2 replace yes pelsperinch 600
   width 8.5 height 11.0;

font f1 gt10;
/*---------------------------------------------*/
/* Define objects */
/*---------------------------------------------*/
/*---------------------------------------------*/
/* Multipage objects OBID type name */
/*---------------------------------------------*/
object npt02 obxname TIFFNT1 obtype other obid MTIFFNT;

/*---------------------------------------------*/
/* PRERIP ALL Multipage objects OBID component id */
/*---------------------------------------------*/
object pac03 obxname MPDF6 obtype other obid 63
   OBKEEP PRELOAD PRERIP RIPPAGE ALL;

/*---------------------------------------------*/
/* PRERIP page Multipage objects OBID type name */
/*---------------------------------------------*/
object ppt04 obxname MPDFT7 obtype other obid MPDFT
   OBKEEP PRELOAD PRERIP RIPPAGE 6;

/*---------------------------------------------*/
/* pgfmt01: */
/*---------------------------------------------*/
pageformat pfmt01 ;

/*---------------------------------------------*/
/* Layout for placing objects */
/*---------------------------------------------*/
/* type name no prerip, pages 1, 2, 3 */
/*---------------------------------------------*/
printline
   object npt02 obpage 1
   object npt02 obpage 2
   object npt02 obpage 3;

/*---------------------------------------------*/
/* pgfmt02: PRERIP ALL */
/*---------------------------------------------*/
pageformat pfmt02 ;

/*---------------------------------------------*/
/* Layout for placing objects PRERIP */
/*---------------------------------------------*/
/*---------------------------------------------*/
/* component id rip all, pages 4, 5, 6 */
/*---------------------------------------------*/
```
The OVERLAY command is used to identify an electronic overlay to be included in the print file. This function is similar to the SEGMENT command. A separate OVERLAY command is required for each overlay, including all overlay names that can be used with the OVERLAY VARIABLE keyword on PRINTLINE, LAYOUT, or XLAYOUT. A maximum of 254 OVERLAY commands (each of the 254 names must be unique) can be specified for each page format.

The OVERLAY commands are nested within the PAGEFORMAT command. For Traditional:

```plaintext
PAGEFORMAT
  [ TRCREF ]
  [ SEGMENT ]
  [ OVERLAY ]
  ...
  [ OVERLAY ]
```

For Record Format and XML:

```plaintext
PAGEFORMAT
  [ SEGMENT ]
  [ OVERLAY ]
  ...
  [ OVERLAY ]
```

An overlay can be requested in the following two ways:
• Place the overlay by using the OVERLAY subcommand on the PRINTLINE command (Traditional), the LAYOUT command (Record Format), or the XLAYOUT command (XML).
• Enter an Include Page Overlay (IPO) structured field in the line data. The name of the overlay on the IPO structured field must match exactly the overlay that is identified by this command. The IPO must specify a value of X’FFFFFF’ for the X and Y offset parameters if the overlay is to be placed relative to the current line position.

To include page overlays without using the IPO structured field, see the “PRINTLINE Command” on page 423.

OVERLAY
Identifies the overlay that is positioned on the page.

x-name
The user access name (external name) for the overlay. x-name can be unquoted or enclosed in quotation marks.

unquoted-name
An unquoted overlay name can be up to 6 characters. It is folded to uppercase, has an O1 prefix added to it, and is translated to EBCDIC code page 500 if necessary.

'quoted-name'
A quoted overlay name can be up to 8 characters. No translation or case folding is done.

Subcommands
These subcommands are used to specify whether to preload overlays, PreRip overlays, or both.

Notes:
1. The printer must support the preloading and preripping functions.
2. Mapping an overlay enhances throughput by allowing the printer to download an object once and use it on subsequent pages of the same print job or possibly on subsequent print jobs. Overlays are always mapped so it is not necessary for you to request mapping. Mapping an overlay provides sufficient performance for most applications.
3. Preloading an overlay consists of loading the object into printer memory before the print job is started. Preloading enhances throughput by removing downloading time from real time to the page build time.
4. Preripping enhances throughput by allowing the resources to be rasterized (RIPped) at the proper size and rotation when they are preloaded.

NOPRELOAD
Do not preload the overlay.

PRELOAD
Preload the overlay before the print job is processed.

NOPRERIP
Do not PreRip the overlay.

PRERIP
Prepare the overlay for printing by rasterizing it with its presentation rotation in regard to the media leading edge.
RIPROTA { 0 | 90 | 180 | 270 }
Specifies the overlay rotation in regard to the leading edge of the media. Up to four rotations can be specified.

Note: Many factors, such as media selection, media side media loading, media orientation, page rotation, and overlay area rotation affect the orientation of the overlay in regard to the media leading edge. Therefore, proper specification of this parameter might require visual inspection of physical output.

Note: The prefix O1 is not part of the six-character user-access name. The overlay name can be alphanumeric.

Example

In the following example, an overlay is defined and placed. The overlay is preripped in two orientations, 180 and 0.
PAGEDEF RipXm2 Replace Yes;
PAGEFORMAT pfl;
    OVERLAY ripit PRELOAD PRERIP OVROTATE 180,0 ;
PRINTLINE OVERLAY ripit;

PAGEDEF Command
Notes:
1. Record Format and XML only.
2. If UDType is not specified for Traditional and Record Format page definitions, the User Data Type is unspecified if not coded. This means that no data type information is added to the page definition. If UDType is not specified for XML page definitions, the User Data Type defaults to the type for the operating system when UDType is not coded. That is, it defaults to EBCDIC when on a z/OS operating system.

A page definition is a resource that defines how data is formatted on a logical page. When generated by PPFA, a page definition is stored as a resource in the page-definition library. You can use the page definition with the Record Format line data by using the PAGEDEF subcommands.

This command must be specified when you define a page definition. All of the PAGEDEF subcommands are optional; defaults are assumed.

For Traditional only: Values that are assigned within the subcommands or the default values become the values for any PAGEFORMAT subcommand not specified. REPLACE is not a PAGEFORMAT subcommand, so its default is not carried forward.
PAGEDEF
Identifies the page definition to be used with the print job.

name
Defines an alphanumeric name of 1 to 6 characters for the page definition. When page definitions are generated, PPFA assigns the prefix P1 to this name as the external resource name.

Subcommands

COMMENT

\[
\text{COMMENT} \quad qstring
\]

Specifies a user comment. This string comment is placed in the NOP structured field of the page definition.

qstring
Specifies a quoted set of strings 1 - 255 characters in total length.

WIDTH

\[
\text{WIDTH} \quad 8.3 \quad \text{IN}
\]

Defines the width of the logical page.

\( n \) A number with up to three decimal places is used. The width might vary according to the type of printer that is used. For more information, see your printer documentation. The default is 8.3 IN.

unit
Specifies a unit of measurement for the WIDTH subcommand. The choices are IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

HEIGHT

\[
\text{HEIGHT} \quad 10.8 \quad \text{IN}
\]

Defines the height of the logical page.

\( n \) A number with up to three decimal places is used. The height might vary according to the type of printer that is used. For more information, see your printer documentation. The default is 10.8 IN.

unit
Specifies a unit of measurement for the HEIGHT subcommand. The choices are IN, MM, CM, POINTS, and PELS.
Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

**LINEONE (Traditional only)**

```plaintext
  LINEONE x-pos y-pos
```

Specifies the values for the MARGIN and TOP parameters that are used in the POSITION subcommand of the PRINTLINE command.

**x-pos**

Specifies the offset from the left edge of the logical page (margin position). The valid options for `x-pos` are described in the SETUNITS command for the horizontal value.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

**y-pos**

Specifies the vertical offset from the top of the logical page (top line position). The valid options for `y-pos` are described in the SETUNITS command for the vertical value.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

**DIRECTION**

```plaintext
  DIRECTION ACROSS
  DIRECTION DOWN
  DIRECTION BACK
  DIRECTION UP
```

Specifies the print direction of the logical page. Not all printers can print in all print directions. For more information, see your printer documentation.

Note: Some printers have a different media origin and require different direction settings than most page printers. For printing in the landscape page presentation when you are using wide forms, the PRESENT subcommand must be specified on the FORMDEF command to produce readable output. Alternatively, if you have existing page definitions, the UP direction can be used in the page definition without changes to the form definition to produce the same result.

**ACROSS**

The page is printed with the characters added left to right in each line, and the lines added from the top to the bottom.

**DOWN**

The page is printed with the characters added to the page from top to bottom, and the lines added from the right to the left.

**BACK**

The page is printed with the characters added to the page from right to left, and the lines added from the bottom to the top.
UP The page is printed with the characters added to the page from bottom to top, and the lines added from the left to the right.

REPLACE

| REPLACE NO |
| REPLACE YES |

Specifies whether the page definition replaces an existing page definition with the same resource name in the library.

NO The page definition does not replace a page definition with the same resource name in the library.

If a page definition with the same resource name does not exist in the library, the page definition is stored.

YES If a page definition with the same resource name exists in the library, the page definition replaces it.

If a page definition with the same resource name does not exist in the library, the page definition is stored.

PELSPERINCH $n$

| PELSPERINCH $n$ |

Specifies the Logical Units in pels per inch for the page definition. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

$n$ Specifies an integer number 1 - 3,276, which determines the Logical Units in pels per inch.

Note: If the L-Units are not specified on the page definition, they are defaulted to 240 pels per inch.

In [Figure 134 on page 409][5], the page definition xmp01 specifies L-Units as 300 pels per inch. Because the PAGEFORMAT P1 does not specify L-Units, it inherits 300 pels per inch. PAGEFORMAT P2 does specify L-Units as 1200 pels per inch. The width and height in PAGEFORMAT P1 (7 in, 3 in) produces internal and structured field values of 2100 and 900, whereas in PAGEFORMAT P2 the same code produces values of 8400 and 3600 because of the difference in L-Units.
This keyword with parameters specifies the amount of space to be reserved at the top of the page. The default is 80% of the current line spacing.

This keyword with parameters specifies the amount of space to be reserved at the bottom of the page. Only PAGETRAILER data can be written into this area. If a graphic is not ended at the time information is being placed in the bottom margin, the graphic is ended before the bottom margin. The default is 0.

This keyword with parameters specifies the amount of space to be reserved at the left of the page. This keyword is to be used only with the DRAWGRAPHIC
commands. Although PPFA collects the left margin information, it uses the value only within PPFA to define an area. The value itself is not passed in the data stream. The default is 0.

**RIGHTMARGIN (Record Format and XML)**

This keyword with parameters specifies the amount of space to be reserved at the right of the page. This keyword is only to be used with the DRAWGRAPHIC commands. Although PPFA collects the right margin information, it uses the value only within PPFA to define an area. The value itself is not passed in the data stream. The default is 0.

**PAGECOUNT (Record Format and XML)**

This keyword specifies how the page counting is handled when you are switching between PAGEFORMATs.

**CONTINUE**

Page counting continues from the previous PAGEFORMAT - this is the default. The \( n \) value is only used on the first PAGEFORMAT in the job; otherwise, it is ignored. If this is the first PAGEFORMAT and no \( n \) value is specified, it defaults to one.

**STOP**

Page counting stops. Page count is captured from the previous PAGEFORMAT, but does not continue to count.

**RESUME**

Page counting continues from wherever it was the last time this PAGEFORMAT was called. The \( n \) value sets the value only the first time the PAGEFORMAT is invoked.

**RESET**

Page counting is reset to the value within the \( n \) value. If no \( n \) value is entered, then the page numbers are reset to one.

**CMP**

Count MO:DCA Pages option. Tells the print server to count any embedded MO:DCA pages in the page count.
CCP
Count Constant Pages options. Tells the print server to count any pages that have no variable data on them.

SOSIFONTS

The SOSIFONTS subcommand causes a Single-Byte Character Set (SBCS) font and a Double-Byte Character Set (DBCS) font to be mapped in a manner that allows the proper font switching when Shift-in and Shift-out control sequences are encountered in printed text.

sbc-font
A Single-Byte Character Set font. This font is selected by the print server when a Shift-In (SI) control byte is encountered in text that is presented.

dbcs-font
A Double-Byte Character Set font. This font is selected by the print server when a Shift-Out (SO) control byte is encountered in text that is presented.

Use one of these methods to use SOSI fonts in a page definition:

- In the PAGEDEF, use the FONT placement subcommand to specify both the SBCS and DBCS fonts to be used. To use this method, define both a single-byte and double-byte font with the FONT or DOFONT commands. Then, reference both fonts, separated by a comma, on the FONT subcommand on the FIELD, PRINTLINE, XLAYOUT, or LAYOUT commands. The single-byte font goes first.

Examples:
1. In a traditional page definition:
   ```
   Pagedef sosiP1 replace yes;
   FONT sb1 GT10 SBCS;
   FONT db1 M40F DBCS;
   PAGEFORMAT PF1;
   PRINTLINE POSITION 1 in 1.2 in FONT sb1,db1;
   ```
2. In a Record Format or XML page definition:
   ```
   Pagedef sosiP1 replace yes;
   FONT sb1 GT10 SBCS;
   FONT db1 M40F DBCS;
   PAGEFORMAT PF1;
   LAYOUT '11' POSITION 1 in 1.2 in FONT sb1,db1;
   ```

- In the PAGEDEF, use the PAGEFORMAT subcommand SOSIFONTS to ensure that a single-byte font is mapped first and a double-byte font is mapped second in the PAGEFORMAT. To use this method, code both a single-byte and double-byte font with the FONT command. Then, use the SOSIFONTS subcommand on the PAGEFORMAT command with the SBCS font coded first and the DBCS font coded next.

Examples:
1. In a traditional page definition:
2. In a Record Format or XML page definition:

```plaintext
Pagedef sosil1 replace yes;
    FONT sb1 GT10 SBCS;
    FONT db1 M40F DBCS;
    PAGEFORMAT PF1 SOSIFONTS sb1,db1;
    LAYOUT '11' POSITION 1 in 1.2;
```

**Note:** The SOSIFONTS subcommand can also be coded on the PAGEFORMAT command. Any PAGEFORMATs that do not code a SOSIFONTS subcommand inherit from the PAGEDEF.

- Specify fonts by using the CHARS JCL parameter and either no fonts in the PAGEDEF or the default PAGEDEF. Using this method, the first font that is defined in CHARS is a SBCS font and the second is a DBCS font.
- Traditional only: Use the TRCREF command to define the SBCS font as 0 and the DBCS font as 1. Do not specify a FONT subcommand on PRINTLINE, FIELD, and other commands when you are using this method. This method is used only with a Traditional page definition. For example:

```plaintext
Pagedef sosil1 replace yes;
    FONT sb1 GT10 SBCS;
    FONT db1 M40F DBCS;
    PAGEFORMAT PF1;
    TRCREF 0 FONT sb1;
    TRCREF 1 FONT db1;
    PRINTLINE;
```

**Note:** You cannot mix Data Object fonts (defined with the DOFONT command) with FOCA fonts (defined with the FONT command) in the page definition in any but the first method of specifying SOSI fonts.

### UDT yp e

- **UDType (EBCDIC or ASCII)**
  - **UDType**
  - **UDType (EBCDIC)**
  - **UDType (ASCII)**
    - **UDType (UTF8)**
    - **UDType (UTF16)**

#### Notes:

1. If UDTyp e is not specified for Traditional and Record Format page definitions, the User Data Type is unspecified if not coded. This means that no data type information is added to the page definition. If UDTyp e is not specified for XML page definitions, the User Data Type defaults to the type for the operating system when UDTyp e is not coded. That is, it defaults to EBCDIC when on a z/OS operating system or ASCII when on an AIX® or Windows operating system.
This subcommand identifies the encoding of your data. If UDTType is not coded on the PAGEDEF, it defaults to either ASCII or EBCDIC to match the operating system. For example, if PPFa is run on a z/OS operating system and UDTType is not coded it defaults to EBCDIC.

UDTType on the PAGEDEF command is used for several things:
1. Allow PPFa to translate fixed text to the specified UDTType from either ASCII or EBCDIC according to the operating system on which the PPFa compile is done.
2. Set the default for all DOFONT (Data Object Font) commands so you do not need to code UDTType on each DOFONT command.
3. Pass encoding information to the printer for converting non-UTF16 user data to UTF16 when you are using a DOFONT command. (TrueType is an example of a DOFONT).
4. Allows PSF or ACIF to know to look for a Byte Order Mark (BOM) when your data type is UTF8 or UTF16 and contains a BOM.

If the data does not match the operating system data type, PPFa translates the following constant page definition data to the encoding specified by UDTType:
- FIELD command text (all page definitions)
- CONDITION text (all page definitions)
- LAYOUT command ‘record ID’ (Record Format page definition only)
- LAYOUT command delimiter (Record Format page definition only)
- XLAYOUT command start tags (XML page definition only)
- XLAYOUT command delimiter (XML page definition only)
- DEFINE QTAG command start tags (XML page definition only)
- FIELD attribute names (XML page definition only)

Notes:
1. For data with a Byte Order Mark (BOM), you must specify UDTType, and the BOM must be the first 2 bytes (UTF16) or 3 bytes (UTF8) in the first line data record following any CC or TRC bytes. A BOM in the data is required if the data is UTF16 Little Endian.
2. If the UDTType parameter is specified, all of the user data that is processed by the page definition must be of that data type. Having data that is not of the specified encoding type might lead to improper translation of that data. For example, the text in a CONDITION statement is not matched to the data.
3. If UDTType is not coded on the PAGEDEF, the UDTType on the DOFONT command or TYPE on the FONT command determines the translation, but only for the data placed by that font.
4. If you have multiple data type encodings in a data file, you must not code UDTType on the page definition. Instead, for Data Objects Fonts, code UDTType on the DOFONT command for the fonts that place the individual fields or records. For regular FOCA fonts, you use fonts of the type that match the data for the individual fields or records.
5. If you use ACIF to generate your print document and the NEWLINE ENCODING value does not agree with the UDTType subcommand on the PAGEDEF command, ACIF issues a warning message but continues processing the file.
6. The UDTType coded on the PAGEDEF is inherited by the DOFONT if NONE is coded on the DOFONT command. And, if no UDTType is coded on either the PAGEDEF or the DOFONT, the DOFONT defaults to the operating system encoding.
7. The UDType coded or not coded on the PAGEDEF does not affect the FONT command inheritance of data type, or in any way except to provide translation for fixed text that is placed by the FONT command.

8. If UDType is not the same as the UDType coded on the DOFONT command, PPFA issues an error message and no page definition is generated.

9. To use multiple font mappings for a line in ASCII, UTF8, or UTF16 you must use the FIELD command, since automatic font switching for single and double byte text is only done for EBCDIC data.

10. SBCS, DBCS, TYPE, and UDTYPE are different parameters that can affect fonts. TYPE indicates that an ASCII or EBCDIC font is defined for printing. UDTYPE, which is not on the FONT command, indicates that the type of the user’s data is ASCII or EBCDIC. SBCS and DBCS indicate what type of character set the defined font uses, either single byte (SBCS) or double byte (DBCS).

11. If UDTYPE is coded on the PAGEDEF and an ID name or text is explicitly defined with another type, an error results and the page definition is not generated. In the next example:

```plaintext
PAGEDEF cmr1is replace yes UDTYPE ASCII;
FONT varb gt10;
SETUNITS LINESP .25 in;
PAGERFORMAT rept1 TOPMARGIN .25 in BOTMARGIN .25 in;
LAYOUT E'startpage' BODY NEWPAGE POSITION .25 in NEXT font varb;
```

PAGEDEF with UDTYPE ASCII and LAYOUT with E'startpage', the ASCII type does not match the 'E'EBCDIC type.

---

**EBCDIC**

Single-byte EBCDIC code page 500

**ASCII**

Single-byte ASCII code page 819

**UTF8**

Unicode encoding form UTF-8 toleration mode (surrogates allowed)

**UTF16**

Unicode encoding form UTF-16

**Note:** The PAGEDEF is created in UTF-16BE (Big Endian). If the data is in UTF16LE, PSF translates it to UTF-16BE before processing.

---

**RECIDLEN**

```
    RECIDLEN 10
    RECIDLEN n
```

Specifies the length of the Record Descriptor ID in bytes. This is also known as the LAYOUT name. If the RECIDLEN parameter is not coded on a PAGEFORMAT command, it inherits the value from the specified or default value on the page definition. If the RECIDLEN parameter is not coded on a PAGEDEF command, the default length is 10 bytes.

**Notes:**

1. Use the RECIDLEN keyword on Record Format page definition only.
2. This parameter can be used only in a Record Format page definition.
Specifies that the 'record ID' on the LAYOUT command is to be \( n \) bytes long. The allowable value of \( n \) is 1 - 250. UTF-16 data characters are 2 bytes long allowing up to 125 UTF-16 characters. Any 'record ID' on a LAYOUT command that is less than this length is padded to the specified length with blanks of the type that is specified or defaulted in the UDTType subcommand on the PAGEDEF command. A 'record ID' that is longer than \( n \) is flagged as an error by PPFA and no page definition is generated.

Note: If the User Data Type (UDType) is UTF16 and this number is odd, it is rounded up to the next even number.

Code Example

In the following example, User Data Type UTF16 and RECIDLLEN 24 are specified on the PAGEDEF command and the RECIDLLEN 26 is specified on the second PAGEFORMAT command (pf2).

For the two page formats pf1 and pf2:
1. pf1 inherits a RECIDLLEN of 24 bytes from the page definition, and the User Data Type for the entire page definition is UTF-16.
   a. The LAYOUT name 'Long Name 1' is translated to UTF-16 and padded to 24 bytes with UTF-16.
   b. The delimiter '/' on the LAYOUT is translated to UTF-16.
   c. The FIELD command text 'abcd' is translated to UTF-16.
2. pf2 specifies a RECIDLLEN of 26 bytes and gets UDTType UTF-16 from the page definition.
   a. The LAYOUT name 'Long Name 2' is translated to UTF-16 and padded to 26 bytes with UTF-16.
   b. The CONDITION command text 'ABCDEFGH' is translated to UTF-16.

Note: The field length of the 8-character string is 16 bytes because each character is 2 bytes long.

```
PAGEDEF xmp1  UDType UTF16  RECIDLLEN 24  REPLACE yes;
  FONT comp a075nc TYPE UNICODE
  FONT comp2 a075bg TYPE UNICODE

PAGEFORMAT pf1;
  LAYOUT 'Long Name 1' DELIMITER '/' Position 2 1 FONT comp;
  FIELD TEXT 'abcd' Position 2.5 1.5,

PAGEFORMAT pf2 RECIDLLEN 26;
  LAYOUT 'Long Name 2'  POSITION 2 1 FONT comp2;
  CONDITION cn1 START 13 Length 16 WHEN EQ 'ABCDEFGH' NULL Pageformat pf1;
```

PAGEFORMAT Command
Notes:

1  Record Format and XML only.

Page formats are subsets of page definitions. If you want to use more than one set of specifications to format a page within a single print job, you must use more than one page format. To change page formats, use conditional processing or insert an Invoke Data Map structured field in your print file. (Page formats are known to the print server as data maps.) If you do not use conditional processing or if you do not insert an Invoke Data Map structured field, the print server uses only the first page format in the page definition. Page formats are placed in the page definition in the order in which they are generated.

PAGEFORMAT subcommands have no fixed defaults. The entire PAGEFORMAT command and all of its subcommands can assume defaults. If any PAGEFORMAT subcommand is omitted, its value is selected from the corresponding subcommand in the governing PAGEDEF command.
This command can be omitted for the first page format in a page definition if only one page format is used. If omitted, PPFA assigns a page format name by using the page-definition name, including the P1 prefix.

**PAGEFORMAT name**

Specifies an alphanumeric name of 1 - 8 characters. This name must be unique within the page definition.

The following subcommands are used for each page format. They can be issued in the same way as in a page definition. Values that are specified in the PAGEDEF subcommands are used if any of the following subcommands are not defined within a page format.

**Subcommands**

**WIDTH**

```
WIDTH n unit
```

Defines the width of the logical page.

- **n** A number with up to three decimal places is used. The width might vary according to the type of printer that is used. For more information, see your printer documentation.

- **unit** Specifies a unit of measurement for the WIDTH subcommand. The choices are IN, MM, CM, POINTS, or PELS.

**Note:** If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

**HEIGHT**

```
HEIGHT n unit
```

Defines the height of the logical page.

- **n** A number with up to three decimal places is used. The height might vary according to the type of printer that is used. For more information, see your printer documentation.

- **unit** Specifies a unit of measurement for the HEIGHT parameter. The choices are IN, MM, CM, POINTS, or PELS.

**Note:** If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

**LINEONE (Traditional)**

```
LINEONE x-pos y-pos
```

Chapter II. Page Definition Command Reference 417
Specifies the values for the MARGIN and TOP parameters that are used in the POSITION subcommand of the PRINTLINE command.

*x-pos*

Specifies the offset from the left edge of the logical page (margin position). The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

**Note:** If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

*y-pos*

Specifies the offset from the top of the logical page (top line position). The valid options for *y-pos* are described in the SETUNITS command for the vertical value.

**Note:** If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

### DIRECTION

<table>
<thead>
<tr>
<th>DIRECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACROSS</td>
</tr>
<tr>
<td>DOWN</td>
</tr>
<tr>
<td>BACK</td>
</tr>
<tr>
<td>UP</td>
</tr>
</tbody>
</table>

Specifies the print direction of the logical page. Not all printers can print in all print directions. For more information, see your printer documentation.

**Note:** Some printers have a different form origin and require different direction settings than most page printers. For printing in the landscape page presentation when you are using wide forms, the PRESENT subcommand must be specified on the FORMDEF command to produce readable output. Alternatively, if you have existing page definitions, the UP direction can be used in the page definition without changes to the form definition to produce the same result.

**ACROSS**

The page is printed with the characters added to the page from left to right, and the lines added from the top to the bottom.

**DOWN**

The page is printed with the characters added to the page from top to bottom, and the lines added from the right to the left.

**BACK**

The page is printed with the characters added to the page from right to left, and the lines added from the bottom to the top.

**UP**

The page is printed with the characters added to the page from bottom to top, and the lines added from the left to the right.

For Record Format and XML DIRECTION effects the meaning of the following new margin parameters.

- If the DIRECTION is ACROSS, then TOPMARGIN refers to the margin in the short end of the physical page where the tops of the characters point toward that same short end.
- If the DIRECTION is DOWN, then TOPMARGIN refers to the margin in the long end of the physical page where the tops of the characters point toward that same long end.

**PELSPERINCH \( n \)**

Specifies the Logical Units in pels per inch for the page format. Use the PELSPERINCH parameter to tell PPFA the pel resolution of your printer to generate more exact object placements.

\( n \)  Specifies an integer number 1 - 3,276, which determines the Logical Units in pels per inch.

**Note:** If the L-Units are not specified on the page format, they are inherited from the page definition that contains the page format. See Figure 134 on page 409.

**TOPMARGIN (Record Format and XML)**

This keyword with parameters specifies the amount of space to be reserved at the top of the page. The default is 80% of the current line spacing.

**BOTMARGIN (Record Format and XML)**

This keyword with parameters specifies the amount of space to be reserved at the bottom of the page. The default is 0.

**LEFTMARGIN (Record Format and XML)**

This keyword with parameters specifies the amount of space to be reserved at the left of the page. This is only used with the DRAWGRAPHIC commands.
Although PPFA collects the left margin information, the value is used only within PPFA to define an area. The value itself is not passed in the datastream. The default is 0.

**RIGHTMARGIN (Record Format and XML)**

```
RIGHTMARGIN n
- IN
- MM
- CM
- POINTS
- PELS
```

This keyword with parameters specifies the amount of space to be reserved at the right of the page. This is only to be used with the DRAWGRAPHIC commands. Although PPFA collects the right margin information, it uses the value only within PPFA to define an area. This value itself is not passed in the datastream. The default is 0.

**PAGECOUNT (Record Format and XML)**

```
PAGECOUNT
- CONTINUE
- STOP
- RESUME
- RESET
- CMP
- CCP
```

This keyword specifies how the page counting is handled when you are switching between page formats.

**CONTINUE**

Page counting continues from the previous page format - this is the default. The $n$ value is used only on the first PAGEFORMAT in the job; otherwise, it is ignored. If this is the first PAGEFORMAT and no $n$ value is specified, it defaults to one.

**STOP**

Page counting stops. Page count is captured from the previous page format, but does not continue to count.

**RESUME**

Page counting continues from wherever it was the last time the page format was called. The $n$ value sets the value only the first time page format is invoked.

**RESET**

Page counting is reset to the value within the $n$ value. If no $n$ value is entered, then the page numbers are reset to one.

**CMP**

Count MO:DCA Pages option. Tells the print server to count any embedded MO:DCA pages in the page count.
**CCP**

Count Constant Pages options. Tells the print server to count any pages that have no variable data on them.

**SOSIFONTS**

The **SOSIFONTS** subcommand causes a Single-Byte Character Set (SBCS) font and a Double-Byte Character Set (DBCS) font to be mapped in a manner that allows the proper font switching when Shift-in and Shift-out control sequences are encountered in printed text.

**sbcs-font**

A Single-Byte Character Set font. This font is selected by the print server when a Shift-In (SI) control byte is encountered in text that is presented.

**dbcs-font**

A Double-Byte Character Set font. This font is selected by the print server when a Shift-Out (SO) control byte is encountered in text that is presented.

Use one of these methods to use SOSI fonts in a page definition:

- In the PAGEDEF, use the FONT placement subcommand to specify both the SBCS and DBCS fonts to be used. To use this method, define both a single-byte and double-byte font with the FONT or DOFONT commands. Then, reference both fonts, separated by a comma, on the FONT subcommand on the FIELD, PRINTLINE, or LAYOUT commands. The single-byte font goes first.

**Examples:**

1. In a traditional page definition:

   ```
   Pagedef sosiP1 replace yes;
   FONT sb1 GT10 SBCS;
   FONT db1 M40F DBCS;
   PAGEFORMAT PF1;
       PRINTLINE POSITION 1 in 1.2 in FONT sb1,db1;
   ```

2. In a Record Format or XML page definition:

   ```
   Pagedef sosiP1 replace yes;
   FONT sb1 GT10 SBCS;
   FONT db1 M40F DBCS;
   PAGEFORMAT PF1;
       LAYOUT '11' POSITION 1 in 1.2 in FONT sb1,db1;
   ```

- In the PAGEDEF, use the PAGEFORMAT subcommand **SOSIFONTS** to ensure that a single-byte font is mapped first and a double-byte font is mapped second in the PAGEFORMAT. To use this method, code both a single-byte and double-byte font with the FONT command. Then, use the SOSIFONTS subcommand on the PAGEFORMAT command with the SBCS font coded first and the DBCS font coded next.

**Examples:**

1. In a traditional page definition:
2. In a Record Format or XML page definition:

Pagedef sosIL1 replace yes;
FONT sb1 GT10 SBCS;
FONT db1 M40F DBCS;
PAGEFORMAT PF1 SOSIFONTS sb1,db1;

LAYOUT 'll' POSITION 1 in 1.2;

**Note:** The SOSIFONTS subcommand can also be coded on the PAGEFORMAT command. It is inherited on any PAGEFORMATs that do not code a SOSIFONTS subcommand.

- Define fonts by using the CHARs JCL parameter and either no fonts in the PAGEDEF or the default PAGEDEF. Using this method, the first font that is defined in the CHARs is an SBCS font and the second is a DBCS font.

- Traditional only: Use the TRCREF command to define the SBCS font as 0 and the DBCS font as 1. Do not specify a FONT subcommand on PRINTLINE, FIELD, and other commands when you are using this method. This method is used only with a Traditional page definition. For example:

Pagedef sosIL1 replace yes;
FONT sb1 GT10 SBCS;
FONT db1 M40F DBCS;
PAGEFORMAT PF1;
TRCREF 0 FONT sb1;
TRCREF 1 FONT db1;
PRINTLINE;

**Note:** You cannot mix Data Object fonts (defined with the DOFONT command) with FOCA fonts (defined with the FONT command) in the page definition in any but the first method of specifying SOSI fonts.

For additional information about using SOSI characters, see the *AFP Programming Guide and Line Data Reference.*

**RECIDLEN**

```
RECIDLEN
RECIDLEN 10
RECIDLEN n
```

Specifies the length of the Record Descriptor ID in bytes. This is also known as the *LAYOUT name*. If the RECIDLEN parameter is not coded on a PAGEFORMAT command, it inherits the value from the specified or default value on the page definition. If the RECIDLEN parameter is not coded on a PAGEDEF command, the default length is 10 bytes.

**Notes:**

1. Use the RECIDLEN keyword on Record Format page definition only.
2. This parameter can be used only in a Record Format page definition.

$n$ Specifies that the ‘record ID’ on the LAYOUT command is to be $n$ bytes long. The allowable value of $n$ is 1 - 250. UTF-16 data characters are 2 bytes long allowing up to 125 UTF-16 characters. Any ‘record ID’ on a
LAYOUT command that is less than this length is padded to the specified length with blanks of the type that is specified or defaulted in the UDType subcommand on the PAGEDEF command. A 'record ID' that is longer than \( n \) is flagged as an error by PPFA and no page definition is generated.

**Note:** If the User Data Type (UDType) is UTF16 and this number is odd, it is rounded up to the next even number.

**Code Example**

In the following example, User Data Type UTF16 and RECIDLEN 24 are specified on the PAGEDEF command and the RECIDLEN 26 is specified on the second PAGEFORMAT command (pf2).

For the two page formats pf1 and pf2:

1. pf1 inherits a RECIDLEN of 24 bytes from the page definition, and the User Data Type for the entire page definition is UTF-16.
   a. The LAYOUT name 'Long Name 1' is translated to UTF-16 and padded to 24 bytes with UTF-16.
   b. The delimiter '/' on the LAYOUT is translated to UTF-16.
   c. The FIELD command text 'abcd' is translated to UTF-16.
2. pf2 specifies a RECIDLEN of 26 bytes and gets UDType UTF-16 from the page definition.
   a. The LAYOUT name 'Long Name 2' is translated to UTF-16 and padded to 26 bytes with UTF-16.
   b. The CONDITION command text 'ABCDEFGH' is translated to UTF-16.

**Note:** The field length of the 8-character string is 16 bytes because each character is 2 bytes long.

```plaintext
PAGEDEF xmp1 UDType UTF16 RECIDLEN 24 REPLACE yes;

FONT comp a075nc TYPE UNICODE
FONT comp2 a075bg TYPE UNICODE

PAGEFORMAT pf1;
LAYOUT 'Long Name 1' DELIMITER '/' Position 2 1 FONT comp;
FIELD TEXT 'abcd' Position 2.5 1.5,

PAGEFORMAT pf2 RECIDLEN 26;
LAYOUT 'Long Name 2' POSITION 2 1 FONT comp2;
CONDITION cn1 START 13 Length 16
WHEN EQ 'ABCDEFGH' NULL Pageformat pf1;
```

---

**PRINTLINE Command**

- PRINTLINE
  - REPEAT \( n \)
  - FIELD
  - CHANNEL \( n \)
  - FONT name1, \( \ldots \) name2
  - PRINTDATA YES
  - PRINTDATA NO

- POSITION SAME
  - NEXT

- POSITION MARGIN
  - SAME or =
  - x-pos

- RELATIVE
  - TOP
  - NEXT
  - SAME or =
  - y-pos

---

Chapter 11. Page Definition Command Reference 423
The PRINTLINE command specifies the printing of one data record on a line. If a formatted printline is to be printed, one or more FIELD commands must follow the governing PRINTLINE command; at least one is required. If one FIELD command is not used, field processing is not done and the unformatted data is printed.

Note: The PRINTLINE command defines a traditional page definition and cannot be mixed with LAYOUT commands that define Record Formatting page definitions or XLA\textunderscore YOUT commands that define XML page definitions.

**Subcommands**

**REPEAT**

Specifies the number of printlines that are to be printed on a logical page. The direction and font that is specified within this printline applies to all lines printed. By using this command, you do not have to write specifications for each line.

Note: If the REPEAT subcommand is omitted, only one line is printed for this PRINTLINE command.

\( n \)   This value specifies the number of printlines for a logical page; the maximum value is 65,535.

**REPEAT 0**

Not valid.

**REPEAT 1**

Only one line is printed.

If the CHANNEL or POSITION subcommands are specified within this PRINTLINE command, they apply only to the first line.

If this PRINTLINE is followed by several FIELD commands, the related field controls are also repeated.

**FIELD**

Specifies that fields that are associated with repetitions of this PRINTLINE are to be positioned based on the first instance of the same field.

This parameter has no effect in fields with the same direction as the PRINTLINE of which they are a part.

This parameter specifies that the direction of repetition for a field is the direction of the first instance of the field plus 90°. Therefore, every field of an ACROSS PRINTLINE is repeated down the page, regardless of the direction of the FIELD.
**LINE**

Specifies that fields that are associated with repetitions of this printline are to be positioned based on the repetition of the PRINTLINE itself.

This parameter has no effect in fields with the same direction as the PRINTLINE of which they are a part.

This parameter specifies that the direction of repetition for a field is the direction of the associated PRINTLINE plus 90°. Therefore, every field of an ACROSS PRINTLINE is repeated down the page, regardless of the direction of the FIELD.

**CHANNEL** \( n \)

```
<CHANNEL—n>
```

Used to specify line spacing, skipping within a logical page, or page ejection (skipping to a new page). This subcommand is equivalent to the Forms Control Buffer (FCB) channel.

\( n \) The range of channels is 1 - 12. The channels correspond to carriage-control characters in the data. No default exists.

**FONT**

```
<FONT—name1 , —name2>
```

Defines the font to be used for the printline.

\( name1 \)

Specifies the name of a font that is used to print the data. This font must be defined in a previous FONT command in the page definition.

If Shift-Out, Shift-In (SOSI) processing is used, \( name1 \) must be the single-byte font.

\( name2 \)

Specify only when you are using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the printline. \( name2 \) must be the double-byte font.

**Notes:**

1. If this subcommand is not specified and TRC (Table Reference Character) bytes are specified in the print data, the print server uses the font indicated by the TRC byte. Otherwise, the print server selects a default font.

2. For ASCII, UTF8, or UTF16 the entire PRINTLINE command must be one font. To use multiple font mappings for a line in ASCII, UTF8, or UTF16, you must use the FIELD command.

**PRINTDATA**

```
<PRINTDATA—YES , —NO>
```
Specifies whether the line of data that is associated with the current PRINTLINE must be printed. The PRINTDATA subcommand is useful when the data stream is interspersed with lines of comments, blank lines, or lines without data that are not meant to be printed.

**YES**

Specifies that the data for the current PRINTLINE is printed. YES is the default.

**NO**

Specifies that the data for the current PRINTLINE is not printed.

**Note:** Any FIELD command that is associated with a PRINTLINE that specifies PRINTDATA NO is ignored and an error message is issued.

The default position for a YES command that specifies PRINTDATA NO is position same. The following example is for PRINTLINE NO.

```
PAGEDEF xmp01;
  SETUNITS LINESP 1 LPI;

  PRINTLINE;
  PRINTLINE PrintData NO;
  PRINTLINE PrintData yes;
  PRINTLINE;
  PRINTLINE Segment X PrintData NO Overlay Y Position Same Next;
  PRINTLINE PrintData yes;
```

The LINESP parameter specifies that one line per inch is to be printed.

1. The first line of data is read and printed.
2. The second line of data is read, but not printed.
3. The third line of data is read and printed 1 inch down from the first line.
4. The fourth line of data is read and printed 1 inch down from the third line.
5. The fifth line of data is read, but not printed.
   - The segment X is printed.
   - The overlay Y is printed.
6. The sixth line of data is read and printed 2 inches down from the fourth line.

**Note:** The data line 2 was not printed and did not affect the positioning of the lines that followed. Line 3 was positioned as though line 2 did not exist.

**POSITION**

```
POSITION--SAME--NEXT--
  POSITION--MARGIN--SAME or RELATIVE--x-pos
  TOP--NEXT--SAME or = y-pos
```

Specifies the starting position of the printline in the printout.

**x-pos**

Specifies the horizontal offset from the left side of the logical page. The value is a number with up to three decimal places. The valid options for x-pos are described in the SETUNITS command for the horizontal value.
MARGIN
Specifies that the line starts at the position that is specified as the horizontal (x) value in the previous LINEONE subcommand within the page definition.

SAME
Specifies this line starts at the same horizontal offset position as the previous printline. If applied to the first printline of a logical page, the horizontal position is 0, which is the default.

= Alternative for SAME.

RELATIVE
Specifies that the following vertical position value is to be processed as a relative value. The printline is positioned relative to the last printline placed on the page.

If a set of printlines were skipped over in the page definition because of a skip-to-channel carriage control, and the new active printline contains a relative vertical position, the output line is positioned relative to the location of the last line that is printed on the page.

Notes:
1. If both TOP and RELATIVE are requested for the Y position value, the RELATIVE request is ignored.
2. When you are using RELATIVE positioning, PPFA does not flag off-the-page conditions for the position of a printline or for any overlays, segments, or objects placed relative to that printline. Printlines that fall outside the bounds of the logical page are flagged by the print server at run time.
3. When you are specifying RELATIVE, use the minus sign to indicate any negative values for the PRINTLINE vertical position; you can use the plus sign to indicate positive values. If no sign is used, a positive value is assumed.
4. The DIRECTION for a relative printline must be ACROSS. Fields that are associated with a relative printline must have the same DIRECTION as the printline and must match the PAGEFORMAT DIRECTION.
5. If RELATIVE is specified with SAME or = as the y value, the relative value in the printline is +0.
6. Relative positioning is allowed on a PRINTLINE command only if the PRINTLINE and all its associated FIELD commands are formatted to print in the same direction as the PAGEFORMAT. That is, the DIRECTION parameter in the PRINTLINE and any associated FIELD commands must specify (or default to) ACROSS. The DIRECTION in the PAGEFORMAT or PAGedef command can be any allowable value: ACROSS, DOWN, BACK, or UP.
7. The PRINTLINE command in which relative positioning is used can specify a CHANNEL parameter. The n value that is specified for the CHANNEL parameter cannot be used for any other PRINTLINE in the same PAGEFORMAT:

```plaintext
setunits linesp 6 lpi;
PAGedef rel9 replace yes
   direction across width 8.5 in height 11.0 in;
PRINTLINE channel 1 repeat 7 position 0 IN 1.0 IN;
/* The fields will be placed at +120 pels, +24 pels (next) */
/* and +48 pels (.20 IN) from lines previously placed on page */
```

Page Printer Formatting Aid: User’s Guide
setunits linesp 10 lpi;
PRINTLINE channel 2 repeat 2 position 0 relative next;
FIELD START 1 LENGTH 3 position 0 IN .5 IN;
FIELD START 4 LENGTH 3 position 0 IN next;
FIELD START 7 LENGTH 3 position current .20 IN;

**y-pos**
Specifies the vertical offset from the top side of the logical page. The value
options for **y-pos** are described in the SETUNITS command for the vertical
value.

**TOP**
Specifies that the printline is placed in the position that is specified as the
vertical (y) value in the previous LINEONE subcommand within the page
definition.

**NEXT**
Specifies the PRINTLINE is to be positioned down (on the logical page)
one line (as defined in the LINESP subcommand of the last SETUNITS
command) from the previous PRINTLINE. The LINESP subcommand of
the SETUNITS command establishes the distance from one line to the next.

When NEXT is specified for the first PRINTLINE of a logical page, the
starting position of the line is one line down from the top of the logical
page, which is the default.

**Note:** The down direction is determined by the direction of the logical
page (as specified in the page format), not the printline direction.
Therefore, NEXT is useful in ACROSS printlines.

**SAME**
Specifies this printline starts at the same vertical position as the previous
printline. If applied to the first printline of a logical page, the horizontal
position is 0, which is the default.

**COLOR**

<table>
<thead>
<tr>
<th>COLOR—colorname</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB—rvalue—gvalue—bvalue</td>
</tr>
<tr>
<td>HIGHLIGHT—hvalue</td>
</tr>
<tr>
<td>CMYK—cvalue—mvalue—yvalue—kvalue</td>
</tr>
<tr>
<td>CIELAB—lvalue—clvalue—c2value</td>
</tr>
</tbody>
</table>

Specifies an OCA or defined color for the text of this field. This subcommand
is recognized only by printers that support multiple-color printing. See your
printer publication for information about the colors that can be printed. See
Chapter 8, “AFP Color Management,” on page 163 for more information about
using color.

**colorname**
Values for **colorname** can be a defined color (see “DEFINE COLOR
Command” on page 280), or an OCA colorname. Values for OCA colornames
are:

- NONE
- DEFAULT
The color choices depend on the printer.

If you do not enter one of these colors, the default color for that printer is used. NONE is the color of the medium. DEFAULT is the printer default color.

**Note:** In some printer manuals, the color turquoise (TURQ) is called cyan, and the color pink (PINK) is called magenta.

PPFA supports the following synonyms:
- CYAN for TURQ
- DARKCYANN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

**color models**

Specifies the color of print for this field that is supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

**RGB rvalue gvalue bvalue**

Three RGB integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values can be specified as a percentage 0 - 100.

**Note:** An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

**HIGHLIGHT hvalue COVERAGE cvalue BLACK bvalue**

Indicates the highlight color model. Highlight colors are device-dependent.

You can use an integer within the range of 0 - 65535 for the hvalue.

**Note:** An hvalue of 0 indicates that no default value is defined; therefore, the default color of the presentation device is used.
COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 - 100 for the cvalue. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

Note: Fractional values are ignored. If COVERAGE is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 - 100 for the bvalue. The amount of black shading that is applied depends on the COVERAGE percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

Note: If BLACK is not specified, a value of 0 is used as a default.

CMYK cvalue mvalue yvalue kvalue
Defines the cyan/magenta/yellow/black color model. cvalue specifies the cyan value. mvalue specifies the magenta value. yvalue specifies the yellow value. kvalue specifies the black value. You can use an integer percentage within the range of 0 - 100 for any of the CMYK values.

CIELAB Lvalue (-) c1value (-) c2value
Defines the CIELAB model. Use a range of 0.00 - 100.00 with Lvalue to specify the luminance value. Use signed integers from -127 to 127 with c1value and c2value to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. No defaults exist for the subvalues.

Notes:
1. Do not specify both an OCA color with the COLOR subparameter and an extended color model on the same FIELD or PRINTLINE command. The output is device-dependent and might not be what you expect.
2. Do not specify two extended COLOR subcommands on the same FIELD or PRINTLINE command.

OVERLAY

OVERLAY Xname VARIABLE LENGTH n START n relx rely OVRotate 0 90 180 270

Specifies the name of an overlay that is to be positioned relative to the location specified in the PRINTLINE command in which the OVERLAY subcommand was named. The PAGEFORMAT OVERLAY command can contain the named overlays. The maximum number of overlays that are specified for a PAGEFORMAT including the PRINTLINE OVERLAY subcommand is 254.

The OVERLAY can be identified by specifying a name (xname) or by getting the name from the input data record (use VARIABLE command).
Xname
The user access name (external name). It can be unquoted or quoted with
descriptor tags, indicating the data type (for example, ASCII) of the data in
the field.

unquoted-name
An unquoted external name can be up to 6 characters. It is folded to
uppercase, has an O1 prefix added to it, and is translated to EBCDIC
code page 500 if necessary.

quoted-name with no data tag
A quoted external name can be up to 8 characters. No translation is
done. It is the data type (EBCDIC or ASCII) as dictated by the
operating system. If not 8 bytes long, it is padded on the right with
EBCDIC or ASCII blanks.

C’quoted-name’
This quoted external name can be up to 8 characters. No translation is
done. It is the data type (EBCDIC or ASCII) as dictated by the
operating system. If not 8 bytes long, it is padded on the right with
EBCDIC or ASCII blanks.

E’quoted-name’
This quoted external name can be up to 8 characters. It is translated, if
necessary, to EBCDIC and padded with EBCDIC blanks if it isn’t 8
bytes long.

A’quoted-name’
This quoted external name can be up to 8 characters. It is translated, if
necessary, to ASCII and padded with ASCII blanks if it isn’t 8 bytes
long.

X’hex-digit-pairs’
This quoted external name can be up to 8 characters (16 hexadecimal
digits). No translation is done. If fewer than 8 characters are coded, the
name is padded on the right with blanks of the operating system type
where the page definition is generated (ASCII on AIX and Windows;
otherwise, EBCDIC). The user can avoid the padding by coding all 16
hexadecimal digits.

VARIABLE
Indicates that the actual name of the overlay, including the O1 prefix, is
read from the data record. The Variable-Name-Locator field specifies where
in the data to get the name.

Notes:
1. Any overlay that is to be included in this manner must be defined in
the PAGEFORMAT by using the OVERLAY command. Any overlay
that is included but not defined causes a runtime print error for a
missing MPO structured field, for example APS263I.
2. If you specify VARIABLE for the OVERLAY name and don’t want to
print the name, then you must have at least one field command, or
code PRINTDATA NO on the PRINTLINE command.

START n
The starting position in the data record to get the overlay name. The
first data byte position of the input record is 1. If START is not coded,
1 is assumed.
LENGTH \( n \)
Length of field. Specifies the number (\( n \)) of bytes to process from the data record, beginning with the position specified in START. The maximum length is 8.

OVROTATE \{0|90|180|270\}
Specifies the rotation of the placed overlay in regard to the \( x \)-axis of the page. See “FORMDEF Command” on page 234 for an OVROTATE example, which is presented in the FORMDEF description.

SEGMENT

Specifies the placement of a segment relative to the location specified in the PRINTLINE command in which the SEGMENT subcommand was named. The PAGEFORMAT SEGMENT command can contain the named segments. The maximum number of segments that are specified for a PAGEFORMAT including the PRINTLINE SEGMENT subcommand is 127.

The SEGMENT can be identified by specifying a name (\( Xname \)) or by getting the name from the input data record by using VARIABLE Variable-Name-Locator.

\( Xname \)
Specifies the user-access name as defined in the SEGMENT command. It can be unquoted or quoted with descriptor tags within indicate the data type of the data in the field.

\( \text{unquoted-name} \)
An unquoted external name can be up to 6 characters. It is folded to uppercase, have an O1 prefix added to it, and be translated to EBCDIC code page 500 if necessary.

\( \text{quoted-name with no data tag} \)
A quoted external name can be up to 8 characters. No translation is done. It is the data type (EBCDIC or ASCII) as dictated by the operating system. If not 8 bytes long, it is padded on the right with EBCDIC or ASCII blanks.

\( \text{C'quoted-name'} \)
A quoted external name can be up to 8 characters. No translation is done. It is the data type (EBCDIC or ASCII) as dictated by the operating system. If not 8 bytes long, it is padded on the right with EBCDIC or ASCII blanks.

\( \text{E'quoted-name'} \)
This quoted external name can be up to 8 characters. It is translated, if necessary, to EBCDIC and padded with EBCDIC blanks if it isn’t 8 bytes long.

\( \text{A'quoted-name'} \)
This quoted external name can be up to 8 characters. It is translated, if necessary, to ASCII and padded with ASCII blanks if it isn’t 8 bytes long.
'hex-digit-pairs'

This quoted external name can be up to 8 characters (16 hexadecimal characters). No translation is done. If fewer than 8 characters are coded, the name is padded on the right with blanks of the operating system type where the page definition was generated (ASCII on AIX and Windows; otherwise, EBCDIC). You can avoid the padding by coding all 16 hexadecimal digits.

**VARIABLE**

Indicates that the actual name of the segment, including the S1 prefix, is read from the data record. The Variable-Name-Locator field specifies where in the data to get the name.

**Notes:**

1. Any page segment that is to be included in this manner must be defined in the PAGEFORMAT by using the SEGMENT command. Defining page segments enhances print performance.
2. If you specify VARIABLE for the SEGMENT name and don't want to print the name, then you must have at least one field command, or code PRINTDATA NO on the PRINTLINE command.

**START** \( n \)

The starting position in the data record to get the overlay name. The first data byte position of the input record is 1. If START is not coded, 1 is assumed.

**LENGTH** \( n \)

Length of field. Specifies the number \( (n) \) of bytes to process from the data record, beginning with the position specified in START. The maximum length is 8.

**DIRECTION**

```
+-----------------+
| DIRECTION       |
| ACROSS          |
|                 |
| DOWN            |
| BACK            |
| UP              |
```

Specifies the print direction of the line relative to the upper-left corner as you view the logical page. Not all printers can print in all print directions. For more information about your printer, see your printer documentation.

If DIRECTION is not specified, the direction that is specified in the PAGEFORMAT command is used. Observe that this direction is additive to the direction specified in the PAGEFORMAT command. See [PAGEFORMAT DIRECTION Subcommand](#).

**ACROSS**

The printline direction is rotated \( 0^\circ \) relative to the direction specified in the PAGEFORMAT (the printlines are oriented in the same direction as the page).

**DOWN**

The printline direction is rotated \( 90^\circ \) relative to the direction specified in the PAGEFORMAT.
BACK
The printline direction is rotated 180° relative to the direction specified in the PAGEFORMAT.

UP
The printline direction is rotated 270° relative to the direction specified in the PAGEFORMAT.

OBJECT

- OBJECT
  - Iname
  - relx relY
- VARIABLE
  - START n relx relY
- LENGTH n
- OBTYPE Parameters
- Other OBJECT Parameters

Specifies the placement of a resource object. If an internal name is coded, it is a known object that is defined by an OBJECT command. Otherwise, the object is a variable-named object whose name is extracted from fields in the line data as described by the START, LENGTH, FLDNUM, or RECID parameters. No OBJECT command exists for these objects; they must be specified with the OBTYPE and OBID parameters.

Notes:
1. All of the OBJECT parameters are treated as positional parameters. All positional parameters must be coded in the exact position and order as specified in the syntax diagram.
2. Multiple page/image objects that are used without specifying a page with OBPAGE default to using the first page in the object.

Iname
Specifies the local name of an object that is up to 16 alphanumeric characters in length. The Iname is used to match the PRINTLINE OBJECT subcommand to its definition from the OBJECT command. An object must be defined with this internal name by the OBJECT command.

relx relY
Specifies the number of units (such as inches or mm) that are added to the position of the current printline to position the top-left corner of the object. The values for the horizontal and vertical positioning are limited by the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command. Each position specification can be a positive or negative number with up to three decimal places. The units that are specified can be one of these: IN, MM, CM, POINTS, or PELS.

VARIABLE
Indicates that the actual name of the object is read from the data record. The Variable-Name-Locator field specifies where in the data to get the name.

Notes:
1. Any object that is to be included in this manner must be defined in the PAGEDEF by using the OBJECT command. Defining objects enhances print performance.
2. If you specify VARIABLE for the OBJECT name and don’t want to print the name, then you must have at least one field command, or code PRINTDATA NO on the PRINTLINE command.
START \( n \)
The starting position in the data record to get the object name. The first data byte position of the input record is 1. If START is not coded, 1 is assumed.

LENGTH \( n \)
Length of field. Specifies the number (\( n \)) of bytes to process from the data record, beginning with the position specified in START. The maximum length is 8.

**OBTYPE Parameters**

<table>
<thead>
<tr>
<th>OBJTYPE</th>
<th>PSEG</th>
<th>IOCA</th>
<th>BCOCA</th>
<th>GOCA</th>
<th>PTOCA</th>
<th>OTHER-OBID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>comp-id</td>
<td>type-name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Used to specify the type of the object. Observe that each of the object types restricts the type of mapping option that is allowed in the placement of the object (OBMAP on the OBJECT subcommand on the PRINTLINE command).

**PSEG**
Specifies a page segment object, as described in the *Mixed Object Document Content Architecture Reference*. All mapping types (OBMAP) are allowed by PPFA; however, the print server issues an error if any of the objects that are contained in the page segment are not compatible with the coded OBMAP parameter.

**GOCA**
Specifies a graphic object, as described in the *Graphics Object Content Architecture Reference*. You use GOCA to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**BCOCA**
Specifies a bar code object, as described in the *Bar Code Object Content Architecture Reference*. You use BCOCA to specify only the LEFT parameter on the OBMAP subcommand.

**IOCA**
Specifies an image object, as described in the *Image Object Content Architecture Reference*. You use IOCA to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.

**PTOCA**
Specifies a presentation text object with Object Environment Group (OEG) as described in the *Presentation Text Object Content Architecture Reference* and the *Mixed Object Document Content Architecture Reference*. You use the PTOCA object type to specify the LEFT parameter in the OBMAP subcommand.

**OTHER**
Specifies other object data. The object data that is included is a paginated presentation object with a format that might be defined by an AFP presentation architecture. When you specify OTHER, you must also specify the OBID parameter. You use OTHER to specify TRIM, FIT, CENTER, REPEAT, and FILL parameters on the OBMAP subcommand.
OBID

Specifies either a component identifier or a type name from Table 17. The OBID is translated into an Encoded OID and matched to the OID inside the object; they must match.

comp-id

Specifies the component identifier.

type-name

The name that is chosen by PPFA as an alternative to coding a component identifier.

Table 17. Non-OCA Objects supported by IOB

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OBID Object Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPS</td>
<td>13</td>
<td>Encapsulated PostScript</td>
</tr>
<tr>
<td>TIFF or TIF</td>
<td>14</td>
<td>Tag Image File Format</td>
</tr>
<tr>
<td>WINDIB</td>
<td>17</td>
<td>Device Dependent Bit Map [DIB], Windows Version</td>
</tr>
<tr>
<td>OS2DIB</td>
<td>18</td>
<td>Device Dependent Bit Map [DIB], PM Version</td>
</tr>
<tr>
<td>PCX</td>
<td>19</td>
<td>Paint Brush Picture File Format</td>
</tr>
<tr>
<td>GIF</td>
<td>22</td>
<td>Graphics Interchange Format</td>
</tr>
<tr>
<td>JFIF, JPEG, or JPG</td>
<td>23</td>
<td>AFPC (AFP Consortium) JPEG Subset</td>
</tr>
<tr>
<td>PDFSPO</td>
<td>25</td>
<td>PDF Single Page Object</td>
</tr>
<tr>
<td>PCLPO</td>
<td>34</td>
<td>PCL Page Object</td>
</tr>
<tr>
<td>EPSTR</td>
<td>48</td>
<td>EPS with Transparency</td>
</tr>
<tr>
<td>PDFSPOTR</td>
<td>49</td>
<td>PDF Single Page Object with Transparency</td>
</tr>
<tr>
<td>TIFF</td>
<td>61</td>
<td>TIFF Multiple Image File</td>
</tr>
<tr>
<td>TIFFFNT</td>
<td>62</td>
<td>TIFF Multiple Image without Transparency File</td>
</tr>
<tr>
<td>MPDF</td>
<td>63</td>
<td>PDF Multiple Page File</td>
</tr>
<tr>
<td>MPDFT</td>
<td>64</td>
<td>PDF Multiple Page with Transparency File</td>
</tr>
<tr>
<td>PNG</td>
<td>65</td>
<td>PNG File Format</td>
</tr>
<tr>
<td>AFPCTIFF</td>
<td>66</td>
<td>AFPC TIFF Subset</td>
</tr>
</tbody>
</table>

Table 18. Object Types that can be referenced as Secondary Resources

<table>
<thead>
<tr>
<th>Type-Name</th>
<th>Component-ID</th>
<th>Description of OID Type-Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFRO</td>
<td>26</td>
<td>PDF Resource Object (new)</td>
</tr>
<tr>
<td>RESCLRPRO</td>
<td>46</td>
<td>Resident Color Profile Resource Object</td>
</tr>
<tr>
<td>IOCAFS45RO</td>
<td>47</td>
<td>IOCA FS45 Resource Object Tile (new)</td>
</tr>
</tbody>
</table>

Other OBJECT parameters

OBSIZE

<table>
<thead>
<tr>
<th>OBSIZE</th>
<th>USEOBJ</th>
<th>wd</th>
<th>unit</th>
<th>hg</th>
<th>unit</th>
</tr>
</thead>
</table>

Chapter 11. Page Definition Command Reference 437
Specifies the size of the object placement area. When no OBSIZE is specified, the default is the size that is specified in the object. If no size is specified in the object, the size of the page is used. The page width is as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

wd  Specifies the width of an object placement area as a number with up to three decimal places. The allowable width might vary with the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

hg  Specifies the height of the object placement area as a number with up to three decimal places. The allowable height might vary with the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

unit  Specifies a unit of measurement for the width parameter. The choices are: IN, MM, CM, POINTS, or PELS.

Note: If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.

USEOBJ

Specifies that the size measurements that are specified in the object are to be used. If no size is specified in the object, the size of the page is used, which is the length and width as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

OBMAP

```
OBMAP | LEFT
      | TRIM
      | FIT
      | CENTER
      | REPEAT
      | FILL
```

Specifies mapping options. The OBMAP parameter defines the mapping of the object to the object placement area. If OBMAP is not coded, the mapping option within the object is used. If the object does not contain a mapping option, then the print server sets it to the created default for the container type.

Each object type (OBTYPE on the OBJECT command) specifies the allowable mapping options for that type. When it can, PPFA issues a message when these rules are violated. However, for an object type of page segment (OBTYPE=PSEG), PPFA does not know what types of objects are contained in it; therefore, PPFA cannot enforce the restrictions. See "OBJECT Command" on page 385 for a description of the restrictions.

LEFT

Specifies that the object is positioned at the upper, left corner of the
object placement area, as defined or defaulted by the *relative-xpos*, *relative-ypos*, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is not trimmed and might cause an exception condition by the presentation system.

**TRIM**

Specifies position and trim. The object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the *relative-xpos*, *relative-ypos*, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is trimmed.

**FIT**

Specifies scale to fit. This is the default value if the OBMAP parameter is not coded. The object is to be scaled to fit within the object placement area, as defined by the OBSIZE parameter. The center of the object is placed in the center of the object placement area and the object is scaled up or down to fit the block. Scaling in the horizontal and vertical directions is symmetrical. The FIT parameter ensures that all of the data in the object is presented in the object placement area at the largest possible size. The object is not trimmed.

**CENTER**

Specifies that the center of the object is positioned at the center of the object placement area. Any portion of the object that falls outside the object placement area is trimmed.

**REPEAT**

Specifies that the origin of the data object is positioned with the origin of the object placement area. The object is then replicated in the X and Y directions. If the last replicated data does not fit in the object area, it is trimmed to fit.

**FILL**

Specifies that the center of the data object is positioned coincident with the center of the object placement area. The data object is then scaled so that it totally fills the object placement area in both the X and Y directions. This might require that the object is asymmetrically scaled by different scale factors in the X and Y directions.

**OBCHPOS**

Specifies the horizontal offset of the object contents within the object placement area.

*x-pos*

The valid options for *x-pos* are described in the SETUNITS command for the horizontal value.

**USEOBJ**

Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.
OBCVPOS

Specifies the vertical offset of the object contents within the object placement area, as defined by the OBSIZE parameter. If OBCVPOS is not specified, it defaults to USEOBJ and uses the value set in the object. If no value is set in the object, the value defaults to 0. The OBCHPOS parameter is used only in LEFT and TRIM mapping of the object into the object placement area.

\textit{y-pos}

Specifies a positive or negative number. The valid options for \textit{y-pos} are described in the SETUNITS command for the vertical value.

USEOBJ

Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

OBROTATE \{0\,90\,180\,270\}

Specifies the object rotation in regard to the current LND's coordinate system.

Notes:

1. An included object is positioned and oriented in the following manner:
   
   \begin{itemize}
   \item All measurements are from the LND position that is established by the PRINTLINE position. Reference these measurements by using the inline direction of the printline.
   \item Measure the \textit{x-pos} and \textit{y-pos} units from the PRINTLINE current position to determine the object area origin.
   \item Apply any rotation from OBROTATE to modify the PRINTLINE axis, and to create the new object area coordinate system.
   \item Use the OBSIZE parameter to determine the object area size within the object area coordinate system, and to define the object placement area.
   \item To determine the object content origin, apply the Object Content Offset from parameters OBCHPOS (Object Content Horizontal Position) and OBCVPOS (Object Content Vertical Position) to the object area origin.
   \end{itemize}

2. The object content offset is used only for position (LEFT) and position and trim (TRIM) mapping options.

OBCOLOR
OBCOLOR—color-name

Specifies the color to be used as the default color or initial color for the object placement area. The OBCOLOR parameter is used only for objects of the PSEG, GOCA, BCOCA, and IOCA type. If the object type is OTHER, this parameter is ignored.

colorname

Values for colorname can be a defined color (see "DEFINE COLOR Command" on page 280) or one of these OCA color spaces:

- NONE
- DEFAULT
- BLACK
- BLUE
- BROWN
- GREEN
- RED
- PINK (or MAGENTA)
- TURQ (or CYAN)
- YELLOW
- DARKBLUE (or DBLUE)
- ORANGE
- PURPLE
- MUSTARD
- GRAY
- DARKGREEN (or DGREEN)
- DARKTURQ (or DTURQ)
- DARKCYAN (or DCYAN)

Note: This function requires both the print server and printer support. Check your print server and printer documentation.

OBPAGE

OBPAGE—n

Specifies the page number of a multipage object or file to be presented. n is the page number. A number 1 - 999999999 (9 digits) is valid.

OBRESOLUTION

OBRESOLUTION—x—y IN CM

Specifies the resolution and unit of measurement of an image. If the resolution is already specified inside the image, this information is ignored by the printer. Use this subcommand for images that do not or might not contain their resolution. Specify resolution of an image so that the printer can print the image correctly.

To specify object resolution, you must have a printer and a print server that support this capability.
If not specified, the default is to assume that the image resolution is the same as the printer. If the image does not print at the size you expect, use OBRESOLUTION to identify the image's resolution. With the resolution information, the printer is then able to print the image at the expected size.

\(x\) Specifies the number to be used for the horizontal resolution of an image. Specify an integer value in the range of 1 - 3276.

\(y\) Specifies the number to be used for the vertical resolution of an image. Specify an integer value in the range of 1 - 3276.

\textit{unit} Specifies a unit of measurement. The choices are:

- \texttt{IN} Inch
- \texttt{CM} Centimeter

In the following code example, the OBJECT subcommand is used to define a JFIF object (which can be specified as JPG). This object has a resolution of 300 pels per inch in both the \(x\) and \(y\) directions.

```
Pagedef obres2 replace yes;

PRINTLINE OBJECT VAR .4 .5 start 2 length 6
OBTYPE OTHER OBID JPG
OBRESOLUTION 300 300 IN;
```

**Example**

In the following example, the page definition \texttt{pd1} defines an object with an external name of \texttt{PSEGxyz}, of object type \texttt{PSEG}. The object has an internal name of \texttt{xyzintname}. The internal name identifies the object for the PRINTLINE OBJECT subcommand when the object is placed.

```
PAGEDEF pd1 Replace Yes
   COMMENT 'this is my program';

OBJECT xzZIntName
   OBXNAME PSEGxyz
   OBTYPE PSEG ;

PAGEFORMAT pf1;
PRINTLINE
   OBJECT xyzintname -1.1 in 2.1 in
   OBSIZE 3 in 5 in
   OBMAP FILL
   OBCOLOR BLUE ;
```

The PRINTLINE in PAGEFORMAT \texttt{pf1} places the object on the page 1.1 inches to the left and 2.1 inches below the current printline position. It also maps the object into the object area with the FILL parameter, which centers the object in the object area and totally fills the area, possibly with different scaling factors in the \(X\) and \(Y\) directions. It has an area size of 3 by 5 inches, and overrides the default presentation space color to BLUE.

---

**SEGMENT Command**

```
\texttt{SEGMENT—name—;}
```

Use the SEGMENT command only if you want page segments to be loaded to the printer before the page begins printing. If segments are used repeatedly and need
to be available in the printer, this command eliminates the need to load them each time. However, they do take up raster-pattern storage.

A separate SEGMENT command is required for each page segment with a maximum of 127 SEGMENT commands within a single page format. A SEGMENT command is nested within the page format and follows the PAGEFORMAT command.

**For Traditional**

```
PAGEFORMAT
TRCREF
SEGMENT
...
SEGMENT
```

*Note:* To include a page segment on a page without using an Include Page Segment structured field within the user data, see the "PRINTLINE Command" on page 423.

**For Record Format and XML**

```
PAGEFORMAT
SEGMENT
...
SEGMENT
```

**SEGMENT name**

Specifies the alphanumeric name of 1 - 6 characters (user-access name) of the page segment. Each name must be unique within a single page format.

*Note:* The prefix S1 is not part of the six-character user-access name.

**SETUNITS Command**

```
SETUNITS x y LINESP n
```

The SETUNITS command specifies the value and the unit of measurement that are the defaults for any subsequent measurement parameter in all of the commands and subcommands. These values remain the default values until another SETUNITS command is specified. The SETUNITS command must be specified as the first command in a page definition. If this command or a measurement parameter is not specified, the defaults that are identified within the following description are used.

**SETUNITS**

Specifies the value and the unit of measurement that are the defaults for any subsequent measurement parameter in all of the commands and subcommands.

- **x** Specifies the number that is used for horizontal measurement. A number with up to three decimal places is used. The default is 1. The choices are IN, MM, CM, POINTS, PELS, or LPI. The default is IN.
Note: This value affects subsequent OFFSET subcommands.

\( y \) Specifies the number that is used for vertical measurement. A number with up to three decimal places is used. The default is 1. The choices are IN, MM, CM, POINTS, PELS, or LPI. The default is IN.

Note: This value affects subsequent OFFSET subcommands.

Using CPI and LPI Units of Measurement: The CPI and LPI units of measurement make it possible to write the following command:

\[ \text{SETUNITS 10 CPI 6 LPI} \]

This command sets the units of measurement for horizontal and vertical spacing in terms of characters per inch and lines per inch. You can then use the OFFSET subcommand specifications to increment the spacing one character or one line at a time. The distance that is specified by \( n \) characters over and by \( n \) lines down is defined in the governing SETUNITS command. This example shows 10 characters per inch (CPI) and 6 lines per inch (LPI).

**Subcommand**

**LINEP**

Determines the line density or **leading** of the text. Any unit of measurement can be used.

**For Traditional**

This subcommand value affects:

- The following PRINTLINE NEXT subcommand
- The vertical (\( y \)) position of the first line on a logical page when the LINEONE subcommand is not specified and the default is assumed

The default is 6 LPI. If LINEP is allowed to default to 6 LPI, the LINEONE default is one L-unit less than 80% of 1/6 inch.

**For Record Format and XML**

This subcommand value affects the LAYOUT NEXT subcommand:

\( n \)

The meaning is determined by the type of unit-of-measurement that is specified in the unit parameter.

**LPI**

The number of lines per inch

**All others**

The distance between lines

\( unit \)

Specifies a unit of measurement. The choices are:

**IN**   Inch

**LPI**

Lines-per-inch

**MM**   Millimeter

**CM**   Centimeter

**PELS**

L-units per inch (The number of L-units per inch can be defined by the user or can default to 240 L-units in an inch)
The TRCREF command specifies the relationship between a font and a table-reference character (TRC) in the data. When specified, the TRCREF command must immediately follow a PAGEFORMAT command.

Depending on the value that is specified for \( n \), the TRC is interpreted by the print server as being either S/370 1403 line-mode compatible or S/370 1403 line-mode incompatible: Notice that, if compatibility TRCs are to be used, no fonts are specified in any PRINTLINE or FIELD commands within the same PAGEFORMAT.

- **0 - 3** Indicate a compatible TRC for a S/370 1403 line-mode data stream.
- **4 - 126** Indicate an incompatible TRC for a S/370 1403 line-mode data stream.

Also, notice that any TRC number outside the range of 0 - 3 results in non-compatibility TRCs for the entire page definition. If compatibility TRCs are used, do not specify fonts on PRINTLINE or FIELD commands within the same PAGEFORMAT.

**Subcommands**

**FONT**

Specifies the font that is associated with the TRC number.

\[ name \]

Specifies the local name of a font; the font must be one that is named in a FONT command.
If you used both the user-access name and the local name in the FONT command, use the local name here. If you used only the user-access name, use it here.

DIRECTION
Specifies the print direction of the line relative to the upper-left corner as you view the logical page. Not all printers can print in all print directions. For more information about your printer, see your printer documentation.

The DIRECTION on the TRCREF command must match the DIRECTION of the PRINTLINE command with which the TRC is to be used. If TRCREF DIRECTION subcommand is not specified, DIRECTION ACROSS is assumed. Observe that this direction is additive to the direction specified in the PAGEFORMAT command.

ACROSS
The page is printed with the characters added to the page from left to right, and the lines added from the top to the bottom.

DOWN
The page is printed with the characters added to the page from top to bottom, and the lines added from the right to the left.

BACK
The page is printed with the characters added to the page from right to left, and the lines added from the bottom to the top.

UP
The page is printed with the characters added to the page from bottom to top, and the lines added from the left to the right.

ROTATION
Specifies the rotation of characters in degrees. The specified value is relative to the inline direction of the printline.

Valid rotations are 0°, 90°, 180°, or 270°; 0 is the default.

If the TRCREF ROTATION subcommand is not specified, the default is the rotation value that is specified on the FONT command.

XsetLayout Command (XML)
Chapter 11. Page Definition Command Reference 447
The XLAYOUT command addresses an XML data item by specifying a QTAG (qualified tag) for that data. A QTAG is a series of XML start tags that fully identify the XML data item.

Before PSF prints the data, it scans the XML data item and matches it to an XLAYOUT command in the page definition by using its QTAG. The matching XLAYOUT command in the page definition is used to position and format the associated XML data item and its attributes on the printed page.

The XML page definition function has the following new PPFA concepts:

**Relative Inline Positioning:**
Relative inline positioning places data relative to the current position. If you position a text field and then place the text, the end of the text becomes the new current position. Graphics, bar codes, objects, segments, and overlays do not change the current position after they are originally positioned. For example, if you position a line with a DRAWGRAPHIC LINE command, the new current position is the starting point of that line. The length of the graphic line does not change the current position.

These restrictions must be observed when you are using relative inline positioning:

1. XLAYOUT commands with relative positioning cannot contain any of these:
   - FIELD commands with inline positioning relative to the XLAYOUT (LPOS)
   - FIELD ATTR (attribute) with inline positioning relative to the XLAYOUT (LPOS)
   - FIELD commands with bar codes
   - DRAWGRAPHIC commands
   - OBJECT subcommands
   - SEGMENT subcommands
   - OVERLAY subcommands

2. You can use the SAME parameter only for inline positioning on the XLAYOUT command when the previously used XLAYOUT command used absolute inline positioning.

**Absolute Inline Positioning:**
Allows absolute inline positioning on a FIELD command for specific placement of elements.

**Attributes are Special FIELDs:**
The attribute is identified by name and the data that is printed is from the attribute value or a portion of the attribute value and not from the element content.

**Notes:**

1. If a FIELD is used for presenting any piece of data on the XLAYOUT command, FIELD commands must be used for all pieces of data that is presented on the XLAYOUT command. Because an attribute is a special field, if you want to print both an attribute value and the element data, you need to code the attribute field for the attribute value and a regular field for the element data.

2. PSF suppresses leading and trailing blanks (X'40' for EBCDIC or X'20' for ASCII) in the data. Multiple embedded blanks are reduced to one blank.
3. The XLAYOUT command defines an XML page definition and cannot be mixed with PRINTLINE commands, which define traditional page definitions, or LAYOUT commands, which define Record Formatting page definitions.

Subcommands

```
DEFAULT
  qtagname

QTAG
  starttag
```

**DEFAULT**

This keyword is used only when the layout type is either PAGEHEADER or PAGETRAILER, and no name is needed. Only one default PAGEHEADER or PAGETRAILER can be specified in a PAGEFORMAT.

**qtagname**

The *qtagname* is a defined Qualified Tag. It is defined by the DEFINE *qtagname* QTAG command at the beginning of the page definition.

**QTAG** **starttag**

An explicit Qualified Tag. It is defined by coding a series of start tags that are separated by commas. A start tag is an XML data element name. Put the start tag in quotation marks if you want to preserve its case. Otherwise, it is folded to uppercase.

In Figure 135, *person*, *name*, and *first* are start tags. The qualifying tag for the data item *Case* is 'person','name','last'. In the example page definition, both of the *x* and *y* XLAYOUT commands address the same XML data item *Case*.

```
<person>
  <name>
    <first>Justin</first>
    <last>Case</last>
  </name>
</person>

PAGEDEF xxx...;
  DEFINE lname QTAG 'person','name','last';
  ...
  Pageformat x ...
    XLAYOUT lname POSITION...
  ...
  Pageformat y ...
    XLAYOUT QTAG 'person','name','last' POSITION ...
  ...
```

*Figure 135. Example of XML data with the associated page definition*

Body
The BODY layout type is used for most of data in your database. This is the default.

**GROUP**

The GROUP parameter indicates that the existing group header must be saved and used for subsequent pages. If this parameter is not set when processing starts on a BODY layout, the active group header record is discarded and not reprinted on subsequent pages.

**XSPACE**

XSPACE indicates the amount of extra space from the position of the layout to the bottom of the group header area. The user can identify the amount of extra space in excess of one text line that is used by the header so that the baseline moves down and the following group data is not placed on top of the header area. This space is not calculated by PPFA and must be explicitly defined by the user. See Figure 136 (shaded space shows group header area):

<table>
<thead>
<tr>
<th>Checks</th>
<th>Check No.</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>352</td>
<td>01/04/90</td>
<td>$321.50</td>
</tr>
<tr>
<td></td>
<td>353</td>
<td>01/05/90</td>
<td>$100.00</td>
</tr>
<tr>
<td></td>
<td>354</td>
<td>01/10/90</td>
<td>$122.30</td>
</tr>
</tbody>
</table>

*Figure 136. Example Showing the Use of XSPACE.*

When a Group Header record is processed and is still active when the PAGEFORMAT is left, the group header record is saved by the presentation services program. Whenever the same PAGEFORMAT is reinvoked, this saved group header record is presented again if the first body record after reinvoking the PAGEFORMAT selects a Body record that has the Group Indicator on.
PAGEHEADER
This layout type specifies a header that is to be printed on each new page. The baseline position of this layout is normally in the top margin, but can be anywhere on a logical page. If RELATIVE is specified, the position is considered to be relative to the page origin. Usually contains customer's name, address, account number, and so forth. Only one default PAGEHEADER layout can be specified in a PAGEFORMAT and no input record data can be specified in a default layout.

CONTINUE
The CONTINUE parameter indicates that this XLAYOUT command is a continuation of the Page Header definition. The formation of the Page Header might require the data from more than one data element, which is done by specifying the CONTINUE parameter.

PAGETRAILER
This layout type specifies a trailer that is to be printed on each new page. The baseline position of this layout is normally in the bottom margin, but can be located anywhere on a logical page and can be specified as RELATIVE. Only one default PAGETRAILER layout can be specified in a PAGEFORMAT and no input record data is processed with a default layout. It can contain the name of the form or a footnote.

CONTINUE
The CONTINUE parameter indicates that this XLAYOUT command is a continuation of the Page Trailer definition. The formation of the Page Trailer might require the data from more than one data element, which is done by specifying the CONTINUE parameter.

GRPHEADER
This layout type specifies a header that is to be printed at the beginning of a group of data. If a logical page eject occurs before the group of data ends, the header is printed after the top margin on each new page until the group ends. The baseline position of this layout can be specified as RELATIVE. It can include column headings.

CONTINUE
The CONTINUE parameter indicates that this XLAYOUT command is a continuation of the Group Header definition. The formation of the Group Header might require the data from more than one data element, which is done by specifying the CONTINUE parameter.

NEWPAGE
This parameter indicates that a new page must be started with this layout name. If it is a header or trailer layout, the print position is moved to the start of a new page before this header or trailer becomes the active header or trailer.

DELIMITER
This parameter allows the definition of a delimiter at the specified position in the record. The delimiter is used to delimit fields in the record.
The delimiter is a one or two-byte code that is specified in either character or
hex indicates a delimiting character within the customer's database and is used
to separate fields. PPFA does not translate these characters. Hex characters
must be entered in uppercase within the quotation marks.

<table>
<thead>
<tr>
<th>PRINTDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
</tr>
<tr>
<td>NO</td>
</tr>
</tbody>
</table>

Specifies whether the line of data that is associated with the current LAYOUT
must be printed. The PRINTDATA subcommand is useful when the data
stream is interspersed with lines of comments, blank lines, or lines without
data that are not meant to be printed.

**YES**
- Specifies that the data for the current XLAYOUT is printed. YES is the
default.

**NO**
- Specifies that the data for the current XLAYOUT is not printed.

<table>
<thead>
<tr>
<th>DIRECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACROSS</td>
</tr>
<tr>
<td>DOWN</td>
</tr>
<tr>
<td>BACK</td>
</tr>
<tr>
<td>UP</td>
</tr>
</tbody>
</table>

Specifies the print direction of the line relative to the upper-left corner as you
view the logical page. Not all printers can print in all print directions. For
more information about your printer, see your printer documentation.

If DIRECTION is not specified, the direction that is specified in the
PAGEFORMAT command is used. Observe that this direction is additive to the
direction specified in the PAGEFORMAT command. See "PAGEFORMAT
Command" on page 415.

**ACROSS**
- The layout direction is rotated 0 degrees relative to the direction specified
  in the PAGEFORMAT (the layouts are oriented in the same direction as the
  page).

**DOWN**
- The layout direction is rotated 90 degrees relative to the direction specified
  in the PAGEFORMAT.

**BACK**
- The layout direction is rotated 180 degrees relative to the direction
  specified in the PAGEFORMAT.

**UP**
- The layout direction is rotated 270 degrees relative to the direction
  specified in the PAGEFORMAT.

<table>
<thead>
<tr>
<th>POSITION</th>
</tr>
</thead>
</table>

452  Page Printer Formatting Aid: User’s Guide
Positions FIELD, DRAWGRAPHIC, and ENDGRAPHIC text and graphics. If Relative is specified or POSITION is not specified, the baseline of the Position is relative to the previous LAYOUT position.

1. For PAGEHEADER RCD: The baseline position can be anywhere on a logical page, but cannot be specified as Relative.
2. For PAGETRAILER, GROUPHEADER, and BODY RCDs: The baseline position can be anywhere on a logical page and can be specified as RELATIVE.

Specifies the starting position of the layout in the printout.

**RELATIVE**

Specifies the horizontal position and vertical position values that are relative to the current inline position.

**horizontal position**

Specifies that the following horizontal position value is to be processed as a value relative to the current inline position.

**x-pos**

Specifies the horizontal offset from the left side of the logical page. The value is a number with up to three decimal places. The valid options for x-pos are described in the SETUNITS command for the horizontal value.

**LEFTMARGIN**

Specifies this line starts at the position that is specified as the horizontal (x) value in the previous LEFTMARGIN subcommand within the page definition.

**SAME**

Specifies this line starts at the same horizontal offset position as the previously coded XLAYOUT. If applied to the first XLAYOUT of a logical page, the horizontal position is 0, which is the default.

**Note:** This parameter is not valid with RELATIVE horizontal.

**=**

Alternative for SAME.

**vertical position**

Specifies that the following vertical position value is to be processed as a relative value. The XLAYOUT is positioned relative to the last XLAYOUT placed on the page.
$y$-pos
Specifies the vertical offset from the top side of the logical page.
The value options for $y$-pos are described in the SETUNITS command for the vertical value.

TOPMARGIN
Specifies that the XREQUEST is placed in the position that is
specified as the vertical (y) value in the TOPMARGIN subcommand within the page definition.

NEXT
Specifies the layout is to be positioned down (on the logical page)
one line (as defined in the LINESP subcommand of the last
SETUNITS command) from the previous field. The LINESP subcommand of the SETUNITS command establishes the distance from one line to the next.

When NEXT is specified for the first XREQUEST of a logical page,
the starting position of the line is one line down from the top of
the logical page, as defined by the TOPMARGIN subcommand.

Note: The down direction is determined by the direction of the logical page (as specified in the page format), not the XREQUEST direction. Therefore, NEXT is useful in ACROSS XREQUESTs.

SAME
Specifies this XREQUEST starts at the same vertical position as the
previous XREQUEST.

= Alternative for SAME.

Notes:
1. If both TOP and RELATIVE are requested for the $y$-pos value, the
   RELATIVE request is ignored.
2. When you are using RELATIVE positioning, PPFA does not flag off-the-page conditions for the position of an XREQUEST or for any overlays, segments, or objects placed relative to that XREQUEST. XREQUESTs that fall outside the bounds of the logical page are flagged by the print server at run time.
3. When you are specifying RELATIVE, use the minus sign to indicate any negative values for the XREQUEST vertical position; you can use the plus sign to indicate positive values. If no sign is used, a positive value is assumed.
4. The DIRECTION for a relative XREQUEST must be ACROSS. Fields that are associated with a relative XREQUEST must have the same DIRECTION as the XREQUEST and must match the PAGEFORMAT DIRECTION.
5. If RELATIVE is specified with SAME or = as the $y$ value, the relative value in the XREQUEST is +0.
6. RELATIVE positioning is allowed on an XREQUEST command only if the XREQUEST and all its associated FIELD commands are formatted to print in the same direction as the PAGEFORMAT. That is, the DIRECTION parameter in the XREQUEST and any associated FIELD commands must specify (or default to) ACROSS. The DIRECTION in the PAGEFORMAT or PAGEDEF command can be any allowable value: ACROSS, DOWN, BACK, or UP.
ENSPACE

If the remaining body space is less than the value specified, ENSPACE causes a logical page eject to be done. For example, ENSPACE can be used on a GRPHEADER layout to ensure that a group header does not print at the end of a page without the first data record of the group. ENSPACE does not include the space within the bottom margin (specified on the PAGEDEF or PAGEFORMAT command). This indicator is ignored on a PAGEHEADER or PAGETRAILER layout.

COLOR

Specifies an OCA or defined color for the text of this field. This subcommand is recognized only by printers that support multiple-color printing. See your printer publication for information about the colors that can be printed.

colormname
 Values for colormname can be a defined color (see “DEFINE COLOR Command” on page 280), or an OCA colormname. Values for OCA colormnames are:

- NONE
- DEFAULT
- BLACK
- BLUE
- BROWN
- GREEN
- RED
- PINK (or MAGENTA)
- TURQ (or CYAN)
- YELLOW
- DARKBLUE (or DBLUE)
- ORANGE
- PURPLE
- MUSTARD
- GRAY
- DARKGREEN (or DGREEN)
- DARKTURQ (DTURQ, or DCYAN, or DARKCYAN)

The color choices depend on the printer.
If you do not enter one of these colors, the default color for that printer is used. NONE is the color of the medium. DEFAULT is the printer default color.

**Note:** In some printer manuals, the color turquoise (TURQ) is called cyan, and the color pink (PINK) is called magenta.

PPFA supports the following synonyms:

- CYAN for TURQ
- DARKCYAN for DARKTURQ
- DBLUE for DARKBLUE
- DCYAN for DARKTURQ
- DGREEN for DARKGREEN
- DTURQ for DARKTURQ
- MAGENTA for PINK

color models

Specifies the color of print for this field that is supported in MO:DCA for the Red/Green/Blue color model (RGB), the highlight color space, the Cyan/Magenta/Yellow/Black color model (CMYK), and the CIELAB color model.

**RGB** rvalue gvalue bvalue

Three RGB integer values are used. The first (rvalue) represents a value for red, the second (gvalue) represents a value for green, and the third (bvalue) represents a value for blue. Each of the three integer values can be specified as a percentage 0 - 100.

**Note:** An RGB specification of 0/0/0 is black. An RGB specification of 100/100/100 is white. Any other value is a color somewhere between black and white, depending on the output device.

**HIGHLIGHT** hvalue **COVERAGE** cvalue **BLACK** bvalue

Indicates the highlight color model. Highlight colors are device-dependent.

You can use an integer within the range of 0 - 65535 for the hvalue.

**Note:** An hvalue of 0 indicates that no default value is defined; therefore, the default color of the presentation device is used.

COVERAGE indicates the amount of coverage of the highlight color to be used. You can use an integer within the range of 0 - 100 for the cvalue. If less than 100 percent is specified, the remaining coverage is achieved with the color of the medium.

**Note:** Fractional values are ignored. If COVERAGE is not specified, a value of 100 is used as a default.

BLACK indicates the percentage of black to be added to the highlight color. You can use an integer within the range of 0 - 100 for the bvalue. The amount of black shading that is applied depends on the COVERAGE percentage, which is applied first. If less than 100 percent is specified, the remaining coverage is achieved with black.

**Note:** If BLACK is not specified, a value of 0 is used as a default.

**CMYK** cvalue mvalue yvalue kvalue

Defines the cyan/magenta/yellow/black color model. cvalue specifies the cyan value. mvalue specifies the magenta value. yvalue specifies the
yellow value. kvalue specifies the black value. You can use an integer percentage within the range of 0 - 100 for any of the CMYK values.

**CIELAB** Lvalue (-)c1value (-)c2value  
Defines the CIELAB model. Use a range of 0.00 - 100.00 with Lvalue to specify the luminance value. Use signed integers from -127 to 127 with c1value and c2value to specify the chrominance differences.

Lvalue, c1value, c2value must be specified in this order. No defaults exist for the subvalues.

**Notes:**
1. Do not specify both an OCA color with the COLOR subparameter and an extended color model on the same FIELD or PRINTLINE command. The output is device-dependent and might not be what you expect.
2. Do not specify two extended COLOR subcommands on the same FIELD or PRINTLINE command.

**FONT**

```
FONT — name1 , — name2
```

Defines the font to be used for the layout.

`name1`  
Specifies the name of a font that is used to print the data. This font must be defined in a previous FONT command in the page definition.

If Shift-Out, Shift-In (SOSI) processing is used, `name1` must be the single-byte font.

`name2`  
Specify when are only using Shift-Out, Shift-In (SOSI) processing to dynamically switch between a single-byte font and a double-byte font within the layout. `name2` must be the double-byte font.

**Notes:**
1. If this subcommand is not specified in the print data, the print server uses the font indicated. Otherwise, the print server selects a default font.
2. `name2` is only valid with EBCDIC data.

**OBJECT**

```
OBJECT — lname 0 0 Other OBJECT Parameters
```

Specifies the local name of an object that is to be positioned and oriented relative to the location specified in the XLAYOUT command in which the OBJECT subcommand was named. The OBJECT, as identified by the `lname` parameter, must be defined by an OBJECT command.
**Note:** Multiple page/image objects that are used without specifying a page with OBPAGE default to using the first page in the object.
You can place multiple objects on the same XLAYOUT command and you can place the same object multiple times. Each placement must have its own set of placement parameters, as follows:

*lname*

Specifies the local name of an object that is up to 16 alphanumeric characters in length. The *lname* parameter is used to match the XLAYOUT OBJECT subcommand to its definition from the OBJECT command. An object must be defined with this internal name by the OBJECT command.

*relX relY*

Specifies the number of units (such as inches or mm) that are added to the position of the current XLAYOUT to position the top-left corner of the object. The values for the horizontal and vertical positioning are limited by the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

Each position specification can be a positive or negative number with up to three decimal places. The units that are specified can be one of these: IN, MM, CM, POINTS, or PELS.

**Other OBJECT Parameters**

**OBSIZE**

```
OBSIZE
```

Specifies the size of the object placement area. When no OBSIZE is specified, the default is the size that is specified in the object. If no size is specified in the object, the size of the page is used. The page width is as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

*wd*

Specifies the width of an object placement area as a number with up to three decimal places. The allowable width might vary with the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

*hg*

Specifies the height of the object placement area as a number with up to three decimal places. The allowable height might vary with the type of printer that is used and the L-units that are specified with the PELSPERINCH parameter on the PAGEDEF or PAGEFORMAT command.

*unit*

Specifies a unit of measurement for the width parameter. The choices are: IN, MM, CM, POINTS, or PELS.

**Note:** If no unit is specified, the default is the most recent SETUNITS command value or IN (inch) if a SETUNITS command is not issued.
USEOBJ

Specifies that the size measurements that are specified in the object are to be used. If no size is specified in the object, the size of the page is used, which is the length and width as specified on the PAGEDEF or PAGEFORMAT commands, or it defaults to 8.3 inches by 10.8 inches.

OBMAP

| OBMAP | LEFT | TRIM | FIT | CENTER | REPEAT | FILL |

Specifies mapping options. The OBMAP parameter defines the mapping of the object to the object placement area. If OBMAP is not coded, the mapping option within the object is used. If the object does not contain a mapping option, then the print server sets it to the created default for the container type.

Each object type (OBTYPE on the OBJECT command) dictates the allowable mapping options for that type. When it can, PPFA issues a message when these rules are violated. However, for an object type of page segment (OBTYPE=PSEG), PPFA does not know what types of objects are contained in it; therefore, PPFA cannot enforce the restrictions. See “OBJECT Command” on page 385 for a description of the restrictions.

LEFT

Specifies that the object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the relative-xpos, relative-ypos, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is not trimmed and might cause an exception condition by the presentation system.

TRIM

Specifies position and trim. The object is positioned at the upper, left corner of the object placement area, as defined or defaulted by the relative-xpos, relative-ypos, OBCHPOS, and OBCVPOS parameters. Any portion of the object that falls outside the object placement area as defined by the OBSIZE parameter is trimmed.

FIT

Specifies scale to fit. This is the default value if the OBMAP parameter is not coded. The object is to be scaled to fit within the object placement area, as defined by the OBSIZE parameter. The center of the object is placed in the center of the object placement area and the object is scaled up or down to fit the block. Scaling in the horizontal and vertical directions is symmetrical. The FIT parameter ensures that all of the data in the object is presented in the object placement area at the largest possible size. The object is not trimmed.

CENTER

Specifies that the center of the object is positioned at the center of
the object placement area. Any portion of the object that falls outside the object placement area is trimmed.

**REPEAT**  
Specifies that the origin of the data object is positioned with the origin of the object placement area. The object is then replicated in the X and Y directions. If the last replicated data does not fit in the object area, it is trimmed to fit.

**FILL**  
Specifies that the center of the data object is positioned coincident with the center of the object placement area. The data object is then scaled so that it totally fills the object placement area in both the X and Y directions. This might require that the object is asymmetrically scaled by different scale factors in the X and Y directions.

**OBCHPOS**

```
| OBCHPOS—USEOBJ |
| OBCHPOS—x-pos  |
```

Specifies the horizontal offset of the object contents within the object placement area as a number.

**x-pos**  
Specifies a positive or negative number. The valid options for **x-pos** are described in the SETUNITS command for the horizontal value.

**USEOBJ**

Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

**OBCVPOS**

```
| OBCVPOS—USEOBJ |
| OBCVPOS—y-pos  |
```

Specifies the vertical offset of the object contents within the object placement area, as defined by the OBSIZE parameter. If OBCVPOS is not specified, it defaults to USEOBJ and uses the value set in the object. If no value is set in the object, the value defaults to 0. The OBCHPOS parameter is used only in LEFT and TRIM mapping of the object into the object placement area.

**y-pos**

Specifies a positive or negative number. The valid options for **y-pos** are described in the SETUNITS command for the vertical value.

**USEOBJ**

Specifies that the offset value from the object is to be used. If no value is set in the object, the value defaults to 0.

**OBROTATE** \{0|90|180|270\}
OBROTATE

Specifies the object rotation in regard to the current LND’s coordinate system.

OBCOLOR

OBCOLOR — colorname

Specifies the color to be used as the default color or initial color for the object placement area. The OBCOLOR parameter is used only for objects of the PSEG, GOCA, BCOCA, and IOCA type. If the object type is OTHER, this parameter is ignored.

Values for colorname can be a defined color (see “DEFINE COLOR Command” on page 280) or one of these OCA color spaces:

- NONE
- DEFAULT
- BLACK
- BLUE
- BROWN
- GREEN
- RED
- PINK (or MAGENTA)
- TURQ (or CYAN)
- YELLOW
- DARKBLUE (or DBLUE)
- ORANGE
- PURPLE
- MUSTARD
- GRAY
- DARKGREEN (or DGREEN)
- DARKTURQ (DTURQ, or DCYAN, or DARKCYAN)

OBPAGE

OBPAGE — n

Specifies the page number of a multipage object or file to be presented. n is the page number. A number 1 - 999999999 (9 digits) is valid.

OVERLAY

OVERLAY — xname — relX — relY — OVROTATE — 0

OVERLAY — xname — relX — relY — OVROTATE — 90

OVERLAY — xname — relX — relY — OVROTATE — 180

OVERLAY — xname — relX — relY — OVROTATE — 270
Specifies the name of an overlay that is to be positioned relative to the location specified in the XLAYOUT command in which the OVERLAY subcommand was named. The PAGEFORMAT OVERLAY command can contain the named overlays. The maximum number of overlays that are specified for a PAGEFORMAT including the XLAYOUT OVERLAY subcommand is 254.

Specifies the electronic overlay that is to be used with this subgroup.

\( Xname \)

Specifies the user-access name as defined in the OVERLAY command.

Notes:
1. PPFA checks for duplication of local names. If a duplication exists, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

\( relX \; relY \)

Specifies the number of units (such as inches or mm) that are added to the position of the layout to position the top-left corner of the overlay. The values for horizontal and vertical can be (+) or (-). The maximum value is + or - 32760 L-units. For example:
- OVERLAY NAME1 2 in 1 in
- OVERLAY NAME2 5 mm 1 mm

Note: Any offset coded in the overlay itself is added to this offset.

\( OVROTATE \{0\;90\;180\;270\} \)

Specifies the rotation of the placed overlay in regard to the x-axis of the page.

See "FORMDEF Command" on page 234 for an OVROTATE example, which is presented in the FORMDEF description.

\[ \text{SEGMENT} \]

\[ \begin{array}{c}
\text{SEGMENT} \quad Xname \\
\hline
\end{array} \]

\( relX \quad relY \)

Specifies the name of a segment that is to be positioned relative to the location specified in the XLAYOUT command in which the SEGMENT subcommand was named. The PAGEFORMAT SEGMENT command can contain the named segments. The maximum number of segments that are specified for a PAGEFORMAT including the XLAYOUT SEGMENT subcommand is 127.

Specifies the page segment that is to be used with this subgroup.

\( Xname \)

Specifies the user-access name as defined in the SEGMENT command.

Notes:
1. PPFA checks for duplication of local names. If a duplication exists, the page definition is generated, but a warning message is issued.
2. PPFA does not check for duplicate user-access names.

\( relX \; relY \)

Specifies the number of units (such as inches or mm) that are added to the position of the layout to position the top-left corner of the page segment.
The values for horizontal and vertical can be (+) or (-). The maximum value is + or - 32760 L-units. For example:

- SEGMENT MYSEG1 2 in 1 in
- SEGMENT MYSEG1 5 mm 1 mm

**Example of printing XML data with a page definition**

This example shows the XML data in Figure 137 that is printed with the page definition in Figure 138 on page 464 to get these results:

<table>
<thead>
<tr>
<th>Home customer: Justin Case</th>
<th>123 Redlight Lane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work customer: Anna Merkin</td>
<td>1911 Colt Lane</td>
</tr>
<tr>
<td></td>
<td>Twistnshout, MA 01050</td>
</tr>
<tr>
<td></td>
<td>Longmont, CO 80501</td>
</tr>
</tbody>
</table>

**XML Data**

```xml
<customer type='Home'>
  <name>
    <first>Justin</first>
    <last>Case</last>
  </name>
  <address>
    <strno>123</strno>
    <street>Redlight Lane</street>
    <city>Twistnshout</city>
    <state>MAMassachusetts</state>
    <zip>01050</zip>
  </address>
</customer>

<customer type='Work'>
  <name>
    <first>Anna</first>
    <last>Merkin</last>
  </name>
  <address>
    <strno>1911</strno>
    <street>Colt Lane</street>
    <city>Longmont</city>
    <state>COColorado</state>
    <zip>80501</zip>
  </address>
</customer>
```

*Figure 137. XML data for printing*
Page Definition

SETUNITS 1 IN 1 IN LINESP 6 LPI;
Pagedef XMLxml replace yes UDType EBCDIC;
FONT E21H0C TYPE EBCDIC;
DEFINE cust QTAG 'customer';
DEFINE name QTAG 'customer','name';
DEFINE fname QTAG 'customer','name','first';
DEFINE lname QTAG 'customer','name','last';
DEFINE addr QTAG 'customer','address';
DEFINE strno QTAG 'customer','address','strno';
DEFINE street QTAG 'customer','address','street';
DEFINE city QTAG 'customer','address','city';
DEFINE state QTAG 'customer','address','state';
DEFINE zip QTAG 'customer','address','zip';

XLAYOUT cust POSITION ABSOLUTE 0
  FIELD ATTR 'type';
  FIELD TEXT 'customer: ';
XLAYOUT fname POSITION ABSOLUTE 2.5 SAME;
XLAYOUT lname POSITION RELATIVE 0.167 SAME;
XLAYOUT strno POSITION ABSOLUTE 5.5 SAME;
XLAYOUT street POSITION RELATIVE 0 SAME;
  FIELD TEXT ' ';
XLAYOUT city POSITION ABSOLUTE 5.5 NEXT;
  FIELD START 1 LENGTH *;
XLAYOUT state POSITION RELATIVE 0 SAME;
  FIELD TEXT ' ';
XLAYOUT zip POSITION RELATIVE 0 SAME;

Figure 138. Page definition used to print XML data
Part 4. Appendixes
Appendix A. System Dependencies for PPFA

PPFA is a cross system product that operates on:
- Virtual Machine (VM)
- Virtual Storage Extended (VSE)
- z/OS

For the level of the operating system on which PPFA can run, see the PPFA/370: Licensed Program Specification, G544-3696.

PPFA creates page definitions and form definitions that are used for printing by PSF for z/OS and PSF/VM. Page definitions and form definitions that are created on one system can be used for printing on another system. However, not all versions of print servers support all functions that are provided by PPFA. Use the Programming Guide or User’s Guide for your print server system to determine which functions are supported by your system.

While page definitions and form definitions created on one system can be used on any of the systems, the method of creating these resources is different.

Each system is presented to show how PPFA creates page definitions and form definitions. In the examples, the prefixes F1 and P1 are automatically added by PPFA to the user name designated for form definitions and page definitions.

Note: All PPFA/370 functions that are described in this document are available on the z/OS operating system. However, because functions on PSF/VM and PSF/VSE and PPFA/370 functions for VM and VSE are not updated, some of the PPFA/370 functions that are described in this document are not available on the VM and VSE operating systems.

VSE Environment

PPFA can operate in any partition of VSE. It operates in batch mode but is able to operate in a partition that is occupied by an interactive processor.

Storing PPFA Resources

Form definitions and page definitions are stored by name in a library. In VSE, sublibraries are created for form-definition and page-definition storage within the system library.

The following job control statements (JCS) give an example of a PPFA execution under VSE. The ¢ in Column 72 indicates a continuation.
Rules for VSE

The rules for VSE commands in a PPFA execution are:

- All characters in the EXEC statement parameters must be uppercase. Each keyword in a parameter must be unique; PPFA issues an error message if any keywords are duplicated.
- AKQPPFA is the program name.
- SIZE= is the maximum available storage in the program. The SIZE parameter is not used to specify a PPFA work area size.
- PARM= is used to input PPFA parameters.
  - FORMLIB= (or PAGELIB=) libraryname.sublibraryname
    - All library names are alphanumeric (1 - 7 characters); the first character must be alphabetic.
    - All sublibrary names are alphanumeric (1 - 8 characters) including the first character.
  - size=nnK or nnnM
    - Defines the work area in which PPFA compiles the page definitions and form definitions. The default is 128K. The minimum is 4K and the maximum is 16M.
- The format for the FORMLIB or PAGELIB parameters is:
  - FORMLIB= (or PAGELIB=) libraryname.sublibraryname, where library names are 1 - 7 characters long and sublibrary names are 1 - 8 characters long.
  - All characters (library and sublibrary names) are alphanumeric, except that the first character must be alphabetic.
- Libraries must be defined before PPFA is run; Otherwise, an ABEND occurs. PPFA can do a syntax check without defined libraries, but it cannot define its own libraries;
- The SYSIPT file drives PPFA. It contains the commands that are used to build form definitions and page definitions. The records are fixed-length records of either 80 or 81 bytes, which can be blocked. The last 8 bytes of the records are treated as comments.
z/OS Environment

The following example shows you how to create page definitions and form
definitions in the z/OS environment.

Form definitions and page definitions are stored by name in a library.

PPF for z/OS is run as a batch program with Job Control Language (JCL). The
JCL statements are an example of PPFA execution under z/OS:

```
//JOBPPFA JOB TOKYO
//STEP EXEC PGM=AKQPPFA
//STEPLIB DD DSN=ppfa.program,DISP=SHR
//SYSPRINT DD SYSOUT=A
//FORMLIB DD DSN=ppfa.formlib,DISP=SHR
//PAGELIB DD DSN=ppfa.pagelib,DISP=SHR
//SYSIN DD *

PPFA control statements
.
.

/*
```

The SYSIN file contains the commands that are used to build form definitions and
page definitions. The records can be fixed length or variable length, and they can
be blocked. The maximum length for fixed-length records is 100 bytes; the
maximum length for variable-length records is 104 bytes. For fixed 80-byte records,
the last 8 bytes are treated as comments.

The record format for the page-definition and form-definition data sets must be
variable blocked (VBM). The block size and record length must be 8209 and 8205.
PPFA uses all of the available storage in the program.

Note: When concatenating multiple data sets in the SYSIN data definition, you
must ensure that the data set with the largest block size is first in the
concatenation order. Otherwise, the output might not be what you expect.

VM Environment

To create a page definition and form definition that runs PPFA under VM, use the
following command syntax:

```
PPFA fn ft [ fm ] [ ( [PAGEDEF ( ft [ fm A1 ] )]
[FORMDEF ( ft [ fm A1 ] )]
[LISTING ( ft [ fm A1 ] )]
[SIZE nnnn{K|M} ] ) ]
```

PPFA is the command to run PPFA on VM. The file name (fn) is the name of your
file that contains the PPFA control statements. The file name (fn) and file type (ft)
are required parameters. When you specify only the fn and ft, the file mode goes to
your default disk.
The record format of the PPFA input source file is either V or F. The variable record length is a maximum of 100 bytes. For a fixed 80-byte record, the last 8 bytes are treated as comments.

The PPFA command can include any of four optional parameters: PAGEDEF, FORMDEF, LISTING, and SIZE.
- Each keyword parameter can be abbreviated as two letters.
- All parameters in the command can be omitted. However, any optional parameter that follows an open parenthesis must be specified.
- Operands must be enclosed in parentheses when more than one operand is specified for one parameter. Parentheses can be omitted when only one operand is specified for one parameter. Also, the final closing parenthesis can be omitted.
- Any operand string longer than 8 characters is truncated to the first 8 characters.
- Any parameter or operand can be separated from others by parentheses or blanks. The only exceptions are the K and the M operands of a size parameter. For example, in size 256K you cannot separate the 256 from the K.
- The same parameter must not be specified more than once in a command. If duplicate parameters or operands appear, PPFA issues an error message and ends the program.
- For errors associated with a VM execution command, PPFA issues an error message with a return code 20, and does not generate any files (object or listing).
- No optional parameters can follow the open parenthesis that occurs after the input source file ID.
- The size parameter varies according to the size of the command stream. Most command streams do not need a size value because the default specifies enough space for processing. The minimum size is 4K and the maximum size is 16M.

**PAGEDEF Parameter**

PAGEDEF (which can be abbreviated as PA) is the keyword that is used to specify the name of a page-definition resource. (The file type is required; the file mode is optional. If you do not specify a file mode, A1 is assumed.) The page-definition file name is obtained from your input file, and P1 is prefixed to that name.

An example is this command:

```
PPFA PCOM DATA A1 ( PAGEDEF ( PAGEOBJ B1 ) )
```

In the example, the input file, PCOM DATA A1, contains the following control statements:

```
PAGEDEF PAGE1;
PRINTLINE;
FORMDEF FORM1;
```

The result is a page-definition resource file with the file name P1PAGE1, the file type PAGEOBJ, and the file mode B1.

If the page definition parameter is not used, a page-definition resource with the default name P1 (the page definition name from input file) PDEF38PP A1 is created.

The record format of the object file is VM and VA (5A records). 5A records contain the character X’5A’ in the first byte of each record. The record size is up to 8205 bytes.
FORMDEF Parameter

FORMDEF (which can be abbreviated as FO) is the keyword that is used to specify the name of a form-definition resource. (The file type is required; the file mode is optional.) The file name is obtained from your input file, and F1 is prefixed to that name.

An example is this command:

```
PPFA PCOM DATA A1 ( FORMDEF ( FORMOBJ B1 ) )
```

In the example, the input file, PCOM DATA A1, contains the following control statements:

```
PAGEDEF PAGE1;
PRINTLINE;
FORMDEF FORM1;
```

The result is a form-definition resource file with the file name F1FORM1, the file type FORMOBJ, and the file mode B1.

If the form-definition parameter is not used, a form-definition resource with the default name F1 (form-definition name from input file) FDEF38PP A1 is created.

The record format of the object file is VM and VA (5A records). The record size is up to 8205 bytes.

LISTING Parameter

LISTING (which can be abbreviated as LI) is the keyword that is used to specify the name of an output listing file. You can specify the file type and file mode of the resource; the file type is required. If you do not specify a file mode, A1 is assumed. The file name is the same as the PPFA input file name.

For this command example, the result is an output listing file with the name PCOM LISTOUT B1:

```
PPFA PCOM DATA A1 ( LISTING ( LISTOUT B1 )
```

If the LISTING parameter is not used, an output listing file with the default name (PPFA input file name) LISTING A1 is created.

The record format of an output listing file is VA. The record length is 121 bytes (120 bytes + 1 byte (channel control number)). CC numbers are 0 - 12 in the first column of the line data file.

RUN and OPTIONS file

Figure 139 and Figure 140 on page 472 are examples of the VM files that print your data file with the form definition and page definition that you specify.

```
/!*THE ENVIRONMENT IS NOW SET UP TO PRINT */
'CP SP PRT TO NET NOHOLD CLASS A FORM PRT035 COPY 1';
'CP TAG DEV PRT WASVM SYSTEM';
'PSF EXAMP1 PRTDATA A1 ( OPTIONS (EXAMP1) )';
/!*RESTORE THE ENVIRONMENT TO PRINT SOMETHING OTHER THAN THIS EXAMPLE*/
*******************************************************************************
```

Figure 139. VM EXEC Example
FORMDEF ( F1EXAMP1  FDEF38PP ) SEND
PAGEDEF ( P1EXAMP1  PDEF38PP ) SYSDISK
OVERLAY ( *  OVLY38PP ) SYSDISK
* COMMON OPTIONS
CC
NOTRC
BIN 1
CKPTPAGE 0
DATACK UNBLOCK
NODUMP
FILE SEND
FONT ( *  FONT3820 ) SYSDISK
MESSAGES NO
NOOPT
PAGESEG ( *  PSEG38PP ) SYSDISK
TRACE OFF

Figure 140. VM OPTIONS Example
Appendix B. More about Direction

In PPFA, directions that are specified with the PRINTLINE and TRCREF commands are relative to the direction specified in the PAGEFORMAT command. If no PAGEFORMAT command is specified, the direction that is specified in the PAGEDEF command is used. If no direction is specified in either of these commands, the default direction for the page format is ACROSS.

The PRINTLINE and TRCREF commands add their DIRECTION values to the DIRECTION value specified with the PAGEFORMAT command. Thus, you can select a PAGEFORMAT direction and code PRINTLINEs and TRCREFs relative to the PAGEFORMAT direction. For more information about the PRINTLINE and TRCREF commands, see Chapter 3, “Using Page Definition Commands for Traditional Line Data,” on page 33.

For instance, if a page is to be printed in the landscape page presentation on a printer that requires the DOWN or UP print direction to generate landscape output, the PAGEFORMAT command can specify DOWN as its DIRECTION. When this direction is established, you can view the page as a landscape page and specify the PRINTLINE and the TRCREF commands with the ACROSS direction. Output that is specified in this way prints ACROSS relative to the landscape page, as shown in Figure 141.

![Figure 141. Printing Across a Landscape Page](image)

If you specify the DOWN direction for the PRINTLINE or the TRCREF command in this case, the output looks like Figure 142 on page 474 because the direction of the page format is also DOWN.
Table 19 shows the final result when all of the possible combinations of DIRECTION are specified. The final direction that PPFA computes from the PAGEFORMAT, PRINTLINE, and TRCREF commands determines the prefix that is assigned to the font names specified in the page definition. The final direction is important when printing on the 3800 printer because its unbounded-box font architecture requires a separate font for each combination of print direction and character rotation. This information is encoded in the prefix of the font name (X1, X3, XA, and XF, for example).

**Table 19. The Effect of Additive DIRECTIONs on Formatting and Font Prefixes**

<table>
<thead>
<tr>
<th>Page Format</th>
<th>PRINTLINE or TRCREF</th>
<th>Final Result</th>
<th>3800 Font Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0°</td>
<td>90°</td>
</tr>
<tr>
<td>Across</td>
<td>Across</td>
<td>Across</td>
<td>X1</td>
</tr>
<tr>
<td>Across</td>
<td>Down</td>
<td>Down</td>
<td>X2</td>
</tr>
<tr>
<td>Across</td>
<td>Back</td>
<td>Back</td>
<td>X3</td>
</tr>
<tr>
<td>Across</td>
<td>Up</td>
<td>Up</td>
<td>X4</td>
</tr>
<tr>
<td>Down</td>
<td>Across</td>
<td>Down</td>
<td>X2</td>
</tr>
<tr>
<td>Down</td>
<td>Down</td>
<td>Back</td>
<td>X3</td>
</tr>
<tr>
<td>Down</td>
<td>Up</td>
<td>Up</td>
<td>X4</td>
</tr>
<tr>
<td>Down</td>
<td>Across</td>
<td>Back</td>
<td>X3</td>
</tr>
<tr>
<td>Back</td>
<td>Across</td>
<td>Back</td>
<td>X3</td>
</tr>
<tr>
<td>Back</td>
<td>Down</td>
<td>Up</td>
<td>X4</td>
</tr>
<tr>
<td>Back</td>
<td>Back</td>
<td>Across</td>
<td>X1</td>
</tr>
<tr>
<td>Back</td>
<td>Up</td>
<td>Down</td>
<td>X2</td>
</tr>
<tr>
<td>Up</td>
<td>Across</td>
<td>Up</td>
<td>X4</td>
</tr>
<tr>
<td>Up</td>
<td>Down</td>
<td>Across</td>
<td>X1</td>
</tr>
<tr>
<td>Up</td>
<td>Back</td>
<td>Down</td>
<td>X2</td>
</tr>
<tr>
<td>Up</td>
<td>Up</td>
<td>Back</td>
<td>X3</td>
</tr>
</tbody>
</table>

The entries in the Final Result column can be computed by using a simple algorithm. If you assume that ACROSS is 0, DOWN is 1, BACK is 2, and UP is 3, you can add the direction specifications in the two commands, subtracting 4 when the result is 4 or greater, to compute the final direction.
Appendix C. Differences in Measurements and REPEATs with AFP Utilities

When you repeat a DRAWRULE (OGL), PRINTLINE (PPFA), DRAWGRAPHIC (PPFA), or Line (PMF), differences exist in the measurements of the repeated lines. For OGL, REPEAT indicates the number of repetitions in addition to the first. For DRAWGRAPHIC (PPFA), REPEAT is the same as OGL. Therefore, REPEAT yields 2 DRAWRULEs. For PPFA, REPEAT indicates the total number of PRINTLINEs. Therefore, REPEAT yields 2 PRINTLINEs.

Another difference occurs when the line spacing (set by SETUNITS in OGL and PPFA, and by a screen item in PMF) results in the distance from one line to the next not being a whole number of pels. Each product handles the fractional pel differently. Because the printer cannot print parts of a pel, fractional pels cannot be represented at the printer. When line spacing calculations result in a fractional pel per line space, these occur:

**OGL** Carries the fractions until they add up to a whole pel, then adds it in. The result is that the final spot of a repeat is within a pel of where it is expected. Therefore, not all of the spaces between rules are even; they can vary by 1 pel.

**PPFA** Truncates the fractional pel before the repeat. Therefore, the spaces between the lines are even, but the total might be shorter than expected.

**PMF** Rounds the fractional pel before the repeat. Therefore, the spaces between the lines are even, but the total might be shorter or longer than expected. If the fractional pel is less than 0.5, it is handled the same as PPFA and the line space is shorter. If the fractional pel is greater than or equal to 0.5, the line space is longer.

Use line spacing in all products that result in a whole number of pels. To resolve existing problems, select the resource that you don’t want to change, and code the remaining resource without using REPEAT because of the way the other products handle the fractional pels.

The following example shows what occurs if you want to print at 9 lines per inch, and repeat it for 20 lines. Starting at zero, and adding 9 lines per inch (converted to pels it is 240/9 = 26.6670), you see the results that are illustrated in Table 20.

<table>
<thead>
<tr>
<th>Repetition</th>
<th>Mathematics</th>
<th>OGL</th>
<th>PPFA</th>
<th>PMF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Position</td>
<td>FromLast</td>
<td>Position</td>
<td>FromLast</td>
</tr>
<tr>
<td>0.000</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>26.667</td>
<td>26.667</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>53.333</td>
<td>26.667</td>
<td>53</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>80.000</td>
<td>26.667</td>
<td>80</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>106.667</td>
<td>26.667</td>
<td>106</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>133.333</td>
<td>26.667</td>
<td>133</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>160.000</td>
<td>26.667</td>
<td>160</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>186.667</td>
<td>26.667</td>
<td>186</td>
<td>26</td>
</tr>
</tbody>
</table>
Table 20. Differences in Measurements and REPEATs with AFP Utilities (continued)

<table>
<thead>
<tr>
<th>Repetition</th>
<th>Mathematics Position</th>
<th>OGL Position</th>
<th>PPFA Position</th>
<th>PMF Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>213.333</td>
<td>213</td>
<td>208</td>
<td>216</td>
</tr>
<tr>
<td>9</td>
<td>240.000</td>
<td>240</td>
<td>234</td>
<td>243</td>
</tr>
<tr>
<td>10</td>
<td>266.667</td>
<td>266</td>
<td>260</td>
<td>270</td>
</tr>
<tr>
<td>11</td>
<td>293.333</td>
<td>293</td>
<td>286</td>
<td>297</td>
</tr>
<tr>
<td>12</td>
<td>320.000</td>
<td>320</td>
<td>312</td>
<td>324</td>
</tr>
<tr>
<td>13</td>
<td>346.667</td>
<td>346</td>
<td>338</td>
<td>351</td>
</tr>
<tr>
<td>14</td>
<td>373.333</td>
<td>373</td>
<td>364</td>
<td>378</td>
</tr>
<tr>
<td>15</td>
<td>400.000</td>
<td>400</td>
<td>390</td>
<td>405</td>
</tr>
<tr>
<td>16</td>
<td>426.667</td>
<td>426</td>
<td>416</td>
<td>432</td>
</tr>
<tr>
<td>17</td>
<td>453.333</td>
<td>453</td>
<td>442</td>
<td>459</td>
</tr>
<tr>
<td>18</td>
<td>480.000</td>
<td>480</td>
<td>468</td>
<td>486</td>
</tr>
<tr>
<td>19</td>
<td>506.667</td>
<td>506</td>
<td>494</td>
<td>513</td>
</tr>
<tr>
<td>20</td>
<td>533.333</td>
<td>533</td>
<td>520</td>
<td>540</td>
</tr>
</tbody>
</table>

To resolve differences in how OGL, PPFA, and PMF handle repeated values, one of these approaches can be taken:

- Do not use REPEAT.
- Code units as PELs.

Note: In all of these products (except PPFA), a PEL is 1/240 of an inch. For PPFA, the PEL size can be set by the user, but defaults to 1/240 of an inch.
Appendix D. More About Bar Code Parameters

This section contains supplemental information about Bar Code Object Content Architecture (BCOCA) specified by the BARCODE subcommand of the FIELD command, and includes the following topics:

- Bar code data
- MOD parameter

For more complete information, see Bar Code Object Content Architecture Reference.

Bar Code Data

The data is specified as a series of single-byte code points from a specific code page. Some symbologies limit the valid code points to just the 10 numerals (0 - 9), other symbologies allow a richer set of code points. The bar code symbol is produced from these code points; the code points are also used, along with a particular type style, when HRI is produced.

Table 21 lists, for each symbology, the valid code page from which characters are chosen and the type style that is used when HRI is printed in terms of a registered CPGID and FGID. More information about these values can be found in IBM AFP Fonts: Font Summary and in IBM AFP Fonts: Technical Reference for Code Pages.

Table 21. Valid Code Pages and Type Styles

<table>
<thead>
<tr>
<th>Type</th>
<th>Bar Code Symbology</th>
<th>EBCDIC-Based CPGID</th>
<th>FGID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>MSI (modified Plessey code)</td>
<td>500</td>
<td>Device specific</td>
</tr>
<tr>
<td>3</td>
<td>UPC/CGPC — Version A</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>5</td>
<td>UPC/CGPC — Version E</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>6</td>
<td>UPC — Two-digit Supplemental (Periodicals)</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>7</td>
<td>UPC — Five-digit Supplemental (Paperbacks)</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>8</td>
<td>EAN-8 (includes JAN-short)</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>9</td>
<td>EAN-13 (includes JAN-standard)</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>10</td>
<td>Industrial 2-of-5</td>
<td>500</td>
<td>Device specific</td>
</tr>
<tr>
<td>11</td>
<td>Matrix 2-of-5</td>
<td>500</td>
<td>Device specific</td>
</tr>
<tr>
<td>12</td>
<td>Interleaved 2-of-5, AIM USS-1 2/5</td>
<td>500</td>
<td>Device specific</td>
</tr>
<tr>
<td>13</td>
<td>Codabar, 2-of-7, AIM USS-Codabar</td>
<td>500</td>
<td>Device specific</td>
</tr>
<tr>
<td>17</td>
<td>Code 128, AIM USS-128</td>
<td>1303</td>
<td>Device specific</td>
</tr>
<tr>
<td>22</td>
<td>EAN Two-digit Supplemental</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>23</td>
<td>EAN Five-digit Supplemental</td>
<td>893</td>
<td>3 (OCR-B)</td>
</tr>
<tr>
<td>24</td>
<td>POSTNET</td>
<td>500</td>
<td>None</td>
</tr>
<tr>
<td>26</td>
<td>RM4CC</td>
<td>500</td>
<td>None</td>
</tr>
<tr>
<td>27</td>
<td>Japan Postal Bar Code</td>
<td>500</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 21. Valid Code Pages and Type Styles (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Bar Code Symbology</th>
<th>EBCDIC-Based CPGID</th>
<th>FGID</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>Data Matrix (2D bar code)</td>
<td>Code page is selectable within the symbol by using ECI protocol</td>
<td>None</td>
</tr>
<tr>
<td>29</td>
<td>MaxiCode (2D bar code)</td>
<td>Code page is selectable within the symbol by using ECI protocol</td>
<td>None</td>
</tr>
<tr>
<td>30</td>
<td>PDF417 (2D bar code)</td>
<td>Code page is selectable within the symbol by using ECI protocol</td>
<td>None</td>
</tr>
<tr>
<td>31</td>
<td>Australia Post Bar Code</td>
<td>Code page is selectable within the symbol by using ECI protocol</td>
<td>Device Specific</td>
</tr>
<tr>
<td>32</td>
<td>QR Code</td>
<td>Code page is selectable within the symbol by using ECI protocol</td>
<td>None</td>
</tr>
<tr>
<td>33</td>
<td>Code 93</td>
<td>500</td>
<td>Device Specific</td>
</tr>
<tr>
<td>34</td>
<td>USPS Four-State</td>
<td>500</td>
<td>Device Specific</td>
</tr>
</tbody>
</table>

As shown in Table 21 on page 477, the font that is used to print HRI depends on the symbology. Some symbologies use OCR-B; others use a device-specific font (typically OCR-A).

Table 22 lists the valid characters for each symbology and specifies how many characters are allowed for a bar code symbol.

Table 22. Valid Characters and Data Lengths

<table>
<thead>
<tr>
<th>Code</th>
<th>Bar Code Type</th>
<th>Valid Characters</th>
<th>Valid Data Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A total of 43 valid input characters.</td>
<td>BCOCa range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td>2</td>
<td>MSI (modified Plessey code)</td>
<td>0123456789</td>
<td>3 - 15 characters for Modifier X'01'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 - 14 characters for Modifier X'02'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - 13 characters for all other modifiers</td>
</tr>
<tr>
<td>3</td>
<td>UPC/CGPC - Version A</td>
<td>0123456789</td>
<td>11 characters</td>
</tr>
<tr>
<td>Code</td>
<td>Bar Code Type</td>
<td>Valid Characters</td>
<td>Valid Data Length</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td>UPC/CGPC - Version E</td>
<td>0123456789</td>
<td>10 characters</td>
</tr>
<tr>
<td>6</td>
<td>UPC - Two-digit Supplemental (Periodicals)</td>
<td>0123456789</td>
<td>2 characters for Modifier X'00'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13 characters for Modifier X'01'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12 characters for Modifier X'02'</td>
</tr>
<tr>
<td>7</td>
<td>UPC - Five-digit Supplemental (Paperbacks)</td>
<td>0123456789</td>
<td>5 characters for Modifier X'00'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16 characters for Modifier X'01'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15 characters for Modifier X'02'</td>
</tr>
<tr>
<td>8</td>
<td>EAN-8 (includes JAN-short)</td>
<td>0123456789</td>
<td>7 characters</td>
</tr>
<tr>
<td>9</td>
<td>EAN-13 (includes JAN-standard)</td>
<td>0123456789</td>
<td>12 characters</td>
</tr>
<tr>
<td>10</td>
<td>Industrial 2-of-5</td>
<td>0123456789</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td>11</td>
<td>Matrix 2-of-5</td>
<td>0123456789</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td>12</td>
<td>Interleaved 2-of-5, AIM USS-1 2/5</td>
<td>0123456789</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td>13</td>
<td>Codabar, 2-of-7, AIM USS-Codabar</td>
<td>0123456789</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16 characters plus 4 start/stop characters (ABCD) (Note 2 on page 482)</td>
</tr>
<tr>
<td>17</td>
<td>Code 128, AIM USS-128 (modifier X'02')</td>
<td>All characters that are defined in the Code 128 code page</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td></td>
<td>UCC/EAN 128 (modifiers X'03' and X'04')</td>
<td>0123456789</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklm nopqrstuvwxyz FNC1</td>
</tr>
<tr>
<td>22</td>
<td>EAN Two-digit Supplemental</td>
<td>0123456789</td>
<td>2 characters for Modifier X'00'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14 characters for Modifier X'01'</td>
</tr>
<tr>
<td>23</td>
<td>EAN Five-digit Supplemental</td>
<td>0123456789</td>
<td>5 characters for Modifier X'00'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17 characters for Modifier X'01'</td>
</tr>
</tbody>
</table>
### Table 22. Valid Characters and Data Lengths (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Bar Code Type</th>
<th>Valid Characters</th>
<th>Valid Data Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>POSTNET</td>
<td>0123456789</td>
<td>5 characters for Modifier X'00'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9 characters for Modifier X'01'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11 characters for Modifier X'02'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11 characters for Modifier X'04'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range for Modifier X'03': 0 - 50 characters (see note [1 on page 482]</td>
</tr>
<tr>
<td>26</td>
<td>Royal Mail (RM4SCC, modifier X'00')</td>
<td>0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ - (hyphen)</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note [1 on page 482]</td>
</tr>
<tr>
<td></td>
<td>Royal Mail (Dutch KIX variation, modifier X'01')</td>
<td>0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ - (hyphen)</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note [1 on page 482]</td>
</tr>
<tr>
<td>27</td>
<td>Japan Postal Bar Code (Modifier X'00')</td>
<td>0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ - (hyphen)</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note [1 on page 482]</td>
</tr>
<tr>
<td></td>
<td>Japan Postal Bar Code (Modifier X'01')</td>
<td>0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ - (hyphen)</td>
<td>No length checking done; refer to the modifier X'01' description.</td>
</tr>
<tr>
<td>28</td>
<td>Data Matrix</td>
<td>Any 1-byte character or binary data</td>
<td>Symbology: up to 3116 depending on whether the data is character or numeric; refer to the symbology specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 3116 characters (see note [1 on page 482]</td>
</tr>
<tr>
<td>29</td>
<td>MaxiCode</td>
<td>Any 1-byte character that is allowed by the symbol mode</td>
<td>Symbology: up to 93 alphanumeric characters per symbol depending on encoding overhead or up to 138 numeric characters per symbol; refer to the symbology specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 138 characters</td>
</tr>
<tr>
<td>30</td>
<td>PDF417</td>
<td>Any 1-byte character or binary data</td>
<td>Symbology: up to 1850 text characters, 2710 ASCII numeric digits, or 1108 bytes of binary data per symbol depending on the security level; refer to the symbology specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 2710 characters</td>
</tr>
</tbody>
</table>
### Table 22. Valid Characters and Data Lengths (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Bar Code Type</th>
<th>Valid Characters</th>
<th>Valid Data Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Australia Post Bar Code –</td>
<td>Refer to the modifier (byte 13) description to see which characters are valid in specific parts of the symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifier X'01' – Standard Customer Barcode</td>
<td>0123456789</td>
<td>Symbology: 8 digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 digits</td>
</tr>
<tr>
<td></td>
<td>Modifier X'02' – Customer Barcode 2 using Table N</td>
<td>0123456789</td>
<td>Symbology: 8 - 16 digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 - 16 digits</td>
</tr>
<tr>
<td></td>
<td>Modifier X'03' – Customer Barcode 2 using Table C</td>
<td>0123456789</td>
<td>Symbology: 8 - 13 characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 - 13 characters</td>
</tr>
<tr>
<td></td>
<td>Modifier X'04' – Customer Barcode 2 using proprietary encoding</td>
<td>0123456789 for sorting code 0 - 3 for customer information</td>
<td>Symbology: 8 - 24 digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 - 24 digits</td>
</tr>
<tr>
<td></td>
<td>Modifier X'05' – Customer Barcode 3 using Table N</td>
<td>0123456789</td>
<td>Symbology: 8 - 23 digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 - 23 digits</td>
</tr>
<tr>
<td></td>
<td>Modifier X'06' – Customer Barcode 3 using Table C</td>
<td>0123456789</td>
<td>Symbology: 8 - 18 characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 - 18 characters</td>
</tr>
<tr>
<td></td>
<td>Modifier X'07' – Customer Barcode 3 using proprietary encoding</td>
<td>0123456789 for sorting code 0 - 3 for customer information</td>
<td>Symbology: 8 - 39 digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 - 39 digits</td>
</tr>
<tr>
<td></td>
<td>Modifier X'08' – Reply Paid Barcode</td>
<td>0123456789</td>
<td>Symbology: 8 digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 8 digits</td>
</tr>
<tr>
<td>32</td>
<td>QR Code</td>
<td>Any 1-byte character or binary data</td>
<td>Symbology: Up to 7,089 characters depending on the size and type of the data; refer to the symbology specification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 7,089 characters</td>
</tr>
<tr>
<td>33</td>
<td>Code 93</td>
<td>0123456789</td>
<td>Symbology: unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BCOCA range: 0 - 50 characters (see note 1 on page 482)</td>
</tr>
<tr>
<td>34</td>
<td>USPS Four-State</td>
<td>0123456789</td>
<td>20 digits for Modifier X'00'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25 digits for Modifier X'01'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>29 digits for Modifier X'02'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>31 digits for Modifier X'03'</td>
</tr>
</tbody>
</table>
Table 22. Valid Characters and Data Lengths (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Bar Code Type</th>
<th>Valid Characters</th>
<th>Valid Data Length</th>
</tr>
</thead>
</table>

Notes:
1. All BCOCA receivers must support at least the BCOCA range. Some receivers support a larger data length.
2. Some descriptions of Codabar show the characters T, N, *, E as stop characters (representing the stop characters A, B, C, D), but the Codabar symbology allows only A, B, C, D as start and stop characters. This alternative representation (T, N, *, E) is used only to distinguish between the start and stop characters when a Codabar symbol is described; when a BCOCA Codabar symbol is coded, start and stop characters must be represented with A, B, C, or D.
3. The data for the UPC and EAN symbologies is numeric and of a fixed length, but not all numbers of the appropriate length are valid. This is because the coding scheme is designed to uniquely identify both a product and its manufacturer. The first part of the symbol represents the manufacturer and is defined in the symbology specification (not all numbers are valid in this part of the symbol). The second part of the symbol represents a unique product identifier code that is assigned by the manufacturer. See the appropriate symbology specification for more details.

Table 23. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128

<table>
<thead>
<tr>
<th>Character</th>
<th>EBCDIC Code Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'F0'</td>
</tr>
<tr>
<td>1</td>
<td>'F1'</td>
</tr>
<tr>
<td>2</td>
<td>'F2'</td>
</tr>
<tr>
<td>3</td>
<td>'F3'</td>
</tr>
<tr>
<td>4</td>
<td>'F4'</td>
</tr>
<tr>
<td>5</td>
<td>'F5'</td>
</tr>
<tr>
<td>6</td>
<td>'F6'</td>
</tr>
<tr>
<td>7</td>
<td>'F7'</td>
</tr>
<tr>
<td>8</td>
<td>'F8'</td>
</tr>
<tr>
<td>9</td>
<td>'F9'</td>
</tr>
<tr>
<td>A</td>
<td>'C1'</td>
</tr>
<tr>
<td>B</td>
<td>'C2'</td>
</tr>
<tr>
<td>C</td>
<td>'C3'</td>
</tr>
<tr>
<td>D</td>
<td>'C4'</td>
</tr>
<tr>
<td>E</td>
<td>'C5'</td>
</tr>
<tr>
<td>F</td>
<td>'C6'</td>
</tr>
<tr>
<td>G</td>
<td>'C7'</td>
</tr>
<tr>
<td>H</td>
<td>'C8'</td>
</tr>
<tr>
<td>I</td>
<td>'C9'</td>
</tr>
<tr>
<td>J</td>
<td>'D1'</td>
</tr>
<tr>
<td>K</td>
<td>'D2'</td>
</tr>
<tr>
<td>L</td>
<td>'D3'</td>
</tr>
<tr>
<td>M</td>
<td>'D4'</td>
</tr>
<tr>
<td>N</td>
<td>'D5'</td>
</tr>
<tr>
<td>O</td>
<td>'D6'</td>
</tr>
<tr>
<td>P</td>
<td>'D7'</td>
</tr>
<tr>
<td>Q</td>
<td>'D8'</td>
</tr>
</tbody>
</table>
Table 23. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128 (continued)

<table>
<thead>
<tr>
<th>Character</th>
<th>EBCDIC Code Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>X'D9'</td>
</tr>
<tr>
<td>S</td>
<td>X'E2'</td>
</tr>
<tr>
<td>T</td>
<td>X'E3'</td>
</tr>
<tr>
<td>U</td>
<td>X'E4'</td>
</tr>
<tr>
<td>V</td>
<td>X'E5'</td>
</tr>
<tr>
<td>W</td>
<td>X'E6'</td>
</tr>
<tr>
<td>X</td>
<td>X'E7'</td>
</tr>
<tr>
<td>Y</td>
<td>X'E8'</td>
</tr>
<tr>
<td>Z</td>
<td>X'E9'</td>
</tr>
<tr>
<td>a</td>
<td>X'81'</td>
</tr>
<tr>
<td>b</td>
<td>X'82'</td>
</tr>
<tr>
<td>c</td>
<td>X'83'</td>
</tr>
<tr>
<td>d</td>
<td>X'84'</td>
</tr>
<tr>
<td>e</td>
<td>X'85'</td>
</tr>
<tr>
<td>f</td>
<td>X'86'</td>
</tr>
<tr>
<td>g</td>
<td>X'87'</td>
</tr>
<tr>
<td>h</td>
<td>X'88'</td>
</tr>
<tr>
<td>i</td>
<td>X'89'</td>
</tr>
<tr>
<td>j</td>
<td>X'91'</td>
</tr>
<tr>
<td>k</td>
<td>X'92'</td>
</tr>
<tr>
<td>l</td>
<td>X'93'</td>
</tr>
<tr>
<td>m</td>
<td>X'94'</td>
</tr>
<tr>
<td>n</td>
<td>X'95'</td>
</tr>
<tr>
<td>o</td>
<td>X'96'</td>
</tr>
<tr>
<td>p</td>
<td>X'97'</td>
</tr>
<tr>
<td>q</td>
<td>X'98'</td>
</tr>
<tr>
<td>r</td>
<td>X'99'</td>
</tr>
<tr>
<td>s</td>
<td>X'A2'</td>
</tr>
<tr>
<td>t</td>
<td>X'A3'</td>
</tr>
<tr>
<td>u</td>
<td>X'A4'</td>
</tr>
<tr>
<td>v</td>
<td>X'A5'</td>
</tr>
<tr>
<td>w</td>
<td>X'A6'</td>
</tr>
<tr>
<td>x</td>
<td>X'A7'</td>
</tr>
<tr>
<td>y</td>
<td>X'A8'</td>
</tr>
<tr>
<td>z</td>
<td>X'A9'</td>
</tr>
<tr>
<td>- (hyphen)</td>
<td>X'60'</td>
</tr>
<tr>
<td># (number sign)</td>
<td>X'7B'</td>
</tr>
<tr>
<td>. (period)</td>
<td>X'4B'</td>
</tr>
<tr>
<td>$</td>
<td>X'5B'</td>
</tr>
</tbody>
</table>
Table 23. Characters and Code Points used in the BCOCA Symbologies; Excluding Code 128 (continued)

<table>
<thead>
<tr>
<th>Character</th>
<th>EBCDIC Code Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>X'61'</td>
</tr>
<tr>
<td>+</td>
<td>X'4E'</td>
</tr>
<tr>
<td>%</td>
<td>X'6C'</td>
</tr>
<tr>
<td>:</td>
<td>X'7A'</td>
</tr>
<tr>
<td>Space</td>
<td>X'40'</td>
</tr>
<tr>
<td>FNC1</td>
<td>X'8F'</td>
</tr>
</tbody>
</table>

The Code 128 code page (CPGID = 1303) is defined as shown in Figure 143 on page 485.
<table>
<thead>
<tr>
<th>HEX DIGITS</th>
<th>0 -</th>
<th>1 -</th>
<th>2 -</th>
<th>3 -</th>
<th>4 -</th>
<th>5 -</th>
<th>6 -</th>
<th>7 -</th>
<th>8 -</th>
<th>9 -</th>
<th>A -</th>
<th>B -</th>
<th>C -</th>
<th>D -</th>
<th>E -</th>
<th>F -</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>SP</td>
<td>-</td>
<td>^</td>
<td>(</td>
<td>)</td>
<td>\</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>SOH</td>
<td>DC1</td>
<td>/</td>
<td>a</td>
<td>j</td>
<td>A</td>
<td>J</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>STX</td>
<td>DC2</td>
<td>b</td>
<td>k</td>
<td>s</td>
<td>B</td>
<td>K</td>
<td>S</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>ETX</td>
<td>DC3</td>
<td>c</td>
<td>l</td>
<td>t</td>
<td>C</td>
<td>L</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>HT</td>
<td>LF</td>
<td>d</td>
<td>m</td>
<td>u</td>
<td>D</td>
<td>M</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td>BS</td>
<td>ETB</td>
<td>e</td>
<td>n</td>
<td>v</td>
<td>E</td>
<td>N</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6</td>
<td>ESC</td>
<td>EOT</td>
<td>f</td>
<td>o</td>
<td>w</td>
<td>F</td>
<td>O</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-7</td>
<td>CAN</td>
<td></td>
<td>g</td>
<td>p</td>
<td>x</td>
<td>G</td>
<td>P</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-8</td>
<td></td>
<td></td>
<td>h</td>
<td>q</td>
<td>y</td>
<td>H</td>
<td>Q</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-9</td>
<td></td>
<td></td>
<td>i</td>
<td>r</td>
<td>z</td>
<td>I</td>
<td>R</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-A</td>
<td>VT</td>
<td></td>
<td>!</td>
<td>:</td>
<td>;</td>
<td>[</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-B</td>
<td></td>
<td></td>
<td>$</td>
<td>#</td>
<td></td>
<td>]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-C</td>
<td>FF</td>
<td>DC4</td>
<td>&lt;</td>
<td>*</td>
<td>%</td>
<td>@</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-D</td>
<td>CR</td>
<td>GS</td>
<td>&gt;</td>
<td>NAK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-E</td>
<td>SO</td>
<td>RS</td>
<td>?</td>
<td>SUB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-F</td>
<td>SI</td>
<td>US</td>
<td>BEL</td>
<td>SUB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** All START, STOP, SHIFT, and CODE characters are generated by the printer to produce the shortest bar code possible from the specified data; these characters are not specified in the Bar Code Symbol Data. All code points not listed in the table are undefined. The code points that do not have graphic character shapes, such as X'00' (NUL) and X'8F' (FN1), are control codes that are defined within the Code 128 symbology; in the HRI, control codes print in a device-dependent manner. The FN1, FN2, FN3, and FN4 characters are also called FNC1, FNC2, FNC3, and FNC4 in the Code 128 Symbology Specification.

**MOD Parameter**

The modifier field gives more processing information about the bar code symbol to be generated. For example, it indicates whether a check-digit is to be generated for the bar code symbol.

Table 24 on page 486 shows the modifier values for each bar code type.
Table 24. Modifier Values by Bar Code Type

<table>
<thead>
<tr>
<th>Bar Code Type</th>
<th>MOD Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Code 39 (3-of-9 Code), AIM USS-39</td>
<td>X'01' and X'02'</td>
</tr>
<tr>
<td>2 – MSI (modified Plessey code)</td>
<td>X'01' - X'09'</td>
</tr>
<tr>
<td>3 – UPC/CGPC Version A</td>
<td>X'00'</td>
</tr>
<tr>
<td>5 – UPC/CGPC Version E</td>
<td>X'00'</td>
</tr>
<tr>
<td>6 – UPC - Two-digit Supplemental</td>
<td>X'00' - X'02'</td>
</tr>
<tr>
<td>7 – UPC - Five-digit Supplemental</td>
<td>X'00' - X'02'</td>
</tr>
<tr>
<td>8 – EAN 8 (includes JAN-short)</td>
<td>X'00'</td>
</tr>
<tr>
<td>9 – EAN 13 (includes JAN-standard)</td>
<td>X'00'</td>
</tr>
<tr>
<td>10 – Industrial 2-of-5</td>
<td>X'01' and X'02'</td>
</tr>
<tr>
<td>11 – Matrix 2-of-5</td>
<td>X'01' and X'02'</td>
</tr>
<tr>
<td>12 – Interleaved 2-of-5, AIM USS-I 2/5</td>
<td>X'01' - X'04'</td>
</tr>
<tr>
<td>13 – Codabar, 2-of-7, AIM USS-Codabar</td>
<td>X'01' and X'02'</td>
</tr>
<tr>
<td>17 – Code 128, AIM USS-128</td>
<td>X'02' - X'05'</td>
</tr>
<tr>
<td>22 – EAN Two-digit Supplemental</td>
<td>X'00' and X'01'</td>
</tr>
<tr>
<td>23 – EAN Five-digit Supplemental</td>
<td>X'00' and X'01'</td>
</tr>
<tr>
<td>24 – POSTNET</td>
<td>X'00' - X'04'</td>
</tr>
<tr>
<td>26 – RM4SCC</td>
<td>X'00' and X'01'</td>
</tr>
<tr>
<td>27 – Japan Postal Bar Code</td>
<td>X'00' and X'01'</td>
</tr>
<tr>
<td>28 – DataMatrix two-dimensional Bar Code</td>
<td>X'00'</td>
</tr>
<tr>
<td>29 – MaxiCode two-dimensional Bar Code</td>
<td>X'00'</td>
</tr>
<tr>
<td>30 – PDF417 two-dimensional Bar Code</td>
<td>X'00' and X'01'</td>
</tr>
<tr>
<td>31 – Australia Postal Bar Code</td>
<td>X'01' - X'08'</td>
</tr>
<tr>
<td>32 – QR CODE two-dimensional Bar Code</td>
<td>X'02'</td>
</tr>
<tr>
<td>33 – CODE 93</td>
<td>X'00'</td>
</tr>
<tr>
<td>34 – USPS Four-State</td>
<td>X'00' - X'03'</td>
</tr>
<tr>
<td>35 – Red Tag</td>
<td>X'00'</td>
</tr>
<tr>
<td>36 – GS1 DataBar</td>
<td>X'00' - X'04' and X'11' - X'1B'</td>
</tr>
</tbody>
</table>

The modifier values, by bar code type, are as follows:

**Code 39 (3-of-9 Code), AIM USS-39**

X'01' Present the bar code without a generated check digit.

X'02' Generate a check digit and present it with the bar code.

**Note:** The Code 39 character set contains 43 characters that include numbers, uppercase alphabetics, and some special characters. The Code 39
Specification also provides a method of encoding all 128 ASCII characters by using 2 bar code characters for those ASCII characters that are not in the standard Code 39 character set. This is sometimes referred to as Extended Code 39 and is supported by all BCOCA receivers. In this case, the 2 bar code characters that are used to specify the extended character are shown in the Human-Readable Interpretation and the bar code scanner interprets the 2-character combination bar/space pattern.

**MSI (modified Plessey code)**

![MSI - no check digit](encoding 80523)

X'01' Present the bar code without check digits that are generated by the printer. Specify 3 - 15 digits of input data.

X'02' Present the bar code with a generated IBM modulo-10 check digit. This check digit is the second check digit; the first check digit is the last character of the data as defined in the associated FIELD START and LENGTH subcommands. Specify 2 - 14 digits of input data.

X'03' Present the bar code with two check digits. Both check digits are generated by using the IBM modulo-10 algorithm. Specify 1 - 13 digits of input data.

X'04' Present the bar code with two check digits. The first check digit is generated by using the NCR modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10. Specify 1 - 13 digits of input data.

X'05' Present the bar code with two check digits. The first check digit is generated by using the IBM modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10. Specify 1 - 13 digits of input data.

X'06' Present the bar code with two check digits. The first check digit is generated by using the NCR modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; a first check digit value of 10 is assigned the value zero. Specify 1 - 13 digits of input data.

X'07' Present the bar code with two check digits. The first check digit is generated by using the IBM modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; a first check digit value of 10 is assigned the value zero. Specify 1 - 13 digits of input data.

X'08' Present the bar code with two check digits. The first check digit is
generated by using the NCR modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10. Specify 1 - 13 digits of input data.

X'09' Present the bar code with two check digits. The first check digit is generated by using the IBM modulo-11 algorithm; the second using the IBM modulo-10 algorithm. The first check digit equals 11 minus the remainder; exception condition EC-0E00 exists if the first check-digit calculation results in a value of 10. Specify 1 - 13 digits of input data.

UPC/CGPC—Version A

UPC Version A
(encoding 01234567890)

X'00' Present the standard UPC-A bar code with a generated check digit. The data to be encoded consists of 11 digits. The first digit is the number-system digit; the next 10 digits are the article number.

Specify 11 digits of input data. The first digit is the number system character; the remaining digits are information characters.

UPC/CGPC—Version E

UPC Version E
(encoding 078349)

X'00' Present a UPC-E bar code symbol. Of the 10 input digits, 6 digits are encoded. The check digit is generated by using all 10 input data digits. The check digit is not encoded; it is only used to assign odd or even parity to the six encoded digits.

Specify 10 digits of input data. Version E suppresses some zeros that can occur in the information characters to produce a shorter symbol. All 10 digits are information characters; the number system character must not be specified (it is assumed to be 0).
UPC—Two-Digit Supplemental

```
  0 6 3 8 9 5 2 6 0 8

UPC A + Two-digit Supplemental
(encoding 0063895260, supplemental = 24)
```

**X'00'** Present a UPC two-digit supplemental bar code symbol. This option assumes that the base UPC Version A or E symbol is presented as a separate bar code object. The bar and space patterns that are used for the two supplemental digits are left-odd or left-even parity, with the parity determined by the digit combination.

Specify 2 digits of input data.

**X'01'** The two-digit UPC supplemental bar code symbol is preceded by a UPC Version A, Number System 0, bar code symbol. The bar code object contains both the UPC Version A symbol and the two-digit supplemental symbol. The input data consists of the number system digit, the 10-digit article number, and the two supplement digits, in that order. A check digit is generated for the UPC Version A symbol. The two-digit supplemental bar code is presented after the UPC Version A symbol by using left odd and even parity as determined by the two supplemental digits.

Specify 13 digits of input data.

**X'02'** The two-digit UPC supplemental bar code symbol is preceded by a UPC Version E symbol. The bar code object contains both the UPC Version E symbol and the two-digit supplemental symbol. The input data consists of the 10-digit article number and the two supplemental digits. The bar code object processor generates the six-digit UPC Version E symbol and a check digit. The check digit is used to determine the parity pattern of the six-digit Version E symbol. The two-digit supplemental bar code symbol is presented after the Version E symbol by using left odd and even parity as determined by the two digits.

Specify 12 digits of input data.

UPC—Five-Digit Supplemental

```
  0 9 8 2 7 7 2 1 1 2 3 6

UPC A + Five-digit Supplemental
(encoding 09827721123, supplemental = 21826)
```

**X'00'** Present the UPC five-digit supplemental bar code symbol. This option assumes that the base UPC Version A or E symbol is presented as a separate bar code object. A check digit is generated
from the five supplemental digits and is used to assign the left-odd and left-even parity of the five-digit supplemental bar code. The supplemental check digit is not encoded or interpreted.

Specify 5 digits of input data.

X'01' The five-digit UPC supplemental bar code symbol is preceded by a UPC Version A, Number System 0, bar code symbol. The bar code object contains both the UPC Version A symbol and the five-digit supplemental symbol. The input data consists of the number system digit, the 10-digit article number, and the five supplement digits, in that order. A check digit is generated for the UPC Version A symbol. A second check digit is generated from the five supplement digits. It is used to assign the left odd and even parity of the five-digit supplemental bar code symbol. The supplement check digit is not encoded or interpreted.

Specify 16 digits of input data.

X'02' The five-digit UPC supplemental bar code symbol is preceded by a UPC Version E symbol. The bar code object contains both the UPC Version E symbol and the five-digit supplemental symbol. The input data consists of the 10-digit article number and the five-digit supplemental data. The bar code object processor generates the six-digit UPC Version E symbol and check digit. The check digit is used to determine the parity pattern of the Version E symbol. The five-digit supplemental bar code symbol is presented after the Version E symbol. A second check digit is calculated for the five-digit supplemental data and is used to assign the left odd and even parity. The supplement check digit is not encoded or interpreted.

Specify 15 digits of input data.

EAN-8 (includes JAN-short)

X'00' Present an EAN-8 bar code symbol. The input data consists of 7 digits: two flag digits and five article number digits. All 7 digits are encoded along with a generated check digit.

EAN-13 (includes JAN-standard)
X'00'  Present an EAN-13 bar code symbol. The input data consists of 12 digits: two flag digits and 10 article number digits, in that order. The first flag digit is not encoded. The second flag digit, the article number digits, and generated check digit are encoded. The first flag digit is presented in HRI form at the bottom of the left quiet zone. The first flag digit governs the A and B number-set pattern of the bar and space coding of the 6 digits to the left of the symbol center pattern.

**Industrial 2-of-5**

![Industrial 2-of-5](image1)

X'01'  Present the bar code without a generated check digit.
X'02'  Generate a check digit and present it with the bar code.

**Matrix 2-of-5**

![Matrix 2-of-5](image2)

X'01'  Present the bar code symbol without a generated check digit.
X'02'  Generate a check digit and present it with the bar code.

**Interleaved 2-of-5, AIM USS-I 2/5**

![Interleaved 2-of-5](image3)

The Interleaved 2-of-5 symbology requires an even number of digits. The printer adds a leading zero if necessary to meet this requirement.

X'01'  Present the bar code symbol without a check digit.
X'02'  Generate a check digit and present it with the bar code.
**Codabar, 2-of-7, AIM USS-Codabar**

X'01' Present the bar code without a generated check digit. The input data consists of a start character, digits to be encoded, and a stop character, in that order. Start and stop characters can be A, B, C, or D, and can be used only at the beginning and end of the symbol.

X'02' Generate a check digit and present it with the bar code. The input data consists of a start character, digits to be encoded, and a stop character, in that order. Start and stop characters can be A, B, C, or D, and can be used only at the beginning and end of the symbol.

**Code 128, AIM USS-128 (modifier values X'02' through X'04')**

The 1986 symbology definition for Code 128 defined an algorithm for generating a start character and then changed that algorithm in 1993 to accommodate the UCC/EAN 128 variation of this bar code. Many BCODA printers implemented the 1986 version (with modifier X'02'), some BCODA printers changed to use the 1993 algorithm (with modifier X'02'), and some BCODA printers support both algorithms. When UCC/EAN 128 bar codes are produced for printers that explicitly support UCC/EAN 128, modifier X'03' or modifier X'04' must be specified. For printers that do not explicitly support UCC/EAN 128, specifying modifier X'02' might produce a valid UCC/EAN 128 bar code (see notes in the modifier descriptions).

The data for UCC/EAN 128 bar codes is in the form: "FNC1, ai, data, m, FNC1, ai, data, m, FNC1, ..., ai, data, m"

FNC1 is the FNC1 function character (X'8F'), ai is an application identifier, data is defined for each registered application identifier, and m is a modulo 10 check digit (calculated with the same check digit algorithm that is used for UPC version A bar codes). Not all application identifiers require a modulo 10 check digit (m). Also, all except the firstFNC1 are field separator characters that appear only when the preceding ai data is of variable length. See **UCC/EAN-128 APPLICATION IDENTIFIER STANDARD** from the Uniform Code Council, Inc. for a description of application identifiers and the use of FNC1. To build the bar code symbol, the printer must:

- Produce a start character based on the 1993 algorithm.
- Bar encode the data, including the FNC1, ai, data, and m check digit.
- Produce a modulo 103 check digit.
- Produce a stop character.
Modifier X’02’ – Code 128 symbol, with original (1986) start-character algorithm

Generate a Code 128 symbol with subset A, B, or C to produce the shortest possible bar code from the specified data, by using the start-character algorithm that was published in the original (1986) edition of the Code 128 Symbology Specification. The Code 128 code page (CPGID = 1303, GCSGID = 1454) is used to interpret the bar code symbol data. Generate a check digit and present it with the bar code.

Notes:
1. Some IPDS printers use the modifier X’03’ start-character algorithm even when modifier X’02’ is specified. This produces a valid UCC/EAN 128 symbol when valid UCC/EAN 128 data is provided. However, in general, modifier X’02’ must not be used to produce UCC/EAN 128 symbols because this value causes other IPDS printers to use the original Code 128 start-symbol algorithm, which generates a Start (Code B) instead of the Start (Code C) that UCC/EAN 128 requires. Some bar code scanners can handle either start character for a UCC/EAN 128 symbol, but others require the Start (Code C) character.
2. These printers use the UCC/EAN 128 start-character algorithm when modifier X’02’ is specified: 4312, 4317, 4324, InfoPrint 20, InfoPrint 21, InfoPrint 32, InfoPrint 40, InfoPrint 45, InfoPrint 70, InfoPrint 2070, InfoPrint 2085, and InfoPrint 2105. Other IPDS printers use the original start-character algorithm when modifier X’02’ is specified.

Modifier X’03’ – UCC/EAN 128 symbol, without parenthesis in the HRI

Generate a Code 128 symbol with subset A, B, or C to produce the shortest possible bar code from the specified data, by using the version of the start-character algorithm that was modified for producing UCC/EAN 128 symbols. If the first data character is FNC1 (as is required for a UCC/EAN 128 symbol) and is followed by valid UCC/EAN 128 data, the printer generates a Start (Code C) character. The Code 128 code page (CPGID =
1303, GCSGID = 1454) is used to interpret the bar code symbol data. Generate a check digit and present it with the bar code.

The UCC/EAN 128 data is checked for validity and exception condition EC-1200 exists if one or more of the following conditions are encountered:

- FNC1 is not the first data character.
- Invalid application identifier (ai) value encountered.
- Data for an ai doesn't match the ai definition.
- Insufficient (or no) data that follows an ai.
- Too much data for an ai.
- Invalid use of FNC1 character.

Notes:

1. UCC/EAN 128 is a variation of Code 128 that begins with an FNC1 character, followed by an Application Identifier and the data to be bar encoded. All of these characters (including the FNC1 character) must be supplied within the Bar Code Symbol Data (BSA). UCC/EAN 128 also requires that the symbol begin in subset C.

2. For UCC/EAN 128 symbols, the start character, the FNC1 characters, the modulo 103 check digit, and the stop character are not shown in the human readable format.

Modifier X'04' – UCC/EAN 128 symbol, with parenthesis in the HRI

(01)90614141007687(15)001230

SCC-14 and Sell-By Date Concatenated in a UCC/EAN-128 Symbol

Generate a Code 128 symbol in the same manner as for modifier X'03', but use parenthesis in the HRI to distinguish each application identifier (ai). The printer inserts the parenthesis in the printed HRI when modifier X'04' is specified; these parentheses are not part of the input data.

EAN Two-Digit Supplemental

041234567890 (encoding 041234567890, supplemental = 99)

X'00' Present the EAN two-digit supplemental bar code symbol. This option assumes that the base EAN-13 symbol is presented as a separate bar code object. The value of the two digit supplemental data determines their bar and space patterns that are chosen from number sets A and B.

Specify 2 digits of input data.
The two-digit supplemental bar code symbol is preceded by a normal EAN-13 bar code symbol. The bar code object contains both the EAN-13 symbol and the two-digit supplemental symbol. The two-digit supplemental bar code is presented after the EAN-13 symbol by using left odd and even parity as determined by the two supplemental digits that are chosen from number sets A and B. Specify 14 digits of input data.

**Note:** Used for both books and paperbacks.

### EAN Five-Digit Supplemental

![EAN Five-Digit Supplemental](image)

**EAN + 5 Digit Supplemental**

(encoding 041234567890, supplemental = 54321)

X'00'

Present the EAN five-digit supplemental bar code. This option assumes that the base EAN-13 symbol is presented as a separate bar code object. A check digit is calculated from the five supplemental digits. The check digit is also used to assign the bar and space patterns from number sets A and B for the five supplemental digits. The check digit is not encoded or interpreted. Specify 5 digits of input data.

X'01'

The five-digit supplemental bar code symbol is preceded by a normal EAN-13 bar code symbol. The bar code object contains both the EAN-13 symbol and the five-digit supplemental symbol. A check digit is generated from the five-digit supplemental data. The check digit is used to assign the bar and space patterns from number sets A and B. The check digit is not encoded or interpreted. Specify 17 digits of input data.

**Note:** Used for books and paperbacks.

### POSTNET and PLANET

**US POSTNET**

<table>
<thead>
<tr>
<th>Zip+4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(encoding 12345+6789)</td>
</tr>
</tbody>
</table>

**PLANET Code**

| (encoding 00123456789) |

For all POSTNET modifiers that follow, the BSA HRI flag field and the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields are not applicable to the POSTNET bar code symbology. These fields are ignored because the POSTNET symbology defines specific values for these parameters.

X'00'

Present a POSTNET ZIP Code bar code symbol. The ZIP Code to be encoded is defined as a five-digit, numeric (0 - 9), data variable to the BSA data structure. The POSTNET ZIP Code bar code

Appendix D. More About Bar Code Parameters  495
consists of a leading frame bar, the encoded ZIP Code data, a correction digit, and a trailing frame bar.

X'01'
Present a POSTNET ZIP+4 bar code symbol. The ZIP+4 code to be encoded is defined as a nine-digit, numeric (0 - 9), data variable to the BSA data structure. The POSTNET ZIP+4 bar code consists of a leading frame bar, the encoded ZIP+4 data, a correction digit, and a trailing frame bar.

X'02'
Present a POSTNET Advanced Bar Code (ABC) bar code symbol. The ABC code to be encoded is defined as an 11-digit, numeric (0 - 9), data variable to the BSA data structure. The POSTNET ABC bar code consists of a leading frame bar, the encoded ABC data, a correction digit, and a trailing frame bar.

Note: An 11-digit POSTNET bar code is called a Delivery Point bar code.

X'03'
Present a POSTNET variable-length bar code symbol. The data to be encoded is defined as an n-digit, numeric (0 - 9), data variable to the BSA data structure. The bar code symbol is generated without length checking; the symbol is not guaranteed to be scannable or interpretable. The POSTNET variable-length bar code consists of a leading frame bar, the encoded data, a correction digit, and a trailing frame bar.

X'04'
Present a PLANET Code symbol. The PLANET Code is a reverse topology variation of POSTNET that encodes 11 digits of data; the first 2 digits represent a service code (such as 21 = Origin Confirm and 22 = Destination Confirm) and the next 9 digits identify the mailpiece. A 12th digit is generated by the printer as a check digit. The PLANET Code symbol consists of a leading frame bar, the encoded data, a check digit, and a trailing frame bar.

Royal Mail (RM4SCC and Dutch KIX)

```
Royal Mail (RM4SCC)
UK and Singapore version
(encoding SN34RD1A)

Royal Mail (RM4SCC)
Dutch KIX version
(encoding SN34RD1A)
```

A 4-state customer code that is defined by the Royal Mail Postal service of England for use in bar coding postal code information. This symbology is also called the Royal Mail bar code or the 4-State customer code. The symbology (as defined for modifier X'00') is used in the United Kingdom and in Singapore. A variation that is called KIX (KlantenIndeX = customer index, as defined for modifier X'01') is used in the Netherlands.

X'00'
Present a RM4SCC bar code symbol with a generated start bit, checksum character, and stop bit. The start and stop bits identify the beginning and end of the bar code symbol and also the orientation of the symbol.

X'01'
Present a RM4SCC bar code symbol with no start bar, no checksum character, and no stop bar.
**Note:** Modifier X’01’ is also known as *Dutch Kix Postal Bar Code*. In addition to the characters allowed in Modifier X’00’, it allows lowercase alphabetical characters, which are folded to uppercase by the printer.

**Japan Postal Bar Code (JPOSTAL)**

```
  Japan Postal Bar Code
  Modifier X'00'
  (encoding 15400233-16-4)
```

A complete Japan Postal Bar Code symbol that consists of a set of distinct bars and spaces for each character, followed by a modulo 19 checksum character and enclosed by a unique start character, stop character, and quiet zones.

X’00’ Present a Japan Postal Bar Code symbol with a generated start character, checksum character, and stop character.

The generated bar code symbol consists of a start code, a 7-digit new postal code, a 13-digit address indication number, a check digit, and a stop code. The variable data to be encoded (BSA bytes 5-n) is used as follows:

1. The first few digits represent the new postal code in either the form nnn-nnnn or the form nnnnnnn; the hyphen, if present, is ignored and the other 7 digits must be numeric. The 7 digits are placed in the new postal code field of the bar code symbol.
2. If the next digit is a hyphen, it is ignored and is not used in generating the bar code symbol.
3. The remainder of the BSA data is the address indication number, which can contain numbers, hyphens, and alphabetic characters (A-Z). Each number and each hyphen represents one digit in the bar code symbol; each alphabetic character is represented by a combination of a control code (CC1, CC2, or CC3) and a numerical code, and handled as two digits in the bar code symbol. Thirteen digits of this address indication number data are placed in the address indication number field of the bar code symbol.
   - If fewer than 13 more digits are present, the shortage is filled in with the bar code corresponding to control code CC4 up to the thirteenth digit.
   - If more than 13 more digits are present, the first 13 are used and the remainder are ignored, with no exception condition reported. However, if the thirteenth digit is the control code for an alphabetic (A-Z) character, only the control code is included and the numeric part is omitted.

X’01’ Present a Japan Postal Bar Code symbol directly from the bar code data. Each valid character in the BSA data field is converted into a bar/space pattern, with no validity or length checking. The printer does not generate start, stop, or check digits.

To produce a valid bar code symbol, the bar code data must contain a start code, a 7-digit new postal code, a 13-digit address indication number, a valid check digit, and a stop code. The new postal code must consist of 7 numeric digits. The address indication number must consist of 13 characters, which can be
numeric, hyphen, or control characters (CC1 through CC8). The following table lists the valid code points for modifier X'01':

<table>
<thead>
<tr>
<th>Bar Code Character</th>
<th>Code Point</th>
<th>Numerical Checking Value</th>
<th>Bar Code Character</th>
<th>Code Point</th>
<th>Numerical Checking Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>X'4C'</td>
<td>0</td>
<td>stop</td>
<td>X'6E'</td>
<td>1</td>
</tr>
<tr>
<td>hyphen</td>
<td>X'60'</td>
<td>10</td>
<td></td>
<td>X'F0'</td>
<td>0</td>
</tr>
<tr>
<td>CC1</td>
<td>X'5A'</td>
<td>11</td>
<td>CC2</td>
<td>X'7F'</td>
<td>12</td>
</tr>
<tr>
<td>CC3</td>
<td>X'7B'</td>
<td>13</td>
<td>CC4</td>
<td>X'E0'</td>
<td>14</td>
</tr>
<tr>
<td>CC5</td>
<td>X'6C'</td>
<td>15</td>
<td>CC6</td>
<td>X'50'</td>
<td>16</td>
</tr>
<tr>
<td>CC7</td>
<td>X'7D'</td>
<td>17</td>
<td></td>
<td>X'F9'</td>
<td>18</td>
</tr>
<tr>
<td>CC8</td>
<td>X'4D'</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:

1. Do not attempt to use the Start and Stop characters in calculating the check digit. You can use the remaining characters to generate check digits; they are the only characters that are valid for check digits. Use the Numeric Checking Values to calculate the check digits.

2. You supply data generation for mod 1. The check digit is the sum of the digits module 19, which is a remainder of X. The check digit is 19 minus X, converted to hex. If this is done incorrectly, the print server displays message APS830I.

The hyphen has a hex value of X'60' and a checking digit numerical of 10.

The following example is a generation of the customer bar code where 154-0023 is the new postal code and 3 - 16 - 4 is the address indication number:

```
address
154
3-16-4, Wakabayshi, Setagaya-ku
```

New postal code + address indication number:

```
154-0023-3-16-4
```

Delete hyphens between the third and fourth digits of the new postal code and between the new postal code and address indication number, as follows:

```
15400233-16-4
```

If the address indication number is shorter than 13 digits, use CC4s to fill the remaining spaces, as in the following example.

```
15400233-16-4 CC4 CC4 CC4 CC4 CC4 CC4 CC4
```

The first 7 digits are ignored as the postal code and the remaining digits are the address indication number. Remember
to count hyphens as digits. In the previous example, the postal code is 1540023 and the address indication number is 3-16-4 plus seven CC4 characters.

Calculate the check digit (CD), based on the table of correspondence between characters for bar code and checking numerals. See Table 25 on page 498 for more information about check digits.

\[
1+5+4+0+0+2+3+10+1+6+10+4+14+14+14+14+14+14+14+14+CD = 147 + CD = \text{integral multiple of 19}
\]

Using the integral multiple of 19, 152 - 147 = 5 for the check digit, based on the table of correspondence between characters for bar code and checking numerals. Five corresponds to checking numerical five.

For the previous postal code and address indication number, calculate the hex value of the check digit. The following table shows how to convert the data to hex values. Add the check digit (CD), start code (STC), and stop code (SPC), as follows:

<table>
<thead>
<tr>
<th>Start Code (STC)</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F1</td>
</tr>
<tr>
<td>5</td>
<td>F5</td>
</tr>
<tr>
<td>4</td>
<td>F4</td>
</tr>
<tr>
<td>0</td>
<td>F0</td>
</tr>
<tr>
<td>0</td>
<td>F0</td>
</tr>
<tr>
<td>2</td>
<td>F2</td>
</tr>
<tr>
<td>3</td>
<td>F3</td>
</tr>
<tr>
<td>3</td>
<td>F3</td>
</tr>
<tr>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>1</td>
<td>F1</td>
</tr>
<tr>
<td>6</td>
<td>F6</td>
</tr>
<tr>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>F4</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CC4</td>
<td>E0</td>
</tr>
<tr>
<td>CD(5)</td>
<td>F5</td>
</tr>
<tr>
<td>SPC</td>
<td>6E</td>
</tr>
</tbody>
</table>

Notice that the check digit (CD) equals 5 and is converted to the hex value of F5.

The following are examples of various Japanese postal bar codes.
This bar code used numeric postal codes only. The 7-digit field contains the start, stop, and checksum characters. The printer does not generate start, stop, or checksum characters.

This bar code used alphanumeric postal codes only. The 13-digit field contains start, stop, checksum, and command codes. The printer does not generate start, stop, or checksum characters.

This bar code used numeric postal codes only. This is a 7-digit character field.

This bar code used alphanumeric postal codes only. This is a 13-digit character field.

**Data Matrix (2DMATRIX)**

![Data Matrix 2D Symbol](encodingA1B2C3D4E5F6G7H8I9J0K1L2)

A two-dimensional matrix bar code symbology that is defined as an AIM International Symbology Specification.

**X'00'** Present a Data Matrix Bar Code symbol by using Error Checking and Correcting (ECC) algorithm 200.

The bar code data is assumed to start with the default character encodation (ECI 000003 = ISO 8859-1). This is an international Latin 1 code page that is equivalent to the ASCII code page 819. To change to a different character encodation within the data, the ECI
protocol as defined in the *AIM International Symbology Specification - Data Matrix*, must be used. This means that whenever a byte value of X'5C' (an escape code) is encountered in the bar code data, the next six characters must be decimal digits (byte values X'30' - X'39') or the next character must be another X'5C'. When the X'5C' character is followed by six decimal digits, the six decimal digits are interpreted as the ECI number, which changes the interpretation of the characters that follow the decimal digits. When the X'5C' character is followed by another X'5C' character, this is interpreted as one X'5C' character (which is a backslash in the default character encodation); alternatively, the escape-sequence handling flag can be used to treat X'5C' as a normal character.

Since the default character encodation for this bar code is ASCII, the EBCDIC-to-ASCII translation flag can be used when all of the data for the bar code is EBCDIC. If the bar code data contains more than one character encodation or if the data needs to be encoded within the bar code symbol in a form other than the default character encodation (such as, in EBCDIC), the bar code data must begin in the default encodation, the EBCDIC-to-ASCII translation flag must be set to B'0', and the ECI protocol must be used to switch into the other encodation.

**Note:** For more information about 2DMATRIX two-dimensional matrix bar codes, see “Data Matrix Special-Function Parameters” on page 517.

**MaxiCode (2DMAXI)**

A two-dimensional matrix bar code symbology that is defined as an AIM International Symbology Specification.

X'00'  Present a MaxiCode bar code symbol.

The bar code data is assumed to start with the default character encodation (ECI 000003 = ISO 8859-1). This is an international Latin 1 code page that is equivalent to the ASCII code page 819. To change to a different character encodation within the data, the ECI protocol as defined in section 4.15.2 of the *AIM International Symbology Specification - MaxiCode*, must be used. This means that whenever a byte value of X'5C' (an escape code) is encountered in the bar code data, the next six characters must be decimal digits (byte values X'30' - X'39') or the next character must be another X'5C'. When the X'5C' character is followed by six decimal digits, the six decimal digits are interpreted as the ECI number, which changes the interpretation of the characters that follow the decimal digits. When the X'5C' character is followed by another X'5C' character, this is interpreted as one X'5C' character (which is a backslash in the default character encodation); alternatively, the escape-sequence handling flag can be used to treat X'5C'
as a normal character. The X'5C' character is allowed anywhere in the bar code data except for Modes 2 and 3 where it is not allowed in the Primary Message portion of the data.

Since the default character encodation for this bar code is ASCII, the EBCDIC-to-ASCII translation flag can be used when all of the data for the bar code is EBCDIC. If the bar code data contains more than one-character encodation or if the data needs to be encoded within the bar code symbol in a form other than the default character encodation (such as, in EBCDIC), the bar code data must begin in the default encodation, the EBCDIC-to-ASCII translation flag must be set to B'0', and the ECI protocol must be used to switch into the other encodation.

**Note:** For more information about 2DMAXI two-dimensional matrix bar codes.

**2DPDF417**

A two-dimensional matrix bar code symbology that is defined as an AIM International Symbology Specification — PDF417.

X'00' Present a full PDF417 bar code symbol.

X'01' Present a truncated PDF417 bar code symbol for use in an environment in which damage to the symbol is unlikely. This version omits the right row indicator and simplifies the stop pattern into a single module width bar.

The bar code data is assumed to start with the default character encodation (GLI 0) as defined in Table 5 of the Uniform Symbology Specification PDF417. To change to another character encodation, the GLI (Global Label Identifier) protocol, as defined in the Uniform Symbology Specification PDF417, must be used. This means that whenever a byte value of X'5C' (an escape code) is encountered in the bar code data, the next three characters must be decimal digits (byte values X'30' - X'39') or the next character must be another X'5C' character. When the X'5C' character is followed by three decimal digits, this is called an escape sequence. When the X'5C' character is followed by another X'5C' character, this is interpreted as one X'5C' character (which is a backslash in the default character encodation); alternatively, the escape-sequence handling flag can be used to treat X'5C' as a normal character.

To identify a new GLI, two or three escape sequences in a row must exist. The first escape sequence must be "\925", "\926", or "\927" (as defined by GLI 0). If the first escape sequence is "\925" or "\927", one other escape sequence must follow that contains a value from "\000" to "\899". If the first escape sequence is "\926", two more escape sequences must follow and each escape sequence must contain a value from "\000" to "\899".
example, to switch to GLI 1 (ISO 8859-1, which is equivalent to ASCII code page 819), the bar code data would contain the character sequence “\927\001”. The “\927” escape sequence is used for GLI values 0 - 899. The “\926” escape sequence is used for GLI values 900 - 810,889. The “\925” escape sequence is used for GLI values 810,900 - 811,799. For more information about how these values are calculated, see section 2.2.6 of the Uniform Symbology Specification PDF417.

In addition to transmitting GLI numbers, the escape sequence is used to transmit other codewords for more purposes. The special codewords are given in Table 8 in Section 2.7 of the Uniform Symbology Specification PDF417. The special codewords “\903” to “\912” and “\914” to “\920” are reserved for future use. The BCoca reader accepts these special escape sequences and adds them to the bar code symbol, resuming normal encoding with the character that follows the escape sequence.

The special codeword “\921” instructs the bar code reader to interpret the data that is contained within the symbol for reader initialization or programming. This escape sequence is only allowed at the beginning of the bar code data.

The special codewords “\922”, “\923”, and “\928” are used for coding a Macro PDF417 Control Block as defined in section G.2 of the Uniform Symbology Specification PDF417. These codewords must not be used within the BCoca data; instead, a Macro PDF417 Control Block can be specified in the special-function parameters. Exception condition EC-2100 exists if one of these escape sequences is found in the bar code data.

Because the default character encodation for this bar code is GLI 0 (an ASCII code page that is similar to IBM code page 437), the EBCDIC-to-ASCII translation flag can be used when all of the data for the bar code is EBCDIC. If the bar code data contains more than one-character encodation, or if the data needs to be encoded within the bar code symbol in a form other than the default character encodation (such as in EBCDIC), the bar code data must begin in the default encodation, the EBCDIC-to-ASCII translation flag must be set to B’0’, and the GLI protocol must be used to switch into the other encodation.

Note: For more information about 2DPDF417 two-dimensional matrix bar codes, see “PDF417 Special-Function Parameters” on page 526.

Australia Post Bar Code (APOSTAL)

A bar code symbology that is defined by Australia Post for use in Australian postal systems. Several formats of this bar code exist, which are identified by the modifier byte as follows:

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Type of bar code</th>
<th>Valid bar code data</th>
</tr>
</thead>
<tbody>
<tr>
<td>X’01’</td>
<td>Standard Customer Barcode (format code = 11)</td>
<td>An 8-digit number that represents the Sorting Code.</td>
</tr>
<tr>
<td>Modifier</td>
<td>Type of bar code</td>
<td>Valid bar code data</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>X'02'</td>
<td>Customer Barcode 2 using Table N (format code = 59)</td>
<td>An 8-digit number that represents the Sorting Code followed by up to 8 numeric digits that represent the Customer Information.</td>
</tr>
<tr>
<td>X'03'</td>
<td>Customer Barcode 2 using Table C (format code = 59)</td>
<td>An 8-digit number that represents the Sorting Code followed by up to 5 characters (A - Z, a - z, 0 - 9, space, #) that represent the Customer Information.</td>
</tr>
<tr>
<td>X'04'</td>
<td>Customer Barcode 2 using proprietary encoding (format code = 59)</td>
<td>An 8-digit number that represents the Sorting Code followed by up to 16 numeric digits (0 - 3) that represent the Customer Information. Each of the 16 digits specifies one of the 4 types of bar.</td>
</tr>
<tr>
<td>X'05'</td>
<td>Customer Barcode 3 using Table N (format code = 62)</td>
<td>An 8-digit number that represents the Sorting Code followed by up to 15 numeric digits that represent the Customer Information.</td>
</tr>
<tr>
<td>X'06'</td>
<td>Customer Barcode 3 using Table C (format code = 62)</td>
<td>An 8-digit number that represents the Sorting Code followed by up to 10 characters (A - Z, a - z, 0 - 9, space, #) that represent the Customer Information.</td>
</tr>
<tr>
<td>X'07'</td>
<td>Customer Barcode 3 using proprietary encoding (format code = 62)</td>
<td>An 8-digit number that represents the Sorting Code followed by up to 31 numeric digits (0 - 3) that represent the Customer Information. Each of the 31 digits specify one of the 4 types of bar.</td>
</tr>
<tr>
<td>X'08'</td>
<td>Reply Paid Barcode (format code = 45)</td>
<td>An 8-digit number that represents the Sorting Code.</td>
</tr>
</tbody>
</table>

The proprietary encoding allows the customer to specify the types of bars to be printed directly by using 0 for a full bar, 1 for an ascending bar, 2 for a descending bar and 3 for a timing bar. If the customer does not specify enough Customer Information to fill the field, the printer uses a filler bar to extend pad the field out to the correct number of bars.

The printer encodes the data by using the proper tables, generate the start and stop bars, generate any needed filler bars, and generate the Reed Solomon ECC bars.

Human-readable interpretation (HRI) can be selected with this bar code type. The format control code, Delivery Point Identifier, and customer information field (if any) appear in the HRI, but the ECC does not.

The proprietary encoding allows the customer to specify the types of bars to be printed directly by using 0 for a full bar, 1 for an ascending bar, 2 for a descending bar, and 3 for a timing bar. If the customer does not specify enough Customer Information to fill the field, the printer uses a filler bar to extend pad the field out to the correct number of bars.

The printer encodes the data by using the proper tables, generate the start and stop bars, generate any needed filler bars, and generate the Reed Solomon ECC bars.

Human readable interpretation (HRI) can be selected with this bar code type. The format control code, Delivery Point Identifier, and customer information field (if any) appear in the HRI, but the ECC does not.
QR Code

QR Code 2D Symbol
A two-dimensional matrix bar code symbology that is defined as an AIM International Technical Standard.

X’02’ Present a Model 2 QR Code Bar Code symbol as defined in AIM International Symbology Specification — QR Code.

The bar code data is assumed to start with the default character encodation (ECI 000020). This is a single-byte code page that represents the JIS8 and Shift JIS character sets; it is equivalent to the ASCII code page 897. To change to a different character encodation within the data, the ECI protocol as defined in the AIM International “Extended Channel Interpretation (ECI) Assignments”, must be used.

Since the default character encodation for this bar code is ASCII, the EBCDIC-to-ASCII translation flag can be used in the following manner:

• When all of the input data for the bar code is single-byte EBCDIC that uses one of the supported code pages (500, 290, or 1027), set the EBCDIC-to-ASCII translation flag to B’1’ and select the correct code page in the conversion parameter.

• When all of the input data for the bar code is mixed-byte EBCDIC AFP Line Data that uses SO and SI controls (SOSI data), set the EBCDIC-to-ASCII translation flag to B’1’ and select the conversion value you want in the conversion parameter.

If the bar code data contains more than one-character encodation or if the data needs to be encoded within the bar code symbol in a form other than the ones that are previously mentioned (such as in an EBCDIC code page that is not supported by the EBCDIC-to-ASCII translation flag), the bar code data must begin in the default encodation, the EBCDIC-to-ASCII translation flag must be set to B’0’, and the ECI protocol must be used to switch to the other encodings.

A quiet zone must exist around the symbol that is at least four modules wide on each of the four sides of the symbol.

Note: For more information about QRCODE two-dimensional bar code see “QR Code Special-Function Parameters” on page 531.

Code 93

Code 93
(encoding 39OR93
yielding a 1.82 inch wide symbol)

A linear bar code symbology similar to Code 39, but more compact than Code 39. Code 93 bar code symbols are made up of a series of characters
each of which is represented by 9 modules that are arranged into 3 bars with their adjacent spaces. The bars and spaces vary between 1 module wide and 4 modules wide.

**X’00’** Present a Code 93 bar code symbol as defined in *AIM International Symbology Specification — Code 93*.

The Code 93 character set contains 47 characters, including numeric digits, uppercase alphabets, four shift characters (a, b, c, and d), and seven special characters. The Code 93 Specification also provides a method of encoding all 128 ASCII characters by using two bar code characters for those ASCII characters that are not in the standard Code 93 character set. This is sometimes referred to as *Extended Code 93*. In this case, the two bar code characters that are used to specify the extended character are shown in the Human-Readable Interpretation (as a • followed by the second character) and the bar code scanner interprets the two-character combination bar/space pattern.

The Human-Readable Interpretation of the Start and Stop characters is represented as an open box (•) and the shift characters (a, b, c, and d) are represented as a filled box (•).

A quiet zone must precede and follow the symbol that is at least 10 modules wide.

**USPS Four-State**

![USPS Four-State Bar Code]

**USPS Four-State Bar Code**

*Modifier X’03’*

*(encoding 01 234 567094 987654321 01234567891)*

The USPS Four-State bar code symbology limits the symbol size; therefore, BSD element height, height multiplier, and wide-to-narrow ratio fields are not applicable to this symbology and are ignored by BCOCA receivers. The module width field allows for two symbol sizes (small and optimal); the small symbol is approximately 2.575 inches wide and the optimal symbol is approximately 2.9 inches wide.

The input data is all numeric and consists of 5 data fields. The first four fields are fixed length and the 5th field can have one of four lengths; the bar code modifier is used to specify the length of the 5th field. The total length of the input data can be 20, 25, 29, or 31 digits and is defined as follows:

- **Bar code ID (2 digits)** - assigned by USPS; the 2nd digit must be 0 - 4. Thus, the valid values are: 00 - 04, 10 - 14, 20 - 24, 30 - 34, 40 - 44, 50 - 54, 60 - 64, 70 - 74, 80 - 84, and 90 - 94.
- **Service Code (3 digits)** - assigned by USPS; valid values are 000 - 999.
- **Subscriber ID (6 digits)** - assigned by USPS; valid values are 000000 - 999999.
- **Unique ID (9 digits)** - assigned by the mailer; valid values are 000000000 - 999999999.

---

8. The United States Postal Service (USPS) developed this symbology for use in the USPS mailstream and named it the OneCode Barcode. The bar code is also known as the *4-state Customer Barcode* and is abbreviated in several ways: OneCode (4CB), OneCode (4-CB), 4CB, or 4-CB.
• Routing ZIP Code (0, 5, 9, or 11 digits) - refer to the modifier for valid values.

USPS Four-State modifier values are defined as follows:

X'00' Present a USPS Four-State bar code symbol with no Delivery Point ZIP Code. The input data for this bar code symbol must be 20 number digits.

X'01' Present a USPS Four-State bar code symbol with a 5-digit Delivery Point ZIP Code. The input data for this bar code symbol must be 25 number digits; the valid values for the Delivery Point ZIP Code are 00000 - 99999.

X'02' Present a USPS Four-State bar code symbol with a 9-digit Delivery Point ZIP Code. The input data for this bar code symbol must be 29 number digits; the valid values for the Delivery Point ZIP Code are 000000000 - 999999999.

X'03' Present a USPS Four-State bar code symbol with an 11-digit Delivery Point ZIP Code. The input data for this bar code symbol must be 31 number digits; the valid values for the Delivery Point ZIP Code are 00000000000 - 99999999999.

Human-Readable Interpretation (HRI) can be printed with a USPS Four-State symbol, but HRI is not used with all types of special services. See Introducing 4-state Customer Barcode for a description of when HRI is appropriate.

A quiet zone must surround the symbol (all four sides) that is at least 0.04 inches above and below and at least 0.125 inches on both sides of the symbol.

Check Digit Calculation Method

Some bar code types and modifiers call for the calculation and presentation of check digits. Check digits are a method of verifying data integrity during the bar coding reading process. Except for UPC Version E, the check digit is always presented in the bar code bar and space patterns, but is not always presented in the HRI. The following table shows the check digit calculation methods for each bar code type and the presence or absence of the check digit in the HRI.

Table 27. Check Digit Calculation Methods For Each Bar Code

<table>
<thead>
<tr>
<th>Bar Code Type</th>
<th>Modifier</th>
<th>In HRI?</th>
<th>Check Digit Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Code 39 (3-of-9 Code), AIM USS-39</td>
<td>X'02</td>
<td>Yes</td>
<td>Modulo 43 of the sum of the data characters' numerical values as described in a Code 39 specification. The start and stop codes are not included in the calculation.</td>
</tr>
</tbody>
</table>
Table 27. Check Digit Calculation Methods For Each Bar Code  (continued)

<table>
<thead>
<tr>
<th>Bar Code Type</th>
<th>Modifier</th>
<th>In HRI?</th>
<th>Check Digit Calculation</th>
</tr>
</thead>
</table>
| 2 – MSI (modified Plessey code) | X'02' - X'09' | No | IBM Modulus 10 check digit:  
1. Multiply each digit of the original number by a weighting factor of 1 or 2 as follows: multiply the units digit by 2, the tens digit by 1, the hundreds digit by 2, the thousands digit by 1, and so forth.  
2. Sum the digits of the products from step 1. This is not the same as summing the values of the products.  
3. The check digit is described by the following equation where sum is the resulting value of step 2:  
   \[(10 - (\text{sum} \mod 10)) \mod 10\]  
   The check digit as 11 minus the remainder is described by the following equation:  
   \[(11 - (\text{sum} \mod 11)) \mod 11\]  
| | | | IBM Modulus 11 check digit:  
1. Multiply each digit of the original number by a repeating weighting factor pattern of 2, 3, 4, 5, 6, 7 as follows: multiply the units digit by 2, the tens digit by 3, the hundreds digit by 4, the thousands digit by 5, and so forth.  
2. Sum the products from step 1.  
3. The check digit depends on the bar code modifier. The check digit as the remainder is described by the following equation where sum is the resulting value of step 2:  
   \[(\text{sum} \mod 11)\]  
   The check digit as 11 minus the remainder is described by the following equation:  
   \[(11 - (\text{sum} \mod 11)) \mod 11\] |
### Table 27. Check Digit Calculation Methods For Each Bar Code (continued)

<table>
<thead>
<tr>
<th>Bar Code Type</th>
<th>Modifier</th>
<th>In HRI?</th>
<th>Check Digit Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCR Modulus 11 check digit:</td>
<td></td>
<td></td>
<td>1. Multiply each digit of the original number by a repeating weight factor pattern of 2, 3, 4, 5, 6, 7, 8, 9 as follows: multiply the units digit by 2, the tens digit by 3, the hundreds digit by 4, the thousands digit by 5, and so forth.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Sum the products from step 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. The check digit depends on the bar code modifier. The check digit as the remainder is described by the following equation where $sum$ is the resulting value of step 2: $(sum \mod 11)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The check digit as 11 minus the remainder is described by the following equation: $(11 - (sum \mod 11)) \mod 11$</td>
</tr>
<tr>
<td>3 – UPC/CGPC Version A</td>
<td>X'00'</td>
<td>Yes</td>
<td>UPC/EAN check digit calculation: 1. Multiply each digit of the original number by a weighting factor of 1 or 3 as follows: multiply the units digit by 3, the tens digit by 1, the hundreds digit by 3, the thousands digit by 1, and so forth.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Sum the products from step 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. The check digit is described by the following equation where $sum$ is the resulting value of step 2: $(10 - (sum \mod 10)) \mod 10$</td>
</tr>
<tr>
<td>5 – UPC/CGPC Version E</td>
<td>X'00'</td>
<td>Yes</td>
<td>See UPC/CGPC Version A</td>
</tr>
<tr>
<td>8 – EAN 8 (includes JAN-short)</td>
<td>X'00'</td>
<td>Yes</td>
<td>See UPC/CGPC Version A</td>
</tr>
<tr>
<td>9 – EAN 13 (includes JAN-standard)</td>
<td>X'00'</td>
<td>Yes</td>
<td>See UPC/CGPC Version A</td>
</tr>
<tr>
<td>10 – Industrial 2-of-5</td>
<td>X'02'</td>
<td>Yes</td>
<td>See UPC/CGPC Version A</td>
</tr>
<tr>
<td>11 – Matrix 2-of-5</td>
<td>X'02'</td>
<td>Yes</td>
<td>See UPC/CGPC Version A</td>
</tr>
<tr>
<td>12 – Interleaved 2-of-5</td>
<td>X'02'</td>
<td>Yes</td>
<td>See UPC/CGPC Version A</td>
</tr>
<tr>
<td>Bar Code Type</td>
<td>Modifier</td>
<td>In HRI?</td>
<td>Check Digit Calculation</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>----------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>13 – Codabar, 2-of-7, AIM USS-Codabar</td>
<td>X'02'</td>
<td>No</td>
<td>Codabar check digit calculation: 1. Sum of the data characters’ numerical values as described in a Codabar specification. All data characters are used, including the start and stop characters. 2. The check digit is described by the following equation where sum is the resulting value of step 1: ((16 - (\text{sum} \mod 16)) \mod 16)</td>
</tr>
<tr>
<td>17 – Code 128, AIM USS-128</td>
<td>X'02'</td>
<td>No</td>
<td>Code 128 check digit calculation: 1. Going left to right starting at the start character, sum the value of the start character and the weighted values of data and special characters. The weights are 1 for the first data or special character, 2 for the second, 3 for the third, and so forth. The stop character is not included in the calculation. 2. The check digit is modulo 103 of the resulting value of step 1.</td>
</tr>
<tr>
<td>24 – POSTNET</td>
<td>X'00' - X'04'</td>
<td>NA</td>
<td>The POSTNET check digit is ((10 - (\text{sum} \mod 10)) \mod 10), where sum is the sum of the ZIP code data.</td>
</tr>
<tr>
<td>26 – RM4SCC</td>
<td>X'00'</td>
<td>NA</td>
<td>The RM4SCC checksum digit is calculated by using an algorithm that weights each of the 4 bars within a character in relation to its position within the character.</td>
</tr>
<tr>
<td></td>
<td>X'01'</td>
<td>NA</td>
<td>None.</td>
</tr>
<tr>
<td>Bar Code Type</td>
<td>Modifier</td>
<td>In HRI?</td>
<td>Check Digit Calculation</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>---------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>27 – Japan Postal Bar Code JPOSTAL</td>
<td>X'00'</td>
<td>N/A</td>
<td>The Japan Postal Bar Code check digit calculation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Convert each character in the bar code data into decimal numbers. Numeric characters are converted to decimal. Each hyphen character is converted to the number 10. Each alphabetic character is converted to two numbers according to the symbology definition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, A becomes 11 and 0, B becomes 11 and 1, ..., J becomes 11 and 9, K becomes 12 and 0, L becomes 12 and 1, ..., T becomes 12 and 9, U becomes 13 and 0, V becomes 13 and 1, ..., and Z becomes 13 and 5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sum the resulting decimal numbers and calculate the remainder modulo 19.</td>
</tr>
<tr>
<td></td>
<td>X'01'</td>
<td>N/A</td>
<td>The check digit is 19 minus the remainder.</td>
</tr>
<tr>
<td>28 – DataMatrix (2DMATRIX)</td>
<td>X'00'</td>
<td>N/A</td>
<td>The DataMatrix symbology uses a Reed-Solomon error checking and correcting algorithm.</td>
</tr>
<tr>
<td>29 – MaxiCode 2DMAXI</td>
<td>X'00'</td>
<td>N/A</td>
<td>The MaxiCode symbology uses a Reed-Solomon error checking and correcting algorithm.</td>
</tr>
<tr>
<td>30 – PDF417</td>
<td>X'00' - X'01'</td>
<td>N/A</td>
<td>The PDF417 symbology uses a Reed-Solomon error checking and correcting algorithm.</td>
</tr>
<tr>
<td>31 – Australia Post Bar Code APOSTAL</td>
<td>X'01' - X'08'</td>
<td>No</td>
<td>The Australian Post Bar Code uses a Reed Solomon error correction code based on Galois Field 64.</td>
</tr>
<tr>
<td>32 — QR Code</td>
<td>X'02'</td>
<td>NA</td>
<td>The QR Code symbology uses a Reed-Solomon Error Checking and Correcting (ECC) algorithm.</td>
</tr>
<tr>
<td>33 — Code 93</td>
<td>X'00'</td>
<td>No</td>
<td>Both check digits (C and K) are calculated as Modulo 47 of the sum of the products of the data-character numerical values as described in the Code 93 specification and a weighting sequence. The start and stop codes are not included in the calculation.</td>
</tr>
<tr>
<td>34 — USPS Four-State</td>
<td>X'00' - X'03'</td>
<td>No</td>
<td>No check digit exists, but error detection and correction is added as part of the encoding process. See Specifications for the Four-State Barcode.</td>
</tr>
</tbody>
</table>
Bar code Exception Conditions

This section lists the BCOCA exception conditions that require detection by the bar code object processor when the bar code data structures are processed. It also specifies the standard actions that are taken.

Specification-Check Exceptions

A specification-check exception indicates that the bar code object processor received a bar code request with invalid or unsupported data parameters or values.

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC-0300</td>
<td>The bar code type that is specified in the BSD data structure is invalid or unsupported.</td>
</tr>
</tbody>
</table>

**Standard Action:** Stop bar code object processing.

<table>
<thead>
<tr>
<th>EC-0400</th>
<th>A font local ID specified in the BSD data structure is unsupported or not available.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For those symbologies that require a specific type style or code page for HRI, the BCOCA receiver cannot determine the type style or code page of the specified font.</td>
</tr>
</tbody>
</table>

**Standard Action:** If the requested font is not available, a font substitution can be made preserving as many characteristics as possible of the originally requested font while still preserving the original code page. Otherwise, stop bar code object processing.

Some bar code symbologies specify a set of type styles to be used for HRI data. Font substitution for HRI data must follow the bar code symbology specification that is used.

<table>
<thead>
<tr>
<th>EC-0500</th>
<th>The color that is specified in the BSD data structure is invalid or unsupported.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Standard Action:</strong> The device default color is used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EC-0505</th>
<th>The unit base that is specified in the BSD data structure is invalid or unsupported.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Standard Action:</strong> Stop bar code object processing.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EC-0600</th>
<th>The module width that is specified in the BSD data structure is invalid or unsupported.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Standard Action:</strong> The bar code object processor uses the closest smaller width. If the smaller value is less than the smallest supported width or zero, the bar code object processor uses the smallest supported value.</td>
</tr>
</tbody>
</table>
EC-0605
The units per unit base that is specified in the BSD data structure is invalid or unsupported.

**Standard Action**: Stop bar code object processing.

EC-0700
The element height that is specified in the BSD data structure is invalid or unsupported.

**Standard Action**: The bar code object processor uses the closest smaller height. If the smaller value is less than the smallest supported element height or zero, the bar code object processor uses the smallest supported value.

EC-0705
The presentation space extents that are specified in the BSD data structure are invalid or unsupported.

**Standard Action**: Stop bar code object processing.

EC-0800
The height multiplier that is specified in the BSD data structure is invalid.

**Standard Action**: The bar code object processor uses X'01'.
**Standard Action:** Stop bar code object processing.

**EC-0F00**
Either the matrix row size value or the number of rows value that is specified in the BSA data structure is unsupported. Both of these values must be within the range of supported sizes for the symbology.

**Standard Action:** Use X'0000' for the unsupported value so that an appropriate size is used based on the amount of symbol data.

**EC-0F01**
An invalid structured append sequence indicator was specified in the BSA data structure. For a Data Matrix symbol, the sequence indicator must be 1 - 16. For a MaxiCode symbol, the sequence indicator must be 1 - 8.

**Standard Action:** Present the bar code symbol without structured append information.

**EC-0F02**
A structured append sequence indicator that is specified in the BSA data structure is larger than the total number of structured append symbols.

**Standard Action:** Present the bar code symbol without structured append information.

**EC-0F03**
Mismatched structured append information was specified in the BSA data structure. One of the sequence-indicator and total-number-of-symbols parameters was X'00', but the other was not X".

**Standard Action:** Present the bar code symbol without structured append information.

**EC-0F04**
An invalid number of structured append symbols was specified in the BSA data structure. For a Data Matrix symbol, the total number of symbols must be 2 - 16. For a MaxiCode symbol, the total number of symbols must be 2 - 8.

**Standard Action:** Present the bar code symbol without structured append information.

**EC-0F05**
For a MaxiCode symbol, the symbol mode value that is specified in the BSA data structure is invalid.

**Standard Action:** Stop bar code object processing.

**EC-0F06**
For a PDF417 symbol, the number of data symbol characters per row value that is specified in the BSA data structure is invalid.

**Standard Action:** Stop bar code object processing.

**EC-0F07**
For a PDF417 symbol, the desired number of rows value that is specified in the BSA data structure is invalid.

This exception condition can also occur when the number of rows times the number of data symbol characters per row is greater than 928.
**Standard Action:** Proceed as if X'FF' was specified.

**EC-0F08**
For a PDF417 symbol, too much data is specified in the BSA data structure.

**Standard Action:** Stop bar code object processing.

**EC-0F09**
For a PDF417 symbol, the security level value that is specified in the BSA data structure is invalid.

**Standard Action:** Proceed as if security level 8 was specified.

**EC-0F0A**
An incompatible combination of Data Matrix parameters was specified in the BSA data structure. The following conditions can cause this exception condition:

- A structured append was specified (byte 10 not X'00'), but either the reader programming flag was set to B'1' or a hdr/trl macro was specified.
- The UCC/EAN FNC1 flag was set to B'1', but either the industry FNC1 flag was set to B'1', the reader programming flag was set to B'1', or a hdr/trl macro was specified.
- The industry FNC1 flag was set to B'1', but either the UCC/EAN FNC1 flag was set to B'1', the reader programming flag was set to B'1', or a hdr/trl macro was specified.
- The reader programming flag was set to B'1', but either a structured append was specified, one of the FNC1 flags was set to B'1', or a hdr/trl macro was specified.
- A hdr/trl macro was specified, but either a structured append was specified, one of the FNC1 flags was set to B'1', or the reader programming flag was set to B'1'.

**Standard Action:** Stop bar code object processing.

**EC-0F0B**
An invalid structured append file identification value was specified in the BSA data structure. Each byte of the 2-byte file identification value must be in the range X'01' - X'FE'.

**Standard Action:** Present the bar code symbol without structured append information.

**EC-0F0C**
A Macro PDF417 Control Block length value that is specified in the BSA data structure is invalid.

**Standard Action:** Stop bar code object processing.

**EC-0F0D**
Data within a Macro PDF417 Control Block that is specified in the BSA data structure is invalid.

**Standard Action:** Present the bar code symbol without a Macro PDF417 Control Block.
EC-0F0E
For a QR Code symbol, an invalid EBCDIC-code page value was specified in the BSA data structure.

Standard Action: Stop the bar code object processing.

EC-0F0F
For a QR Code symbol, an invalid version value was specified in the BSA data structure.

Standard Action: Proceed as if X'00' is specified.

EC-0F10
For a QR Code symbol, an invalid error-correction level value was specified in the BSA data structure.

Standard Action: Proceed as if X'03' is specified.

EC-0F11
For a QR Code symbol, an invalid combination of special-function flags was specified in the BSA data structure. Only one of the FNC1 flags can be B'1'.

Standard Action: Stop the bar code object processing.

EC-0F12
For a QR Code symbol, an invalid application-indicator value was specified in the BSA data structure.

Standard Action: Present the bar code symbol without structured append information.

EC-1000
The human-readable interpretation location that is specified in the BSA data structure is invalid.

Standard Action: Stop bar code object processing.

EC-1100
A portion of the bar code, including the bar and space patterns and the HRI, extends outside of either:
- The bar code presentation space
- The intersection of the mapped bar code presentation space and the controlling environment object area
- The maximum presentation area.

Standard Action: Stop bar code object processing.

All bar code symbols must be presented in their entirety. Whenever a partial bar code pattern is presented, for whatever reason, it is obscured to make it unscannable.

EC-1200
Invalid data is encountered in a UCC/EAN 128 symbol; one or more of the following conditions was encountered:
- FNC1 is not the first data character.
- Invalid application identifier (ai) value is encountered.
- Data for an ai does not match the ai definition.
- Insufficient (or no) data follows an ai.
• Too much data for an ai.
• Invalid use of FNC1 character.

**Standard Action:** Stop bar code object processing.

### Data-Check Exceptions

A data-check exception indicates that the bar code object processor detected an undefined character.

#### Exception Description

**EC-2100**

An invalid or undefined character, according to the rules of the symbology specification, is detected in the bar code data.

**Standard Action:** End bar code object processing.

### Data Matrix Special-Function Parameters

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name</th>
<th>Range</th>
<th>Meaning</th>
<th>BCD1 Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>BITS</td>
<td></td>
<td></td>
<td>Control flags</td>
<td></td>
</tr>
</tbody>
</table>
| bit 0  | EBCDIC | B'0’ B’1’ | EBCDIC-to-ASCII translation:  
  Do not translate  
  Convert data from EBCDIC to ASCII                                         | Not supported in BCD1 |
| bit 1  | Escape sequence handling | B’0’ B’1’ | Escape-sequence handling:  
  Process escape sequences  
  Ignore all escape sequences                                                | Not supported in BCD1 |
<p>| bits 2 - 7 |        | B'000000’ | Reserved                                                          |            |
| 6 - 7  | UBIN | Desired row size | X'00000' - X'FFE' | No size is specified. Matrix row size as allowed by symbology; see field description. | Not supported in BCD1 |
| 8 - 9  | UBIN | Desired number of rows | X'00000' - X'FFE' | No size is specified. Number of rows as allowed by symbology; see field description. | Not supported in BCD1 |
| 10     | UBIN | Sequence indicator | X'00’ - X'10’ | Structured append sequence indicator                                   | Not supported in BCD1 |
| 11     | UBIN | Total symbols    | X'00’ or X'02’ - X'10’ | Total number of structured-append symbols                             | Not supported in BCD1 |
| 12     | UBIN | File ID 1st byte | X'01’ - X'FE’ | High-order byte of a 2-byte unique file identification for a set of structured-append symbols | Not supported in BCD1 |
| 13     | UBIN | File ID 2nd byte | X'01’ - X'FE’ | Low-order byte of a 2-byte unique file identification for a set of structured-append symbols | Not supported in BCD1 |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name</th>
<th>Range</th>
<th>Meaning</th>
<th>BCD1 Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BITS</td>
<td>Special-function flags</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bit 0</td>
<td>UCC/EAN FNC1</td>
<td>B'0'</td>
<td>Alternate data type identifier:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B'1'</td>
<td>User-defined symbol</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Symbol conforms to UCC/EAN standards</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bit 1</td>
<td>Industry FNC1</td>
<td>B'0'</td>
<td>Alternate data type identifier:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B'1'</td>
<td>User-defined symbol</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Symbol conforms to industry standards</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>bit 2</td>
<td>Reader programming</td>
<td>B'0'</td>
<td>Reader programming symbol:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B'1'</td>
<td>Symbol encodes a data symbol.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Symbol encodes a message that is used to program the reader system.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bit 3 - 4</td>
<td>Hdr/Trl Macro</td>
<td>B'00'</td>
<td>Header and trailer instructions to the bar code reader:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B'01'</td>
<td>No header or trailer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B'10'</td>
<td>Use the 05 Macro header/trailer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B'11'</td>
<td>Use the 06 Macro header/trailer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No header or trailer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bit 5 - 7</td>
<td></td>
<td>B'000'</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

A desired symbol size can be specified in bytes 6 - 9, but the actual size of the symbol depends on the amount of data to be encoded. If not enough data is supplied, the symbol is padded with null data to reach the requested symbol size. If too much data is supplied for the requested symbol size, the symbol is bigger than requested, but the aspect ratio is maintained as closely as possible.

**Byte 5 Control flags.**

These flags control how the bar code data (bytes n+1 to end) is processed by the BCOCX receiver.

**Bit 0** EBCDIC-to-ASCII translation.

If this flag is B'0', the data is assumed to begin in the default character encodation and no translation is done.

If this flag is B'1', the BCOCX receiver converts each byte of the bar code data from EBCDIC code page 500 into ASCII code page 819 before this data is used to build the bar code symbol.

**Bit 1** Escape-sequence handling.

If this flag is B'0', each X'5C' (backslash) within the bar code data is treated as an escape character according to the Data Matrix symbology specification.

If this flag is B'1', each X'5C' within the bar code data is treated as a normal data character and therefore all escape sequences are ignored. In this case, no ECI code page switching can occur within the data.

**Note:** If the EBCDIC-to-ASCII translation flag is also set to B'1', all EBCDIC backslash characters (X'E0') are first converted into X'5C' before the escape-sequence handling flag is applied.

**Bits 2 - 7**

Reserved.
Bytes 6 - 7
Desired row size.

For a Data Matrix symbol, this parameter specifies the desired number of modules in each row, including the finder pattern. An even number of modules per row and an even number of rows must exist. Square symbols exist with sizes from 10x10 to 144x144, and rectangular symbols exist with sizes from 8x18 to 16x48 not including quiet zones. The following table lists the complete set of supported sizes. Exception condition EC-0F00 exists EC-0F00 if an unsupported size value is specified.

If X’0000’ is specified for this parameter, an appropriate row size is used based on the amount of symbol data.

<table>
<thead>
<tr>
<th>Table 28. Supported Sizes for a Data Matrix symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Symbol Size</strong></td>
</tr>
<tr>
<td><strong>Number of rows</strong></td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>36</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>44</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>72</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>88</td>
</tr>
<tr>
<td>96</td>
</tr>
<tr>
<td>104</td>
</tr>
<tr>
<td>120</td>
</tr>
<tr>
<td>132</td>
</tr>
<tr>
<td>144</td>
</tr>
</tbody>
</table>

Bytes 8 - 9
Desired number of rows.
For a Data Matrix symbol, this parameter specifies the desired number of rows, including the finder pattern. Exception condition EC-0F00 exists if an unsupported size value is specified.

If X'0000' is specified for this parameter, an appropriate number of rows are used based on the amount of symbol data.

**Byte 10**
Structured append sequence indicator.

Multiple data matrix bar code symbols (called structured appends) can be logically linked together to encode large amounts of data. The logically linked symbols can be presented on the same or on different physical media, and are logically recombined after they are scanned. 2 - 16 Data Matrix symbols can be linked. This parameter specifies where this symbol is logically linked (1 - 16) in a sequence of symbols.

If X'00' is specified for this parameter, this symbol is not part of a structured append. Exception condition EC-0F01 exists if an invalid sequence indicator value is specified. Exception condition EC-0F02 exists if the sequence indicator is larger than the total number of symbols (byte 11).

If this field is not X'00', the reader programming flag must be B'0' and the hdr/trl macro flags must be either B'00' or B'11'. Exception condition EC-0F0A exists if an incompatible combination of these parameters is specified.

**Byte 11**
Total symbols in a structured append.

This parameter specifies the total number of symbols (2 - 16) that is logically linked in a sequence of symbols.

If X'00' is specified for this parameter, this symbol is not part of a structured append. If this symbol is not part of a structured append, both bytes 10 and 11 must be 00 or exception condition EC-0F03 exists.

Exception condition EC-0F04 exists if an invalid number of symbols is specified.

**Byte 12**
High-order byte of structured append file identification.

This parameter specifies the high-order byte of a 2-byte unique file identification for a set of structured-append symbols, which helps ensure that the symbols from two different structured appends are not linked together. The low-order byte of the 2-byte field is specified in byte 13. Each of the 2 bytes can contain a value in the range X'01' - X'FE'.

This parameter is ignored if this symbol is not part of a structured append.

If this symbol is part of a structured append, but byte 12 contains an invalid value (X'00' or X'FF'), exception condition EC-0F0B exists.

**Byte 13**
Low-order byte of structured append file identification.

This parameter specifies the low-order byte of a 2-byte unique file identification for a set of structured-append symbols. The high-order byte of the 2-byte field is specified in byte 12. Each of the 2 bytes can contain a value in the range X'01' - X'FE'.

This parameter is ignored if this symbol is not part of a structured append.
If this symbol is part of a structured append, but byte 13 contains an invalid value (X'00' or X'FF'), exception condition EC-0F0B exists.

**Byte 14**

Special-function flags.

These flags specify special functions that can be used with a Data Matrix symbol.

**Bit 0** UCC/EAN FNC1 alternate data type identifier.

If this flag is B'1', an FNC1 is added in the first data position (or fifth position of a structured append symbol) to indicate that this symbol conforms to the UCC/EAN application identifier standard format. In this case, the industry FNC1 flag must be B'0', the reader programming flag must be B'0', and the hdr/trl macro must be B'00' or B'11'. Exception condition EC-0F0A exists if an incompatible combination of these parameters is specified.

**Bit 1** Industry FNC1 alternate data type identifier.

If this flag is B'1', an FNC1 is added in the second data position (or sixth position of a structured append symbol) to indicate that this symbol conforms to a particular industry standard format. In this case, the UCC/EAN FNC1 flag must be B'0', the reader programming flag must be B'0', and the hdr/trl macro must be B'00' or B'11'. Exception condition EC-0F0A exists if an incompatible combination of these parameters is specified.

**Bit 2** Reader programming.

If this flag is B'1', this symbol encodes a message that is used to program the reader system. In this case, the structured append sequence indicator must be X'00', the UCC/EAN FNC1 and industry FNC1 flags must both be B'0', and the hdr/trl macro flags must be either B'00' or B'11'. Exception condition EC-0F0A exists if an incompatible combination of these parameters is specified.

**Bits 3 - 4**

Header and trailer instructions to the bar code reader.

This field provides a means of instructing the bar code reader to insert an industry-specific header and trailer around the symbol data.

If this field is B'00' or B'11', no header or trailer is inserted. If this field is B'01', the bar code symbol contains a 05 Macro codeword. If this field is B'10', the bar code symbol contains a 06 Macro codeword.

If these flags are B'01' or B'10', the structured append sequence indicator must be X'00', the UCC/EAN FNC1 and industry FNC1 flags must both be B'0', and the reader programming flag must be B'0'. Exception condition EC-0F0A exists if an incompatible combination of these parameters is specified.

**Bits 5 - 7**

Reserved.
## MaxiCode Special-Function Parameters

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name</th>
<th>Range</th>
<th>Meaning</th>
<th>BCD1 Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>BITS</td>
<td></td>
<td></td>
<td><strong>Control flags</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>bit 0</strong> EBCDIC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B’0’ B’1’</td>
<td>EBCDIC-to-ASCII translation:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not translate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Convert data from EBCDIC to ASCII</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>bit 1</strong> Escape sequence handling</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B’0’ B’1’</td>
<td>Escape-sequence handling:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Process escape sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ignore all escape sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>bits 2 - 7</strong> Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CODE</td>
<td>Symbol mode</td>
<td>X’02’ X’03’ X’04’ X’05’ X’06’</td>
<td><strong>Mode 2</strong> <strong>Mode 3</strong> <strong>Mode 4</strong> <strong>Mode 5</strong> <strong>Mode 6</strong></td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>7</td>
<td>UBIN</td>
<td>Sequence indicator</td>
<td>X’00’ - X’08’</td>
<td><strong>Structured append sequence indicator</strong></td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>8</td>
<td>UBIN</td>
<td>Total symbols</td>
<td>X’00’ or X’02’ - X’08’</td>
<td><strong>Total number of structured-append symbols</strong></td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>9</td>
<td>BITS</td>
<td></td>
<td></td>
<td><strong>Special-function flags</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>bit 0</strong> Zipper B’0’ B’1’</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No zipper pattern</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Vertical zipper pattern on right</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>bit 1 - 7</strong> Reserved</td>
<td></td>
</tr>
</tbody>
</table>

**Byte 5** Control flags.

These flags control how the bar code data (bytes n+1 to end) is processed by the BCOCA receiver.

**Bit 0** EBCDIC-to-ASCII translation.

If this flag is B’0’, the data is assumed to begin in the default character encodation and no translation is done.

If this flag is B’1’, the BCOCA receiver converts each byte of the bar code data from EBCDIC code page 500 into ASCII code page 819 before this data is used to build the bar code symbol.

**Bit 1** Escape-sequence handling.

If this flag is B’0’, each X’5C’ (backslash) within the bar code data is treated as an escape character according to the MaxiCode symbology specification.

If this flag is B’1’, each X’5C’ within the bar code data is treated as a normal data character and therefore all escape sequences are ignored. In this case, no ECI code page switching can occur within the data.

**Note:** If the EBCDIC-to-ASCII translation flag is also set to B’1’, all EBCDIC backslash characters (X’E0’) are first converted into X’5C’ before the escape-sequence handling flag is applied.
Bits 2 - 7
Reserved.

Byte 6  Symbol mode.

Mode 2
Structured Carrier Message - numeric postal code.

This mode is designed for use in the transport industry, encoding the postal code, country code, and service class with the postal code that is numeric. The bar code data must be structured as described in B.2.1 and B.3.1 of the AIM International Symbology Specification - MaxiCode. The postal code, country code, and service class are placed in the primary message portion of the MaxiCode symbol and the rest of the bar code data is placed in the secondary message portion of the MaxiCode symbol. The first part of the bar code data includes in order the postal code, country code, and service class, which are separated by the [GS] character (X'1D').

This information can be preceded by the character sequence "[)>RS01GSyy", where RS and GS are single characters and yy are two decimal digits that represent a year. This character sequence is represented in hex bytes X'5B293E1E3031Dxxxx', where each xx is a value X'30' - X'39'. This sequence indicates that the message conforms to particular open system standards. This first portion of the bar code data must be encoded by using the MaxiCode default character set (ECI 000003 = ISO 8859-1). This first portion of the bar code data must not contain the backslash escape character to change the ECI character set. The postal code must be 1 - 9 decimal digits with each digit that is represented by the byte values from X'30' - X'39'. The country code must be 1 - 3 decimal digits with each digit a byte value X'30' - X'39'. The service code must also be 1 - 3 decimal digits, again with each digit a byte value X'30' - X'39'. The primary message portion of the MaxiCode symbol uses Enhanced Error Correction (EEC) and the secondary message portion of the MaxiCode symbol uses Standard Error Correction (SEC).

When the postal code portion of the Structured Carrier Message is numeric, mode 2 must be used.

Mode 3
Structured Carrier Message - alphanumeric postal code.

This mode is designed for use in the transport industry, encoding the postal code, country code, and service class with the postal code that is alphanumeric. The bar code data must be structured as described in B.2.1 and B.3.1 of the AIM International Symbology Specification - MaxiCode. The postal code, country code, and service class are placed in the primary message portion of the MaxiCode symbol and the rest of the bar code data is placed in the secondary message portion of the MaxiCode symbol. The first part of the bar code data includes in order the postal code, country code, and service class, which are separated by the [GS] character (X'1D').

This information can be preceded by the character sequence "[)>RS01GSyy", where RS and GS are single characters and yy are two decimal digits that represent a year. This character sequence is represented in hex bytes X'5B293E1E3031Dxxxx', where each xx is a value from X'30' - X'39'. This sequence indicates that the message conforms to particular open system standards. This first portion of the
the bar code data must be encoded by using the MaxiCode default character set (ECI 000003 = ISO 8859-1). This first portion of the bar code data must not contain the backslash escape character to change the ECI character set. The postal code must be 1 - 6 alphanumeric characters with each character one of the printable characters in MaxiCode Code Set A. Postal codes fewer than 6 characters are padded with trailing spaces; postal codes longer than 6 characters are truncated. These characters include the letters A to Z (X'41' - X'5A'), the space character (X'20'), the special characters (X'22' - X'2F'), the decimal digits (X'30' - X'39'), and the colon (X'3A'). The country code must be 1 - 3 decimal digits with each digit a byte value X'30' - X'39'. The service code must also be 1 - 3 decimal digits, again with each digit a byte value X'30' - X'39'. The primary message portion of the MaxiCode symbol uses Enhanced Error Correction (EEC) and the secondary message portion of the MaxiCode symbol uses Standard Error Correction (SEC).

When the postal code portion of the Structured Carrier Message is alphanumeric, mode 3 must be used.

**Mode 4**

Standard Symbol.

The symbol employs EEC for the Primary Message and SEC for the Secondary Message. The first nine codewords are placed in the Primary Message and the rest of the codewords are placed in the Secondary Message. This mode provides for a total of 93 codewords for data. If the bar code data consists of only characters from MaxiCode Code Set A, the number of codewords matches the number of bar code data characters. However, if the bar code data contains other characters, the number of codewords is greater than the number of bar code data characters due to the overhead of switching to and from the different code sets. The Code Set A consists of the byte values X'0D', X'1C' to X'1E', X'20', X'22' to X'3A', and X'41' to X'5A'.

**Mode 5**

Full ECC Symbol.

The symbol employs EEC for the Primary Message and EEC for the Secondary Message. The first nine codewords are placed in the Primary Message and the rest of the codewords are placed in the Secondary Message. This mode provides for a total of 77 codewords for data. If the bar code data consists of only characters from MaxiCode Code Set A, the number of codewords matches the number of bar code data characters. However, if the bar code data contains other characters, the number of codewords is greater than the number of bar code data characters due to the overhead of switching to and from the different code sets. The Code Set A consists of the byte values X'0D', X'1C' to X'1E', X'20', X'22' to X'3A', and X'41' to X'5A'.

**Mode 6**

Reader Program, SEC.

The symbol employs EEC for the Primary Message and SEC for the Secondary Message. The data in the symbol is used to program the bar code reader system. The first nine codewords are placed in the
Primary Message and the rest of the codewords are placed in the Secondary Message. This mode provides for a total of 93 codewords for data. If the bar code data consists of only characters from MaxiCode Code Set A, the number of codewords matches the number of bar code data characters. However, if the bar code data contains other characters, the number of codewords is greater than the number of bar code data characters due to the overhead of switching to and from the different code sets. The Code Set A consists of the byte values X'0D', X'1C' to X'1E', X'20', X'22' to X'3A', and X'41' to X'5A'.

Exception condition EC-0F05 exists if an invalid symbol-mode value is specified.

**Byte 7** Structured append sequence indicator.

Multiple MaxiCode bar code symbols (called structured appends) can be logically linked together to encode large amounts of data. The logically linked symbols can be presented on the same or on different physical media, and are logically recombined after they are scanned. MaxiCode symbols 2 - 8 can be linked. This parameter specifies where this particular symbol is logically linked (1&–8) in a sequence of symbols.

If X'00' is specified for this parameter, this symbol is not part of a structured append. Exception condition EC-0F01 exists if an invalid sequence indicator value is specified. Exception condition EC-0F02 exists if the sequence indicator is larger than the total number of symbols (byte 8).

**Byte 8** Total symbols in a structured append.

This parameter specifies the total number of symbols (2 - 8) that is logically linked in a sequence of symbols.

If X'00' is specified for this parameter, this symbol is not part of a structured append. If this symbol is not part of a structured append, both bytes 6 and 7 must be X'00', or exception condition EC-0F03 exists.

Exception condition EC-0F04 exists if an invalid number of symbols is specified.

**Byte 9** Special-function flags.

These flags specify special functions that can be used with a MaxiCode symbol.

**Bit 0** Zipper pattern.

If this flag is B'1', a vertical zipper-like test pattern and a contrast block are printed to the right of the symbol. The zipper provides a quick visual check for printing distortions. If the symbol presentation space is rotated, the zipper and contrast block are rotated along with the symbol.

To maintain consistency among printers, the zipper pattern and contrast block must approximate the guideline dimensions that are shown in Figure 144 on page 526. The zipper pattern and contrast block is made up of several filled rectangles that must be created such that each rectangle is as close to the specified dimensions as possible for the particular printer pel resolution, then the pattern is repeated to yield an evenly spaced zipper pattern and contrast block.
Guideline Dimensions for the Zipper and Contrast Block

Contrast block anchor point

This pattern repeats for a total of 9 bars, with each bar 2/203 inch by 28/203 inch. The space between each pair of bars is 1/203 inch.

The contrast block anchor point is 38/203 inch directly above the zipper anchor point.

Zipper pattern anchor point

This pattern repeats for approximately one inch (a total of 40 of these zipper teeth at 203 DPI). Each of the zipper teeth is made up of three 2/203 inch by 12/203 inch rectangles.

The space between each pair of teeth is 1/203 inch.

The zipper anchor point is 19/203 inch right of the rightmost column of hexagons that forms the MaxiCode Symbol and is aligned with the top of the MaxiCode symbol.

Figure 144. Example of a MaxiCode Bar Code Symbol with Zipper and Contrast Block

Bits 1 - 7
Reserved.

PDF417 Special-Function Parameters

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name</th>
<th>Range</th>
<th>Meaning</th>
<th>BCD1 Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>BITS</td>
<td>EBCDIC</td>
<td>B’0’</td>
<td>EBCDIC-to-ASCII translation:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B’1’</td>
<td>Do not translate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Convert data from EBCDIC to ASCII</td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td></td>
<td>Escape sequence handling</td>
<td>B’0’</td>
<td>Escape-sequence handling:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B’1’</td>
<td>Process escape sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ignore all escape sequences</td>
<td></td>
</tr>
<tr>
<td>bits 2 - 7</td>
<td></td>
<td></td>
<td>B’000000’</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>UBIN</td>
<td>Data symbols</td>
<td>X’01’ - X’1E’</td>
<td>Number of data symbol characters per row</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>Offset</td>
<td>Type</td>
<td>Name</td>
<td>Range</td>
<td>Meaning</td>
<td>BCD1 Range</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>------------</td>
<td>----------------</td>
<td>----------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>7</td>
<td>UBIN</td>
<td>Rows</td>
<td>X'03' - X'5A'</td>
<td>Desired number of rows</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X'FF'</td>
<td>Minimum necessary rows</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>UBIN</td>
<td>Security</td>
<td>X'00' - X'08'</td>
<td>Security level</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>9 - 10</td>
<td>UBIN</td>
<td>Macro length</td>
<td>X'0000' - X'7FED'</td>
<td>Length of Macro PDF417 Control Block that follows</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>11 - n</td>
<td>UBIN</td>
<td>Macro data</td>
<td>Any value</td>
<td>Data for Macro PDF417 Control Block</td>
<td>Not supported in BCD1</td>
</tr>
</tbody>
</table>

**Byte 5** Control flags.

These flags control how the bar code data is processed by the BCOCA receiver.

**Bit 0** EBCDIC-to-ASCII translation (for bytes 11 to end).

If this flag is B'0', the data is assumed to begin in the default character encoding and no translation is done.

If this flag is B'1', the BCOCA receiver converts each byte of the bar code data (bytes n+1 to end) and each byte of the Macro PDF417 Control Block data (bytes 11 - n) from a subset of EBCDIC code page 500 into the default character encoding (GLI 0) before this data is used to build the bar code symbol. This translation covers 181 code points, which include alphanumerics and many symbols; the 75 code points that are not covered by the translation do not occur in EBCDIC and are mapped to X'7F' (127). See Figure 145 on page 528 for a picture that shows the 181 EBCDIC code points that can be translated.

The EBCDIC-to-ASCII translation flag must not be used if any of the 75 code points that have no EBCDIC equivalent are needed for the bar code data or for the Macro PDF417 Control Block data.

Table 5 in the *Uniform Symbology Specification – PDF417* shows the full set of GLI 0 code points; from this set, the 75 code points that have no EBCDIC equivalent are as follows: 158, 159, 169, 176–224, 226–229, 231–240, 242–245, 247, 249, 251–252, and 254.

The 75 EBCDIC code points that are not covered by the translation and are thus mapped into X'7F' are as follows:

X'04', X'06', X'08' - X'0A', X'14' - X'15', X'17',
X'1A' - X'1B', X'20' - X'24', X'28' - X'2C', X'30' - X'31',
X'33' - X'36', X'38' - X'3B', X'3E' - X'46', X'62',
X'64' - X'66', X'6A', X'70', X'72' - X'78', X'80',
X'8C' - X'8E', X'9D' X'9F', X'AC' - X'AF', X'B4' - X'B6',
X'B9', X'BC' - X'BF', X'CA', X'CF', X'DA', X'EB',
X'ED' - X'EF', X'FA' - X'FB', X'FD' - X'FF'
<table>
<thead>
<tr>
<th>Hex Digits</th>
<th>0-</th>
<th>1-</th>
<th>2-</th>
<th>3-</th>
<th>4-</th>
<th>5-</th>
<th>6-</th>
<th>7-</th>
<th>8-</th>
<th>A-</th>
<th>B-</th>
<th>C-</th>
<th>D-</th>
<th>E-</th>
<th>F-</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>DLE</td>
<td>(SP)</td>
<td>&amp;</td>
<td>_</td>
<td>\</td>
<td>0</td>
<td>\</td>
<td>0</td>
<td>\</td>
<td>0</td>
<td>\</td>
<td>0</td>
<td>\</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>SOH</td>
<td>DC1</td>
<td>(RSP)</td>
<td>é</td>
<td>/</td>
<td>£</td>
<td>a</td>
<td>j</td>
<td>£</td>
<td>A</td>
<td>J</td>
<td>\</td>
<td>1</td>
<td>\</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>STX</td>
<td>DC2</td>
<td>SYN</td>
<td>à</td>
<td>è</td>
<td>b</td>
<td>k</td>
<td>s</td>
<td>¥</td>
<td>B</td>
<td>K</td>
<td>S</td>
<td>2</td>
<td>\</td>
<td>0</td>
</tr>
<tr>
<td>-3</td>
<td>ETX</td>
<td>DC3</td>
<td></td>
<td>è</td>
<td>À</td>
<td>c</td>
<td>l</td>
<td>t</td>
<td>C</td>
<td>L</td>
<td>T</td>
<td>3</td>
<td>\</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>HT</td>
<td>LF</td>
<td></td>
<td>à</td>
<td>è</td>
<td>d</td>
<td>m</td>
<td>u</td>
<td>D</td>
<td>M</td>
<td>U</td>
<td>4</td>
<td>\</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td>IS</td>
<td></td>
<td></td>
<td>á</td>
<td>i</td>
<td>ê</td>
<td>Å</td>
<td>Æ</td>
<td>E</td>
<td>N</td>
<td>V</td>
<td>5</td>
<td>\</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-6</td>
<td>BS</td>
<td>ETB</td>
<td></td>
<td>í</td>
<td>o</td>
<td>w</td>
<td>F</td>
<td>O</td>
<td>W</td>
<td>6</td>
<td>\</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-7</td>
<td>DEL</td>
<td>ESC</td>
<td>EOT</td>
<td>å</td>
<td>i</td>
<td>Ä</td>
<td>g</td>
<td>p</td>
<td>x</td>
<td>G</td>
<td>P</td>
<td>X</td>
<td>7</td>
<td>\</td>
<td>0</td>
</tr>
<tr>
<td>-8</td>
<td>CAN</td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>C</td>
<td>h</td>
<td>q</td>
<td>y</td>
<td>H</td>
<td>Q</td>
<td>Y</td>
<td>8</td>
<td>\</td>
<td>0</td>
</tr>
<tr>
<td>-9</td>
<td>EM</td>
<td></td>
<td></td>
<td>ñ</td>
<td>Ñ</td>
<td>ï</td>
<td>ñ</td>
<td>ñ</td>
<td>I</td>
<td>R</td>
<td>Z</td>
<td>9</td>
<td>\</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-A</td>
<td>VT</td>
<td></td>
<td></td>
<td>[</td>
<td>]</td>
<td>※</td>
<td>#</td>
<td>»</td>
<td>&quot;</td>
<td>\</td>
<td>0</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>-B</td>
<td>FF</td>
<td>FS</td>
<td></td>
<td>*</td>
<td>%</td>
<td>@</td>
<td>æ</td>
<td>ò</td>
<td>ü</td>
<td>Ö</td>
<td>Ü</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>-C</td>
<td>CR</td>
<td>GS</td>
<td>ENQ</td>
<td>(</td>
<td>)</td>
<td>_</td>
<td>'</td>
<td>&quot;</td>
<td>'</td>
<td>ò</td>
<td>ü</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>-D</td>
<td>SO</td>
<td>RS</td>
<td>ACK</td>
<td>À</td>
<td>¹</td>
<td>Å</td>
<td>Æ</td>
<td>ò</td>
<td>ü</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>-E</td>
<td>SI</td>
<td>US</td>
<td>BEL</td>
<td>!</td>
<td>?</td>
<td>&quot;</td>
<td>±</td>
<td>ÿ</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td>\</td>
<td></td>
</tr>
</tbody>
</table>

Note: If the EBCDIC-to-ASCII translation flag is also set to B1', all EBCDIC backslash characters (X'5C') are first converted into X'5C' before the escape-sequence handling flag is applied.

Bits 2 - 7
Reserved.

Figure 145. Subset of EBCDIC code page 500 that can be translated to GLI 0

Bit 1 Escape-sequence handling (for bytes n+1 to end).
If this flag is B0', each X'5C' (backslash) within the bar code data is treated as an escape character according to the PDF417 symbology specification.
If this flag is B1', each X'5C' within the bar code data is treated as a normal data character and therefore all escape sequences are ignored. In this case, no GLI code page switching and no reader programming can occur within the data.

Note: If the EBCDIC-to-ASCII translation flag is also set to B1', all EBCDIC backslash characters (X'E0') are first converted into X'5C' before the escape-sequence handling flag is applied.

Bits 2 - 7
Reserved.
Byte 6  Data symbol characters per row.

This parameter specifies the number of data symbol characters per row. Each row consists of a start pattern, a left row indicator codeword, 1 - 30 data symbol characters, a right row indicator codeword (omitted in a truncated symbol), and a stop pattern. The aspect ratio of the bar code symbol is determined by the number of data symbol characters and the number of rows.

Exception condition EC-0F06 exists if an invalid number of data symbol characters per row is specified.

Because of the Error Checking and Correction (ECC) algorithm and the data compaction method that is used by the printer when the symbol is built, the number of data symbol characters is not necessarily the same as the number of characters in the bar code data.

Byte 7  Desired number of rows.

This parameter specifies the desired number of rows in the bar code symbol. From 3 - 90 rows can be specified or X'FF' can be specified to instruct the printer to generate the minimum number of rows necessary. The number of rows times the number of data symbol characters per row cannot exceed 928. Exception condition EC-0F07 exists if an invalid number of rows is specified.

The actual number of rows that are generated depends on the amount of data to be encoded and on the security level selected. If more rows than necessary are specified, the symbol is padded to fill the requested number of rows. If not enough rows are specified, enough extra rows are inserted by the printer to produce the symbol.

If too much data is specified to fit in the bar code symbol, exception condition EC-0F08 exists.

Byte 8  Security level.

This parameter specifies the desired security level for the symbol as a value 0 - 8. Each higher security level causes more error correction codewords to be added to the symbol. At a particular security level, a number of codewords can be missing or erased and the symbol can still be recovered. Also, PDF417 can recover from mis-decodes of codewords. The formula is: Maximum Limit >= Erasures + 2*Misdecodes. The relation of security level to error correction capability is as follows:

<table>
<thead>
<tr>
<th>Security level</th>
<th>Maximum Limit &gt;= Erasures + 2*Misdecodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>126</td>
</tr>
<tr>
<td>7</td>
<td>254</td>
</tr>
<tr>
<td>8</td>
<td>510</td>
</tr>
</tbody>
</table>
For example, at security level 6, a total of 126 codewords can be either missing or destroyed and the entire symbol can still be completely recovered. The following table provides a recommended security level for various amounts of data:

<table>
<thead>
<tr>
<th>Number of Data Codewords</th>
<th>Recommended Security Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 40</td>
<td>2</td>
</tr>
<tr>
<td>41 - 160</td>
<td>3</td>
</tr>
<tr>
<td>161 - 320</td>
<td>4</td>
</tr>
<tr>
<td>321 - 863</td>
<td>5</td>
</tr>
</tbody>
</table>

Exception condition EC-0F09 exists if an invalid security level value is specified.

**Bytes 9 - 10**

Length of Macro PDF417 Control Block that follows.

This field specifies the length of a Macro PDF417 Control Block that follows in bytes 11 - n; this length does not contain the length field itself.

If X'0000' is specified, no Macro PDF417 Control Block is specified as a special function and this is the last field of the special-function parameters; what follows is the bar code data itself.

If a value between X'0001' and X'7FED' is specified, the BCOCA receiver builds a Macro PDF417 Control Block at the end of the bar code symbol by using the data in bytes 11 - n.

If an invalid length value is specified, exception condition EC-0F0C exists.

**Bytes 11 - n**

Macro PDF417 Control Block data.

The special codewords \922, \923, and \928 are used for coding a Macro PDF417 Control Block as defined in section G.2 of the Uniform Symbology Specification PDF417, but these codewords must not be used within the bar code data. Exception condition EC-2100 exists if one of these escape sequences is found in the bar code data. If a Macro PDF417 Control Block is needed, it is specified in bytes 11 - n.

The data for this Macro PDF417 Control Block must adhere to the following format; exception condition EC-0F0D exists if this format is not followed:

- For the symbol in a Macro PDF417 that represents the last segment of the Macro PDF417, the data must contain \922. For all symbols in a Macro PDF417, except the one representing the last segment:
  - A Macro PDF417 Control Block starts with a \928 escape sequence.
  - Followed by 1 to 5 numeric digits (bytes values X'30' - X'39'), representing a segment index value 1 - 99,999.
  - Followed by a variable number of escape sequences that contain values \000 - \899, representing the file ID.
  - Followed by zero or more optional fields, with the following layout:
    - \923 escape sequence, signaling an optional field
    - Escape sequence that contains the field designator with a value \000 - \006
- Followed by a variable number of text characters (for field designators \000, \003, and \004) or a variable number of numeric digits (for field designators \001, \002, \005, and \006). The field designators are defined in Table G1 of the Uniform Symbology Specification. For text characters, the byte values must be X’09’, X’0A’, X’0D’, or from X’20’ through X’7E’. These values represent the uppercase letters A - Z, the lowercase letters a - z, and the digits 0-9, plus some punctuation and special characters (for GLI 0). For the numeric digits, the byte values must be X’30’ - X’39’.

- For field designator \001, the 1 - 5 numeric digits that follow represent the segment count. This value must be greater than or equal to the segment index value.

- For field designator \002, the 1 - 11 numeric digits that follow represent the time stamp on the source file that is expressed as the elapsed time in seconds since January 1, 1970 00:00 GMT.

- For field designator \005, one or more numeric digits must follow.

- For field designator \006, the 1 - 5 numeric digits that follow represent the decimal value of the 16-bit CRC checksum over the entire source file. This checksum value must be a decimal value from 0 through 65,535.

The file name, segment count, time stamp, sender, addressee, file size, and checksum are provided in the optional fields of the Macro PDF417 Control Block and the BCOCA receiver makes no attempt to calculate or verify these values (other than the previously stated restrictions). If the Macro PDF417 Control Block data does not follow these rules, exception condition EC-0F0D exists. The Uniform Symbology Specification PDF417 has the following additional claims. The BCOCA receiver does not check for these claims nor does it report any exceptions conditions if these claims are violated:

- If the optional Segment Count is given in the Macro PDF417 Control Block of one of the segments (symbols) of the macro, then it must be used in all of the segments (symbols) of the macro.

- All optional fields, other than the Segment Count, need to appear only in one of the segments (symbols) of the macro.

- If an optional field with the same field designator appears in more than one segment (symbol) of the same macro, then it must appear identically in every segment (symbol).

### QR Code Special-Function Parameters

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name</th>
<th>Range</th>
<th>Meaning</th>
<th>BCD1 Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>BITS</td>
<td></td>
<td></td>
<td>Control flags</td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td>EBCDIC</td>
<td>B’0’</td>
<td>B’1’</td>
<td>EBCDIC-to-ASCII translation:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not translate</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Convert data from EBCDIC to ASCII</td>
<td></td>
</tr>
<tr>
<td>bit 1</td>
<td>Escape sequence handling</td>
<td>B’0’</td>
<td>B’1’</td>
<td>Escape-sequence handling:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Process escape sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ignore all escape sequences</td>
<td></td>
</tr>
<tr>
<td>bits 2 - 7</td>
<td></td>
<td>B’000000’</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Type</td>
<td>Name</td>
<td>Range</td>
<td>Meaning</td>
<td>BCD1 Range</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>6</td>
<td>CODE</td>
<td>EBCDIC code page</td>
<td>X'00' - X'03'</td>
<td>EBCDIC code page that is used to encode data:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- No code page specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Code page 500 (International #5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Code page 290 (Japanese Katakana Ext.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Code page 1027 (Japanese Latin Extended)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CODE</td>
<td>Version</td>
<td>X'00' - X'28'</td>
<td>Version of symbol:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Smallest symbol</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Version number (1 - 40)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CODE</td>
<td>Error correction level</td>
<td>X'00' - X'03'</td>
<td>Level of error correction:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Level L (7% recovery)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Level M (15% recovery)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Level Q (25% recovery)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Level H (30% recovery)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>UBIN</td>
<td>Sequence indicator</td>
<td>X'00' - X'10'</td>
<td>Structured append sequence indicator</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td>10</td>
<td>UBIN</td>
<td>Total symbols</td>
<td>X'00' - X'02'</td>
<td>Total number of structured-append symbols</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X'10'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>UBIN</td>
<td>Parity Data</td>
<td>X'00' - X'FF'</td>
<td>Structured append parity data</td>
<td>X'00' - X'FF'</td>
</tr>
<tr>
<td>12</td>
<td>BITS</td>
<td>Special-function flags</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UCC/EAN FNC1</td>
<td>B'0' - B'1'</td>
<td>Alternate data type identifier:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- User-defined symbol</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Symbol conforms to UCC/EAN standards</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Industry FNC1</td>
<td>B'0' - B'1'</td>
<td>Alternate data type identifier:</td>
<td>Not supported in BCD1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- User-defined symbol</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- Symbol conforms to industry standards</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits 2 - 7</td>
<td>B'0000000'</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>CODE</td>
<td>Application indicator</td>
<td>See field description</td>
<td>Application indicator for Industry FNC1</td>
<td>Not supported in BCD1</td>
</tr>
</tbody>
</table>

A desired symbol size is specified by the version parameter (byte 7), but the actual size of the symbol depends on the amount of data to be encoded. If not enough data is supplied, the symbol is padded with null data to reach the requested symbol size. If too much data is supplied for the requested symbol size, the symbol is bigger than requested and is the smallest symbol that can accommodate that amount of data.

**Byte 5** Control flags.

These flags control how the bar code data (bytes n+1 to end) is processed by the BCOC receiver; the receiver can be an IPDS printer or any other product that processes BCOC objects.

**Bit 0** EBCDIC-to-ASCII translation.

If this flag is B'0', the data is assumed to begin in the default character encodation (ECI 000020) and no translation is done.

If this flag is B'1' and an EBCDIC code page is selected in byte 6, the BCOC receiver converts each byte of the bar code data from
the EBCDIC code page that is specified in byte 6 into ASCII code page 897 before this data is used to build the bar code symbol. The following EBCDIC code pages can be used to encode the bar code data:

- Code page 500 (International #5) — specify X'01' in byte 6.
  Only 128 of the characters within ECI 000020 can be specified in code page 500. The code page 500 characters that can be translated are shown in
- Code page 290 (Japanese Katakana Extended) — specify X'02' in byte 6.
- Code page 1027 (Japanese Latin Extended) — specify X'03' in byte 6.

EBCDIC characters that are not defined within ECI 000020 are mapped to X'7F' (DEL).

**Bit 1** Escape-sequence handling.

If this flag is B'0', each X'5C' within the bar code data is treated as an escape character according to the QR Code symbology specification.

If this flag is B'1', each X'5C' within the bar code data is treated as a normal data character and therefore all escape sequences are ignored. In this case, no ECI code page switching can occur within the data.

**Note:** If the EBCDIC-to-ASCII translation flag is also set to B'1', all EBCDIC characters are first converted into X'5C' before the escape-sequence handling flag is applied.

**Byte 6** Conversion.

When the EBCDIC-to-ASCII translation flag is B'1', this parameter specifies the method that is used to convert EBCDIC input data into the default character encodation. When the EBCDIC-to-ASCII translation flag is B'0', this parameter is not used and must be set to X'00'.

For the first three values (used when the input data is encoded with a single-byte EBCDIC code page), this parameter identifies the EBCDIC code page that encodes single-byte EBCDIC bar code data. The following EBCDIC code pages are supported:

- X'01' Code page 500 (International #5).
  Only 128 of the characters within ECI 000020 can be specified in code page 500. The code page 500 characters that can be translated are shown in [Figure 146 on page 535](#).

- X'02' Code page 290 (Japanese Katakana Extended).

- X'03' Code page 1027 (Japanese Latin Extended).

For the remaining values (used when the input data is SOSI), this parameter identifies the desired conversion from EBCDIC SOSI input data to a specific mixed-byte ASCII encoding.

**Note:** The values X'04' - X'09' are defined for the Additional Bar Code Parameters (X'7B') triplet that is used with AFP Line Data; these values are not valid within a BCOCA object that is built for a non-line-data environment, such as MO:DCA and IPDS. See [Advanced Function](#).
The following choices are supported:

\[ X'04' \]
CCSID 1390 - CCSID 943

**Convert from:**
CCSID 1390 – Extended Japanese Katakana-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS & DBCS euro).

**Convert to:**
CCSID 943 – Japanese PC Data Mixed for Open environment (Multi-vendor code): 6878 JIS X 0208-1990 chars, 386 IBM selected DBCS chars, 1880 UDC (X'F040' - X'F9FC').

\[ X'05' \]
CCSID 1399 - CCSID 943

**Convert from:**
CCSID 1399 – Extended Japanese Latin-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS & DBCS euro).

**Convert to:**
CCSID 943 – Japanese PC Data Mixed for Open environment (Multi-vendor code): 6878 JIS X 0208-1990 chars, 386 IBM selected DBCS chars, 1880 UDC (X'F040' - X'F9FC').

\[ X'06' \]
CCSID 1390 - CCSID 932

**Convert from:**
CCSID 1390 – Extended Japanese Katakana-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS & DBCS euro).

**Convert to:**
CCSID 932 – Japanese PC Data Mixed including 1880 UDC.

\[ X'07' \]
CCSID 1399 - CCSID 932

**Convert from:**
CCSID 1399 – Extended Japanese Latin-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS & DBCS euro).

**Convert to:**
CCSID 932 – Japanese PC Data Mixed including 1880 UDC.

\[ X'08' \]
CCSID 1390 - CCSID 942

**Convert from:**
CCSID 1390 – Extended Japanese Katakana-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS & DBCS euro).

**Convert to:**
CCSID 942 – Japanese PC Data Mixed including 1880 UDC, Extended SBCS.
Convert from:
CCSID 1399 – Extended Japanese Latin-Kanji Host Mixed for JIS X0213 including 6205 UDC, Extended SBCS (includes SBCS & DBCS euro).

Convert to:
CCSID 942 – Japanese PC Data Mixed including 1880 UDC, Extended SBCS.

EBCDIC characters that are not defined within ECI 000020 are mapped to the substitute character, X'7F' or X'FCFC'; exception condition EC-2100 exists when an undefined character is encountered.

Exception condition EC-0F0E exists if an invalid or unsupported conversion value is specified.

### Byte 7
Version of symbol.

---

<table>
<thead>
<tr>
<th>Hex Digits</th>
<th>0-</th>
<th>1-</th>
<th>2-</th>
<th>3-</th>
<th>4-</th>
<th>5-</th>
<th>6-</th>
<th>7-</th>
<th>8-</th>
<th>9-</th>
<th>A-</th>
<th>B-</th>
<th>C-</th>
<th>D-</th>
<th>E-</th>
<th>F-</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-</td>
<td>NUL</td>
<td>DLE</td>
<td>(SP)</td>
<td>&amp;</td>
<td>_</td>
<td>{</td>
<td>}</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-</td>
<td>SOH</td>
<td>DC1</td>
<td>a</td>
<td>j</td>
<td>A</td>
<td>J</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-</td>
<td>STX</td>
<td>DC2</td>
<td>b</td>
<td>k</td>
<td>s</td>
<td>¥</td>
<td>B</td>
<td>K</td>
<td>S</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-</td>
<td>ETX</td>
<td>DC3</td>
<td>c</td>
<td>l</td>
<td>t</td>
<td>C</td>
<td>L</td>
<td>T</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-</td>
<td>HT</td>
<td>LF</td>
<td>d</td>
<td>m</td>
<td>u</td>
<td>D</td>
<td>M</td>
<td>U</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-</td>
<td>LF</td>
<td>LF</td>
<td>e</td>
<td>n</td>
<td>v</td>
<td>E</td>
<td>N</td>
<td>V</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-</td>
<td>BS</td>
<td>ETO</td>
<td>f</td>
<td>o</td>
<td>w</td>
<td>F</td>
<td>O</td>
<td>W</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-</td>
<td>DEL</td>
<td>ESC</td>
<td>g</td>
<td>P</td>
<td>x</td>
<td>G</td>
<td>P</td>
<td>X</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-</td>
<td>CAN</td>
<td>LF</td>
<td>h</td>
<td>q</td>
<td>y</td>
<td>H</td>
<td>Q</td>
<td>Y</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-</td>
<td>EM</td>
<td>LF</td>
<td>i</td>
<td>r</td>
<td>z</td>
<td>I</td>
<td>R</td>
<td>Z</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 146. Subset of EBCDIC Code Page 500 that can be translated to ECI 000020

---

Appendix D. More About Bar Code Parameters 535
This parameter specifies the desired size of the symbol; each version specifies a particular number of modules per row and column. The size of each square module is specified by the module width parameter (byte 17 in the BSD). The following table lists the complete set of supported versions. Exception condition EC-0F0F exists if an invalid version value is specified.

<table>
<thead>
<tr>
<th>Version</th>
<th>Symbol Size</th>
<th>Version</th>
<th>Symbol Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (X'00')</td>
<td>smallest</td>
<td>21 (X'15')</td>
<td>101x101</td>
</tr>
<tr>
<td>1 (X'01')</td>
<td>21x21</td>
<td>22 (X'16')</td>
<td>105x105</td>
</tr>
<tr>
<td>2 (X'02')</td>
<td>25x25</td>
<td>23 (X'17')</td>
<td>109x109</td>
</tr>
<tr>
<td>3 (X'03')</td>
<td>29x29</td>
<td>24 (X'18')</td>
<td>113x113</td>
</tr>
<tr>
<td>4 (X'04')</td>
<td>33x33</td>
<td>25 (X'19')</td>
<td>117x117</td>
</tr>
<tr>
<td>5 (X'05')</td>
<td>37x37</td>
<td>26 (X'1A')</td>
<td>121x121</td>
</tr>
<tr>
<td>6 (X'06')</td>
<td>41x41</td>
<td>27 (X'1B')</td>
<td>125x125</td>
</tr>
<tr>
<td>7 (X'07')</td>
<td>45x45</td>
<td>28 (X'1C')</td>
<td>129x129</td>
</tr>
<tr>
<td>8 (X'08')</td>
<td>49x49</td>
<td>29 (X'1D')</td>
<td>133x133</td>
</tr>
<tr>
<td>9 (X'09')</td>
<td>53x53</td>
<td>30 (X'1E')</td>
<td>137x137</td>
</tr>
<tr>
<td>10 (X'0A')</td>
<td>57x57</td>
<td>31 (X'1F')</td>
<td>141x141</td>
</tr>
<tr>
<td>11 (X'0B')</td>
<td>61x61</td>
<td>32 (X'20')</td>
<td>145x145</td>
</tr>
<tr>
<td>12 (X'0C')</td>
<td>65x65</td>
<td>33 (X'21')</td>
<td>149x149</td>
</tr>
<tr>
<td>13 (X'0D')</td>
<td>69x69</td>
<td>34 (X'22')</td>
<td>153x153</td>
</tr>
<tr>
<td>14 (X'0E')</td>
<td>73x73</td>
<td>35 (X'23')</td>
<td>157x157</td>
</tr>
<tr>
<td>15 (X'0F')</td>
<td>77x77</td>
<td>36 (X'24')</td>
<td>161x161</td>
</tr>
<tr>
<td>16 (X'10')</td>
<td>81x81</td>
<td>37 (X'25')</td>
<td>165x165</td>
</tr>
<tr>
<td>17 (X'11')</td>
<td>85x85</td>
<td>38 (X'26')</td>
<td>169x169</td>
</tr>
<tr>
<td>18 (X'12')</td>
<td>89x89</td>
<td>39 (X'27')</td>
<td>173x173</td>
</tr>
<tr>
<td>19 (X'13')</td>
<td>93x93</td>
<td>40 (X'28')</td>
<td>177x177</td>
</tr>
<tr>
<td>20 (X'14')</td>
<td>97x97</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If X'00' is specified for this parameter, an appropriate row/column size is used based on the amount of symbol data; the smallest symbol that can accommodate the amount of data is produced.

**Byte 8** Level of error correction.

This parameter specifies the level of error correction to be used for the symbol. Each higher level of error correction causes more error correction codewords to be added to the symbol and therefore leaves fewer codewords for symbol data. See the QR Code symbology specification for more information about how many codewords are available for symbol data for each version and error-correction level combination.

Four different levels of Reed-Solomon error correction can be selected:

- Level L (X'00') allows recovery of 7% of symbol codewords.
- Level M (X'01') allows recovery of 15% symbol codewords.
- Level Q (X'02') allows recovery of 25% symbol codewords.
- Level H (X'03') allows recovery of 30% symbol codewords.
Exception condition EC-0F10 exists if an invalid level-of-error-correction value is specified.

**Byte 9** Structured append sequence indicator.

Multiple QR Code bar code symbols (called structured appends) can be logically linked together to encode large amounts of data. The logically linked symbols can be presented on the same or on different physical media, and are logically recombined after they are scanned. From 2 - 16 QR Code symbols can be linked. This parameter specifies where this symbol is logically linked (1 - 16) in a sequence of symbols.

If X'00' is specified for this parameter, this symbol is not part of a structured append. Exception condition EX-0F01 exists if an invalid sequence indicator value is specified. Exception condition EC-0F02 exists if the sequence indicator is larger than the total number of symbols (byte 10).

**Byte 10**

Total number of structured-append symbols.

This parameter specifies the total number of symbols (2 - 16) that is logically linked in a sequence of symbols.

If X'00' is specified for this parameter, this symbol is not part of a structured append. If this symbol is not part of a structured append, both bytes 9 and 10 must be X'00', or exception condition EC-0F03 exists.

Exception condition EC-0F04 exists if an invalid number of symbols is specified.

**Byte 11**

Structured append parity data.

This parameter specified parity data for a structured append symbol. The parity-data value must be calculated from the entire message that is broken into structured-append symbols; the parity-data value must be the same in each of the structured-append symbols.

The parity-data value is obtained by XORing byte by byte the ASCII/JIS values of all the original input data before division into structured-append symbols.

If this symbol is not a structured append, this parameter is ignored and must be set to X'00'.

**Byte 12**

Special-function flags.

These flags specify special functions that can be used with a QR Code symbol.

**Bit 0** UCC/EAN FNC1 alternate data type identifier.

If this flag is B'1', this QR Code symbol indicates that it conforms to the UCC/EAN application identifiers standard. In this case, the industry FNC1 flag must be B'0'. Exception condition EC-0F11 exists if an incompatible combination of these bits is specified.

**Bit 1** Industry FNC1 alternate data type identifier.

If this flag is B'1', this QR Code symbol indicates that it conforms to the specific industry or application specifications previously agreed with AIM International. In this case, the UCC/EAN FNC1
flag must be B’0’. Exception condition EC-0F11 exists if an incompatible combination of these bits is specified.

When this flag is B’1’, an application indicator is specified in byte 13.

**Bits 2 - 7**
Reserved.

**Byte 13**
Application indicator for Industry FNC1.

When the Industry FNC1 flag is B’1’, this parameter specifies an application indicator. The application indicator is a 1-byte value that is specified either as an alphabetic value (from the ASCII set a-z, A-Z) plus 100 or as a two-digit number (00 - 99 represented as a hexadecimal value).

For example:
- For application indicator a (ASCII value X’61’), specify X’C5’.
- For application indicator Z (ASCII value X’5A’), specify X’BE’.
- For application indicator 01, specify X’01’.
- For application indicator 99, specify X’63’.

When the Industry FNC1 flag is B’0’, this parameter is ignored and must be set to X’00’.

Exception condition EC0F12 exists if an invalid application-indicator value is specified.
Appendix E. Set Media Origin (SMO)

This appendix explains how to generate the correct Set Media Origin (SMO) information in a form definition.

Background

IBM 3800-3

The first AFP printer was the IBM 3800-3 delivered in 1983. It had a single, fixed media origin or top, left corner as shown in Figure 147 on page 540. Print data could be placed to print across, down, or up the page. Across and down are shown. Page definitions (pagedefs) were used to place the line data on the page. IBM provided page definitions for common line spacings for both portrait and landscape orientation. The two most common paper sizes used were 12” x 8.5” (shown on left) and 9.5” x 11” (shown on right). As shown in the following figure, different page definitions are required to print the same output on the two different sizes of paper. For example, an across page definition is used to print landscape output on 12” x 8.5” paper and a down page definition is used for landscape output on 9.5” x 11” paper. Forms, like 12” x 8.5”, that have a width greater than their length are referred to as wide continuous forms and forms, like 9.5” x 11”, that have a length greater than their width are called narrow continuous forms.
IBM 3820

In 1985 the IBM 3820 was introduced. It was the first IBM AFP cut-sheet printer. Like the 3800-3, it also had a single, fixed media origin as shown. The 3820 could print in the same directions as the 3800-3 plus one new direction, back. As shown in Figure 148 on page 541, it had similar print direction characteristics as a 3800-3 using 9.5" x 11" paper.
The Problem

Many customers had both 3800-3s and 3820s. The 3800-3 was the fastest printer of its time at 215 impressions per minute. To achieve that speed, it had to use the 12" x 8.5" paper. This paper had the additional benefit of reducing maintenance charges because the 3800s usage charge was based on the number of linear feet printed. The use of the shorter, 12" x 8.5" paper cost less per page than the longer, 9.5" x 11" paper. Unfortunately, when the 12" x 8.5" paper is used on the 3800-3 it causes an incompatibility with the 3820. As shown in Figure 149, the 3800-3 with 12" x 8.5" paper has a different media orientation than the 3820.

Another Problem

AFP also provides the ability to print image data. The print direction that is coded in the page definition controls how the text data is oriented on the page.
Unfortunately it has no effect on data that is contained within an image. In Figure 150, an image that contains the word, LOGO is placed on the 3800-3 and 3820 pages. It is oriented as desired on the 3820 but appears to be rotated 90 degrees counterclockwise on the 3800-3. In reality, the image is not rotated, the paper is. To compensate for this, a special, rotated version of the image was built for use on the 3800-3 and care was taken to use the proper version on both printers.

![Figure 150. Rotated Text and Image Data](image)

When IBM was developing its second, continuous forms AFP printer, the 3835, it was decided to develop a solution for the problem of different media origins between cut-sheet and continuous forms printers. The AFP architecture was enhanced to allow the media origin to be moved to any of the four corners of the paper. This function is called Set Media Origin and is controlled by a value in the form definition (formdef). This value is called the Medium Origin and is set by using the Page Printer Formatting Aid (PPFA) PRESENT and DIRECTION parameters on the FORMDEF and COPYGROUP commands.

**FORMDEF PRESENT and DIRECTION Parameters**

The use of PRESENT and DIRECTION parameters on the FORMDEF is confusing. The examples in "The SMO Reference Pages" on page 543 show a sheet of paper and the placement of the Medium Origin for a particular PRESENT and DIRECTION combination. Notice that for a particular Medium Origin combination, the results for cut-sheet paper, wide continuous forms paper, narrow continuous forms paper, and Cut Sheet Emulation are different. This was done to guarantee that the application pages would be oriented in the same manner on all printer and media types. Cut Sheet Emulation (CSE) divides a continuous forms printer’s paper web into two equal sheetlets.

The PRESENT parameter has two possible values of PORTRAIT and LANDSCAPE. The DIRECTION has values of ACROSS and DOWN. An additional optional parameter is CUTSHEET. It specifies whether the SMO command is sent to cut-sheet printers. This is invoked by coding a value of YES. If NO is coded or allowed to default the media origin on cut-sheet printers is not changed and is at

---

9. Some older printers do not support the SMO function.
the default location, which is equivalent to a form definition with PRESENT PORTRAIT and DIRECTION ACROSS. Coding the N_UP parameter (1-up, 2-up, 3-up, 4-up) has the same effect as coding CUTSHEET. The CUTSHEET parameter was created to allow form definitions that do not have N-UP coded to enable SMO on cut-sheet printers.

When Print Services Facility (PSF) converts an output file to Intelligent Printer Data Stream (IPDS) to send to the printer, it includes an SMO value if SMO is supported by the printer. This value is calculated by examining the Medium Origin of the form definition and then checking to see whether the printer is cut-sheet, wide continuous, or narrow continuous. CSE on the printer reports back wide or narrow continuous. PSF/IPM then specifies the proper SMO. The resulting SMO values are included in the figures for completeness.

**The SMO Reference Pages**

The following pages show how the four possible SMO combinations behave on the following four different printer types:
- Cut-sheet
- Continuous forms with wide paper
- Continuous forms with narrow paper
- Continuous forms with Cut Sheet Emulation enabled.

Four summary pages show the media origin and print directions for the listed printer types. These summary pages are followed by detail pages. The top of each detail page shows the:
- Printer type.
- Values for Presentation and Direction that are coded on the form definition that is used.
- Medium Origin value that this combination produces.

Each detail page shows six images of a single paper sheet. The media origin is also called the top, left corner. Three sheets show how data is oriented for:
- Simplex sheet or front of a duplex sheet
- Back of a normal duplex sheet
- Back of a tumble duplex sheet

Simulated hole punches and rotation axis lines are provided to help visualize how the back of a duplex sheet relates to the front. An image that is made from a photograph is included on these sheets to help understand the orientation. The top, left corner of the image is in the same corner as the media origin.

One sheet shows the four print directions and how they relate to the media origin; one sheet shows sample line data that is formatted by using a page definition that is coded with an across print direction; and one sheet shows data that is formatted with a page definition that is coded with a down print direction. The page definition name, its direction, total lines, and lines-per-inch spacing are given for each page definition page.
Cut-Sheet Printer

Figure 151. Cut-Sheet Printer Summary of Set Medium Origin
Figure 152. Cut-Sheet Printer with a Medium Origin of X'00'
Figure 153. Cut-Sheet Printer with a Medium Origin of X’01’
Figure 154. Cut-Sheet Printer with a Medium Origin of X’04’
Figure 155. Cut-Sheet Printer with a Medium Origin of X’05’
Wide Continuous Forms Paper

Figure 156. Wide Continuous Forms Printer Paper Summary of Set Media Origin
Figure 157. Wide Continuous Forms Printer Paper with a Medium Origin of X'00'
Continuous Forms Printer, Wide Paper, Landscape Across Formdef, MO:DCA Medium Origin = X’01’, IPDS SMO = X’03’

Figure 158. Wide Continuous Forms Printer Paper with a Medium Origin of X’01’
Figure 159. Wide Continuous Forms Printer Paper with a Medium Origin of X'04'
Presentation: Landscape  Direction: Down

Medium Origin: X’05’

Continuous Forms Printer, Wide Paper,
Landscape Down Formdef,
MO:DCA Medium Origin = X'05', IPDS SMO = X'02'

Figure 160. Wide Continuous Forms Printer Paper with a Medium Origin of X'05'
Narrow Continuous Forms Paper

Figure 161. Narrow Continuous Forms Printer Paper Summary of Set Media Origin
Continuous Forms Printer, Narrow Paper, Portrait Across Formdef, MO:DCA Medium Origin = X’00’, IPDS SMO = X’00’

Figure 162. Narrow Continuous Forms Printer Paper with a Medium Origin of X’00’
Presentation: Landscape Direction: Across

Medium Origin: X’01’

Continuous Forms Printer, Narrow Paper,
Landscape Across Formdef,
MO:DCA Medium Origin = X’01’, IPDS SMO = X’01’

Figure 163. Narrow Continuous Forms Printer Paper with a Medium Origin of X’01’
Presentation: Portrait  Direction: Down

Medium Origin: X’04’

Continuous Forms Printer, Narrow Paper,
Portrait Down Formdef,
MO:DCA Medium Origin = X’04’, IPDS SMO = X’03’

Figure 164. Narrow Continuous Forms Printer Paper with a Medium Origin of X’04’
Figure 165. Narrow Continuous Forms Printer Paper with a Medium Origin of X'05'
Cut-Sheet Emulation

Figure 166. Cut-Sheet Emulation for Continuous Forms Printer Wide and Narrow Paper Set Media Origin
Appendix F. PPFA Keywords

A keyword is a word in PPFA that must be entered exactly as shown. Keywords cannot be used as second names for commands (like FONT and OVERLAY), which can have two positional parameters as names. This is less restrictive than prior versions of PPFA.

Note: When keywords are longer than 5 characters, they can be abbreviated to the first 5 characters. The shorthand form of the keyword must also be avoided as a name. For example, since PAGEH and CONDI are 5 character forms for PAGEHEADER and CONDITION, they cannot be used as second names.

PPFA reserved keywords are shown in the following list:

<table>
<thead>
<tr>
<th>ABSOLUTE</th>
<th>COPYGROUP</th>
<th>HILITE</th>
<th>OFFSET</th>
<th>RESOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACROSS</td>
<td>HRI</td>
<td>HRIFONT</td>
<td>OPCODE</td>
<td>RGB</td>
</tr>
<tr>
<td>ADJUST</td>
<td>CP</td>
<td>INVOKE</td>
<td>OPERATION</td>
<td>ROTATION</td>
</tr>
<tr>
<td>AFIELD</td>
<td>CS</td>
<td>JG</td>
<td>OPOFFSET</td>
<td>SBCS</td>
</tr>
<tr>
<td>ALIGN</td>
<td>CUTSHEET</td>
<td>LAYOUT</td>
<td>OPOFFSET</td>
<td>SCOPE</td>
</tr>
<tr>
<td>AXIS1</td>
<td>CVERROR</td>
<td>LENGTH</td>
<td>OTHERWISE</td>
<td>SEGMENT</td>
</tr>
<tr>
<td>AXIS2</td>
<td>DBCS</td>
<td>LINE</td>
<td>OUTBIN</td>
<td>SETUNITS</td>
</tr>
<tr>
<td>BACK</td>
<td>DEFINE</td>
<td>LINEONE</td>
<td>OVERLAY</td>
<td>SOSIFONT</td>
</tr>
<tr>
<td>BARCODE</td>
<td>DEFINE CMRNAME</td>
<td>LINESP</td>
<td>OVROTATE</td>
<td>SPACE_THEN_PRINT</td>
</tr>
<tr>
<td>BCCOLOR</td>
<td>DELIMITER</td>
<td>LINETYPE</td>
<td>PAGECOUNT</td>
<td>SSASTERISK</td>
</tr>
<tr>
<td>BCXPARMS</td>
<td>DIRECTION</td>
<td>LINETYPE</td>
<td>PAGEDEF</td>
<td>START</td>
</tr>
<tr>
<td>BIN</td>
<td>DOWN</td>
<td>MEDIUM</td>
<td>PAGEFORMAT</td>
<td>SUBGROUP</td>
</tr>
<tr>
<td>BINERROR</td>
<td>DOFONT</td>
<td>METRICTECHNOLOGY</td>
<td>PAGEHEADER</td>
<td>SUPPBLANKS</td>
</tr>
<tr>
<td>BOTH</td>
<td>DRAGRAPHIC</td>
<td>METTECH</td>
<td>PAGENUM</td>
<td>SUPPRESSION</td>
</tr>
<tr>
<td>BOX</td>
<td>DUPLEX</td>
<td>MOD</td>
<td>PAGETRAILER</td>
<td>TEXT</td>
</tr>
<tr>
<td>BOXSIZE</td>
<td>ELLIPSE</td>
<td>MODWIDTH</td>
<td>PARTITION</td>
<td>TEXTERROR</td>
</tr>
<tr>
<td>CHANNEL</td>
<td>ENDSPACE</td>
<td>N_UP</td>
<td>POSITION</td>
<td>TONERSAVER</td>
</tr>
<tr>
<td>CIELAB</td>
<td>ENDSUBPAGE</td>
<td>NEWPAGE</td>
<td>PRELOAD</td>
<td>TRCREF</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>EXREF</td>
<td>NOGROUP</td>
<td>PRESENT</td>
<td>TYPE</td>
</tr>
<tr>
<td>CLRTRAP</td>
<td>FIELD</td>
<td>NOPRELOAD</td>
<td>PRINTDATA</td>
<td>UDTYPE</td>
</tr>
<tr>
<td>CMR</td>
<td>FILL</td>
<td>NORASTER</td>
<td>PRINTLINE</td>
<td>WHEN</td>
</tr>
<tr>
<td>CMRNAME</td>
<td>FINISH</td>
<td>OBID</td>
<td>PROCESSING</td>
<td>WIDTH</td>
</tr>
<tr>
<td>CMRTAGFIDELITY</td>
<td>FLASH</td>
<td>OBJECT</td>
<td>QUALITY</td>
<td>XATTR</td>
</tr>
<tr>
<td>CMYK</td>
<td>FLDNUM</td>
<td>OBKEEP</td>
<td>RADIUS</td>
<td>XLAYOUT</td>
</tr>
<tr>
<td>COLOR</td>
<td>FONT</td>
<td>OBNOKEEP</td>
<td>RASTER</td>
<td>XMSIZE</td>
</tr>
<tr>
<td>COLORVALUERR</td>
<td>FONTFID</td>
<td>OBTYPE</td>
<td>RATIO</td>
<td>XSPACE</td>
</tr>
<tr>
<td>COMMENT</td>
<td>FORMFID</td>
<td>OBXNAME</td>
<td>RECID</td>
<td>YMSIZE</td>
</tr>
<tr>
<td>COMPATPGP1</td>
<td>FRONT</td>
<td>OB2CMR</td>
<td>REFERENCE</td>
<td></td>
</tr>
<tr>
<td>COMPATPGP1</td>
<td>GRAPHID</td>
<td>OB2ID</td>
<td>RELATIVE</td>
<td></td>
</tr>
<tr>
<td>CONDITION</td>
<td>GRID</td>
<td>OB2RESOURCE</td>
<td>RENDER</td>
<td></td>
</tr>
<tr>
<td>CONSTANT</td>
<td>GROUP</td>
<td>OB2TYPE</td>
<td>REPEAT</td>
<td></td>
</tr>
<tr>
<td>COPIES</td>
<td>GRPHEADER</td>
<td>OB2XNAME</td>
<td>REPLACE</td>
<td></td>
</tr>
<tr>
<td>COPY</td>
<td>HEIGHT</td>
<td>OCA</td>
<td>RES</td>
<td></td>
</tr>
</tbody>
</table>
Appendix G. PPFA Media Names

Table 31 lists the PPFA media names, media types, and component identifiers.

**Note:** The range of component IDs 12,288 - 268,435,455 is reserved for user-defined media types.

<table>
<thead>
<tr>
<th>Media Name</th>
<th>Media Type</th>
<th>Component ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSNS ENV</td>
<td>North American business envelope</td>
<td>143</td>
</tr>
<tr>
<td>COM 10 ENV</td>
<td>Com10 envelope (9.5 x 4.125 in)</td>
<td>75</td>
</tr>
<tr>
<td>C5 ENV</td>
<td>C5 envelope (229 x 110 mm)</td>
<td>79</td>
</tr>
<tr>
<td>DL ENV</td>
<td>DL envelope (220 x 110 mm)</td>
<td>77</td>
</tr>
<tr>
<td>EXEC</td>
<td>North American executive (7.25 x 10.5 in)</td>
<td>65</td>
</tr>
<tr>
<td>INDEX CD</td>
<td>Index Card</td>
<td>150</td>
</tr>
<tr>
<td>ISO A3</td>
<td>ISO A3 white (297 x 420 mm)</td>
<td>10</td>
</tr>
<tr>
<td>ISO A3 CO</td>
<td>ISO A3 colored</td>
<td>11</td>
</tr>
<tr>
<td>ISO A4</td>
<td>ISO A4 white (210 x 297 mm)</td>
<td>0</td>
</tr>
<tr>
<td>ISO A4 CO</td>
<td>ISO A4 colored</td>
<td>1</td>
</tr>
<tr>
<td>ISO A4 TAB</td>
<td>ISO A4 tab (225 x 297 mm)</td>
<td>7</td>
</tr>
<tr>
<td>ISO A4 THD</td>
<td>ISO 1/3 A4</td>
<td>5</td>
</tr>
<tr>
<td>ISO A4 TR</td>
<td>ISO A4 transparent</td>
<td>2</td>
</tr>
<tr>
<td>ISO A5</td>
<td>ISO A5 white (148.5 x 210 mm)</td>
<td>20</td>
</tr>
<tr>
<td>ISO A5 CO</td>
<td>ISO A5 colored</td>
<td>21</td>
</tr>
<tr>
<td>ISO A6 PC</td>
<td>ISO A6 Postcard</td>
<td>152</td>
</tr>
<tr>
<td>ISO B4</td>
<td>ISO B4 white (257 x 364 mm)</td>
<td>30</td>
</tr>
<tr>
<td>ISO B4 CO</td>
<td>ISO B4 colored</td>
<td>31</td>
</tr>
<tr>
<td>ISO B4 ENV</td>
<td>ISO B4 envelope</td>
<td>83</td>
</tr>
<tr>
<td>ISO B5</td>
<td>ISO B5 white (176 x 250 mm)</td>
<td>40</td>
</tr>
<tr>
<td>ISO B5 CO</td>
<td>ISO B5 colored</td>
<td>41</td>
</tr>
<tr>
<td>ISO B5 ENV</td>
<td>ISO B5 envelope</td>
<td>73</td>
</tr>
<tr>
<td>ISO C4 ENV</td>
<td>ISO C4 envelope</td>
<td>93</td>
</tr>
<tr>
<td>ISO C5 ENV</td>
<td>ISO C5 envelope</td>
<td>103</td>
</tr>
<tr>
<td>ISO LNG ENV</td>
<td>ISO long envelope</td>
<td>113</td>
</tr>
<tr>
<td>JIS B4</td>
<td>JIS B4 (257 x 364 mm)</td>
<td>42</td>
</tr>
<tr>
<td>JIS B5</td>
<td>JIS B5 (182 x 257 mm)</td>
<td>43</td>
</tr>
<tr>
<td>JP PC</td>
<td>Japan postcard (Hagaki) (100 x 148 mm)</td>
<td>81</td>
</tr>
<tr>
<td>JP PC ENV</td>
<td>Japan postcard envelope (200 x 150 mm)</td>
<td>80</td>
</tr>
<tr>
<td>LEDGER</td>
<td>North American ledger (11 x 17 in)</td>
<td>67</td>
</tr>
<tr>
<td>LEGAL</td>
<td>North American legal white (8.5 x 14 in)</td>
<td>60</td>
</tr>
<tr>
<td>LEGAL CO</td>
<td>North American legal colored</td>
<td>61</td>
</tr>
</tbody>
</table>
### Table 31. Registered Media Types Sorted By Media Name (continued)

<table>
<thead>
<tr>
<th>Media Name</th>
<th>Media Type</th>
<th>Component ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEGAL TAB</td>
<td>Legal tab (9 x 14 in)</td>
<td>146</td>
</tr>
<tr>
<td>LEGAL 13</td>
<td>North American legal 13 (Folio) (8.5 x 14 in)</td>
<td>63</td>
</tr>
<tr>
<td>LETTER</td>
<td>North American letter white (8.5 x 11 in)</td>
<td>50</td>
</tr>
<tr>
<td>LETTER CO</td>
<td>North American letter colored</td>
<td>51</td>
</tr>
<tr>
<td>LETTER TAB</td>
<td>Letter tab (9 x 11 in)</td>
<td>145</td>
</tr>
<tr>
<td>LETTER TR</td>
<td>North American letter transparent</td>
<td>52</td>
</tr>
<tr>
<td>MON ENV</td>
<td>Monarch envelope (7.5 x 3.875 in)</td>
<td>76</td>
</tr>
<tr>
<td>RA3</td>
<td>Oversize A3 (16.923 x 12.007 in)</td>
<td>153</td>
</tr>
<tr>
<td>RA4</td>
<td>Oversize A4 (8.465 x 12.007 in)</td>
<td>162</td>
</tr>
<tr>
<td>STATEMNT</td>
<td>North American statement (5.5 x 8.5 in)</td>
<td>69</td>
</tr>
<tr>
<td>US PC</td>
<td>US Postcard</td>
<td>151</td>
</tr>
<tr>
<td>9x12 ENV</td>
<td>North American 9 x 12 envelope</td>
<td>133</td>
</tr>
<tr>
<td>10x13 ENV</td>
<td>North American 10 x 13 envelope</td>
<td>123</td>
</tr>
<tr>
<td>9x12 MAN</td>
<td>Manual (9 x 12 in)</td>
<td>147</td>
</tr>
<tr>
<td>8x10 MED</td>
<td>Media (8 x 10 in)</td>
<td>160</td>
</tr>
<tr>
<td>8x10.5 MED</td>
<td>Media (8 x 10.5 in)</td>
<td>148</td>
</tr>
<tr>
<td>8.5x10 MED</td>
<td>Media (8.5 x 10 in)</td>
<td>157</td>
</tr>
<tr>
<td>9x14 MED</td>
<td>Media (9 x 14 in)</td>
<td>149</td>
</tr>
<tr>
<td>12x18 MED</td>
<td>Media (12 x 18 in)</td>
<td>155</td>
</tr>
<tr>
<td>14x17 MED</td>
<td>Media (14 x 17 in)</td>
<td>154</td>
</tr>
<tr>
<td>14x18 MED</td>
<td>Media (14 x 18 in)</td>
<td>156</td>
</tr>
</tbody>
</table>
Appendix H. Fill Patterns for DRAWTGRAPHIC Commands

Figure 167. Fill Patterns for DRAWTGRAPHIC Commands
Appendix I. PPFA Messages and Codes

At the end of processing for each command, the maximum error level that is encountered during processing is printed on the system printer, providing the error was not caused by the system printer itself. The meaning of the return codes is shown in Table 32.

Table 32. Return Codes

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Code 0</td>
<td>I = Information; the command is processed.</td>
<td>PPFA did not encounter any problems. No warning, error, severe-error, or termination-error message was issued.</td>
</tr>
<tr>
<td>Return Code 4</td>
<td>W = Warning; the command is processed.</td>
<td>PPFA encountered at least one non-terminating error, which was solved by an assumption. At least one warning message was issued. No error, severe-error, or terminating-error message was issued. The requested function was most likely correctly done. The program ran to completion.</td>
</tr>
<tr>
<td>Return Code 8</td>
<td>E = Error; the command is partially processed.</td>
<td>PPFA encountered at least one error, but no severe or terminating error. A requested function might be partially incomplete.</td>
</tr>
<tr>
<td>Return Code 12</td>
<td>S = Severe error; the command is not processed.</td>
<td>PPFA encountered a severe error. The program ran to completion, but some of the functions requested were not done.</td>
</tr>
<tr>
<td>Return Code 16</td>
<td>T = Termination error; the job is ended.</td>
<td>PPFA encountered a terminating error. The program ended prematurely.</td>
</tr>
</tbody>
</table>

PPFA Messages and Their Meanings

The general format of the error message is as follows:

All messages consist of a standard seven-character prefix, followed by the message text:

AKQnnns THIS IS THE MESSAGE TEXT . . .

- AKQ is the three-character identifier of Page Printer Formatting Aid (PPFA).
- nnn is the message number.
- s is the message-severity indicator. The severity indicators are defined in Table 32.

Notes:

1. You cannot use the psfmsg command to view PPFA messages.
2. PPFA errors are written to a listing file.
3. PPFA issues a maximum of 269 user errors that are generated within a source file, and one additional message that is used for the message queue to indicate an out-of-storage condition.

AKQ001E END OF COMMENT (*/) IS NOT SPECIFIED.

Explanation: The end mark of a comment (*/) is not specified.

System action: The page definition or form definition
is not generated. The syntax check might be ended.

**Operator response:** Specify the end mark of a comment.

<table>
<thead>
<tr>
<th>Code</th>
<th>Error Message</th>
<th>Explanation</th>
<th>System action</th>
<th>Operator response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKQ002E</td>
<td><strong>DBCS STRING DOES NOT END WITH SHIFT-IN.</strong></td>
<td><strong>Explanation:</strong> DBCS strings in comments must end with shift-in.</td>
<td><strong>System action:</strong> The form definition or page definition is not generated. The syntax check continues from the next keyword.</td>
<td><strong>Operator response:</strong> Specify a valid subcommand.</td>
</tr>
<tr>
<td>AKQ003E</td>
<td><strong>LITERAL DOES NOT END WITH APOSTROPHE.</strong></td>
<td><strong>Explanation:</strong> A literal must end with an apostrophe.</td>
<td><strong>System action:</strong> The page definition is not generated. The syntax check continues, assuming an apostrophe.</td>
<td><strong>Operator response:</strong> Specify the required name.</td>
</tr>
<tr>
<td>AKQ004E</td>
<td><strong>DBCS LITERAL DOES NOT END WITH SHIFT-IN AND APOSTROPHE.</strong></td>
<td><strong>Explanation:</strong> A DBCS literal must end with shift-in and apostrophe.</td>
<td><strong>System action:</strong> The page definition is not generated. The syntax check continues, assuming the end of the DBCS literal at the end of a record.</td>
<td><strong>Operator response:</strong> Specify a valid literal that is ended by shift-in and apostrophe.</td>
</tr>
<tr>
<td>AKQ101E</td>
<td><strong>COMMAND SEQUENCE IS INVALID.</strong></td>
<td><strong>Explanation:</strong> The command sequence is invalid.</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues from a valid command.</td>
<td><strong>Operator response:</strong> Specify commands in a valid sequence.</td>
</tr>
<tr>
<td>AKQ102E</td>
<td><strong>INVALID COMMAND (erroneous entry) IS SPECIFIED.</strong></td>
<td><strong>Explanation:</strong> An invalid command is specified in the input data.</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues from a valid command.</td>
<td><strong>Operator response:</strong> Specify a valid command.</td>
</tr>
<tr>
<td>AKQ103E</td>
<td><strong>INVALID SUBCOMMAND (value) IS SPECIFIED.</strong></td>
<td><strong>Explanation:</strong> An invalid subcommand was specified in the input data. This message is often issued when a semicolon (;) is missing</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues from the next keyword.</td>
<td><strong>Operator response:</strong> Specify a valid subcommand.</td>
</tr>
<tr>
<td>AKQ104E</td>
<td><strong>(command or parameter name) NAME IS NOT SPECIFIED.</strong></td>
<td><strong>Explanation:</strong> The required name is not specified.</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues, assuming blanks or default as the name.</td>
<td><strong>Operator response:</strong> Specify the required name.</td>
</tr>
<tr>
<td>AKQ105E</td>
<td><strong>REQUIRED PARAMETER IN (subcommand name) IS NOT SPECIFIED.</strong></td>
<td><strong>Explanation:</strong> The subcommand that is indicated in the message requires a correct PPF A format.</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues, assuming the default values.</td>
<td><strong>Operator response:</strong> See the command reference section of this publication for help in specifying a valid subcommand parameter.</td>
</tr>
<tr>
<td>AKQ106E</td>
<td><strong>(command or parameter name) NAME IS SPECIFIED WITH INVALID SYNTAX.</strong></td>
<td><strong>Explanation:</strong> The required name is specified with invalid syntax.</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues, assuming the default values.</td>
<td><strong>Operator response:</strong> Specify a valid name.</td>
</tr>
<tr>
<td>AKQ107E</td>
<td><strong>PARAMETER IN (subcommand name) IS INVALID.</strong></td>
<td><strong>Explanation:</strong> The parameter in the subcommand is invalid (invalid format or out of range).</td>
<td><strong>System action:</strong> A page definition or form definition is not generated. The syntax check continues, assuming the default values as the parameter.</td>
<td><strong>Operator response:</strong> Specify a valid parameter value.</td>
</tr>
</tbody>
</table>
AKQ108E  (subcommand name) SUBCOMMAND IS DUPLICATED IN ONE COMMAND.

Explanation: The subcommand that is indicated in the message was specified more than once in the same command. Only one such subcommand is allowed within this command.

System action: A page definition or form definition is not generated. The syntax check continues, ignoring the duplicate subcommand.

Operator response: Delete one subcommand.

AKQ109E  (subcommand name) SUBCOMMAND CONFLICTS WITH (subcommand name) SUBCOMMAND.

Explanation: One subcommand conflicts with another (FONT, PRINTLINE, FIELD, OPCOUNT, OPPOS)

System action: A page definition or form definition is not generated. The syntax check continues, ignoring the latter subcommand.

User response: Delete one of the subcommands.

AKQ110E  THE VALUE OF THE (command name) SUBCOMMAND IS TOO LARGE OR TOO SMALL.

Explanation: The parameter in the subcommand is out of range.

IN  136.5
MM  3467.1
CM  346.7
POINTS  9828.0
PELS (L-units)  32760

These values are specified in:
FORMDEF N_UP OVERLAY relative_xpos relative_ypos
PAGEDEF PRINTLINE OVERLAY / SEGMENT relative_xpos relative_ypos

Note: The values that are specified for the CPI and LPI are set in the SETUNITS subcommand.

Operator response: Specify a valid parameter value.

AKQ111E  SUBCOMMAND SEQUENCE IS INVALID: (subcommand name) OCCURS AFTER (subcommand name)

Explanation: For example, a WHEN subcommand occurs after an OTHERWISE subcommand in a CONDITION command.

System action: A page definition or form definition is not generated. The syntax check continues, ignoring the subcommand.

Operator response: Reorder or rewrite the conditions.

AKQ112E  CONDITION COMMAND DOES NOT ALLOW '*' IN ITS START SUBCOMMAND.

Explanation: A relative position ("*", "* + n", or "* - n") was specified in a START subcommand of a CONDITION command.

System action: A page definition or form definition is not generated. The syntax check continues from the valid subcommand.

Operator response: Specify an absolute starting position.

AKQ113E  MORE THAN ONE 'WHEN' SUBCOMMAND SPECIFIED THE CHANGE PARAMETER.

Explanation: More than one WHEN subcommand specified CHANGE for its field comparison.

System action: A page definition or form definition is not generated. The syntax check continues from the valid subcommand.

Operator response: Remove the extra subcommands that specify the CHANGE parameter.

AKQ114E  NUMBER OF PARAMETERS EXCEED LIMIT FOR (subcommand name) SUBCOMMAND OR KEYWORD.

Explanation: The named subcommand or keyword in the message limits the number of parameters that can be coded with a single subcommand or keyword. The number of parameters that can be coded with the named subcommand or keyword is defined in


System action: The form definition is not generated. The syntax check continues from the valid subcommand.

Operator response: Remove the extra parameters.

AKQ115E  REQUIRED PARAMETER(S) (PARM1, PARM2, ...) IN (COMMAND OR SUBCOMMAND) IS (ARE) NOT SPECIFIED.

Explanation: This generic message indicates that one or more parameters are missing on a subcommand or
command. For example, a DRAWGRAPHIC BOX must have a BOXSIZE subcommand coded.

System action: A page or form definition is not generated.

Operator response: Provide the correct parameters on the command or subcommand.

AKQ116E PARAMETER (PARM1) IN (COMMAND OR SUBCOMMAND) IS INVALID.

Explanation: This generic message indicates that a parameter in a subcommand or command is invalid.

System action: A page or form definition is not generated.

Operator response: Provide the correct parameter on the command or subcommand.

AKQ117E PARAMETER (PARM1) IN (COMMAND OR SUBCOMMAND) IS DUPLICATED.

Explanation: This generic message indicates that a parameter in a subcommand or command is coded more than once. For example, ...LINEWT LIGHT BOLD... shows two different line weights in the same subcommand.

System action: A page or form definition is not generated.

Operator response: Remove one of the parameters.

AKQ118E MUTUALLY EXCLUSIVE PARAMETERS ON THE (INSERT1) COMMAND OR SUBCOMMAND ARE DUPLICATED.

Explanation: A command or subcommand contains more than one mutually exclusive parameter. For example, the PAGECOUNT subcommand on the PAGEDEF command cannot have both STOP and CONTINUE coded.

System action: A page or form definition is not generated.

Operator response: Remove one of the parameters.

AKQ119E GRAPHICS-TYPE (BOX, LINE, CIRCLE, ELLIPSE) MUST IMMEDIATELY FOLLOW DRAWGRAPHIC.

Explanation: The DRAWGRAPHIC command must have the graphics type (BOX, LINE, CIRCLE, ELLIPSE) immediately following the command.

System action: A page or form definition is not generated.

Operator response: Code one of the graphics types.
AKQ203W  (command name) NAME IS DUPLICATED.

Explanation: The required name must be unique for OVERLAY, COPYGROUP, FONT, PAGEFORMAT, OBJECT, or SUPPRESSION.

System action: A page definition or form definition is generated.

Operator response: Specify a unique name.

AKQ204E  (object) NAME IS DUPLICATED.

Explanation: The name must be unique (OVERLAY, COPYGROUP, FONT, PAGEFORMAT, SEGMENT).

System action: A page definition or form definition is not generated. The syntax check continues.

Operator response: Specify a unique name.

AKQ205E  PAGEFORMAT (pageformat name) WAS NOT FOUND IN THIS PAGE DEFINITION.

Explanation: A WHEN or OTHERWISE subcommand of CONDITION specifies a PAGEFORMAT name that is not found in the page definition that is processed.

System action: A page definition or form definition is not generated. The syntax check continues.

Operator response: Specify a pageformat name that is in the page definition.

AKQ206E  CONDITION (condition name) HAS ALREADY BEEN DEFINED.

Explanation: A CONDITION command specifies LENGTH, WHEN, or OTHERWISE, and the condition with this condition name is already defined by an earlier CONDITION command.

System action: A page definition is not generated. The syntax check continues.

Operator response: Define the condition only the first time it occurs.

AKQ210E  THE RELATIVE POSITION VALUE EXCEEDS THE ALLOWED RANGE

System action: The value that is specified for the relative x position or relative y position on the N_UP subcommand (for an OVERLAY) or PRINTLINE command (for an OVERLAY or SEGMENT) exceeds the range of +32760 to -32760 L-units. For example, assuming the default of 240 pels per inch is being used, the values must be equal to, or less than these units:

<table>
<thead>
<tr>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>136.5</td>
</tr>
<tr>
<td>MM</td>
<td>346.7</td>
</tr>
<tr>
<td>CM</td>
<td>346.7</td>
</tr>
</tbody>
</table>

The value that is specified for CPI or LPI in the SETUNITS command determines whether the value exceeds 32760 L-units.

System action: The page definition or form definition is not generated. The syntax check continues.

Operator response: Correct the relative x and y position values within the allowed range.

AKQ211E  FRONT/BACK SIDE IS NOT SPECIFIED FOR DUPLEX.

Explanation: The SUBGROUP specified with BACK does not exist after the SUBGROUP specified with FRONT, or the SUBGROUP specified with FRONT does not exist before the SUBGROUP specified with BACK.

System action: A form definition is not generated. The syntax check continues.

Operator response: Specify subgroups for both sides.

AKQ212W  PAPER SIDE IS SPECIFIED FOR SIMPLEX.

Explanation: A subgroup that is specified with BOTH, FRONT, or BACK is invalid with single-sided printing.

System action: A form definition is generated, which ignores the subcommand that specifies the paper side.

User response: Either delete the subcommand that specified the paper side or specify DUPLEX.

AKQ213E  LOGICAL PAGE POSITION EXCEEDS THE LIMIT.

Explanation: The logical page position that is specified by the OFFSET subcommand in the FORMDEF or COPYGROUP command exceeds the limits.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ214E  MORE THAN 127 SUPPRESSIONS ARE SPECIFIED IN ONE FORMDEF.

Explanation: More than 127 suppressions are specified in one FORMDEF.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.
AKQ215E  MORE THAN 127 OVERLAYS ARE SPECIFIED IN ONE COPYGROUP.

Explanation: More than 127 OVERLAYS are specified in one copy group. PPFA can issue this message for an N_UP subcommand that specifies more than 127 overlays.

System action: No form definition is generated. The syntax check continues.

User response: Correct the error.

AKQ216E  MORE THAN ONE RASTER OVERLAY IS SPECIFIED IN ONE COPYGROUP.

Explanation: More than one raster OVERLAY is specified in one copy group.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ217W  LOGICAL PAGE POSITION FOR BACK SIDE OF PAGE SPECIFIED IN SIMPLEX PROCESSING

Explanation: The logical-page position that is specified by the OFFSET subcommand in a FORMDEF or COPYGROUP command for the back side of a page was specified, but simplex was specified in a COPYGROUP command.

System action: A form definition is generated, with the back side logical page position included, as if duplex is specified. The syntax check continues.

Operator response: Correct the error by specifying duplex in the COPYGROUP command or remove the second set of coordinates in the OFFSET subcommand.

AKQ218E  MORE THAN 255 COPIES ARE SPECIFIED IN ONE COPYGROUP.

Explanation: More than 255 copies are specified in a COPYGROUP.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ219E  MORE THAN 127 SUBGROUPS ARE SPECIFIED IN ONE COPYGROUP.

Explanation: More than 127 subgroups are specified in a COPYGROUP.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ220E  MORE THAN 8 OVERLAYS ARE SPECIFIED IN ONE SUBGROUP.

Explanation: More than eight overlays are specified in one SUBGROUP.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ221E  MORE THAN 8 SUPPRESSIONS ARE SPECIFIED IN ONE SUBGROUP.

Explanation: More than eight suppressions are specified in one SUBGROUP.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ222W  DIFFERENT NUMBERS OF COPIES ARE SPECIFIED FOR EACH SIDE OF DUPLEX.

Explanation: The number of copies for BACK side is not equal to the ones for FRONT side.

System action: A form definition is generated, which assumes the number of copies that are specified for front side.

Operator response: Check the number of copies.

AKQ223E  LOGICAL PAGE POSITION FOR (page side) SIDE OF PAGE EXCEEDS THE LIMIT.

Explanation: The logical-page position that is specified by the OFFSET subcommand in a FORMDEF or COPYGROUP command exceeds the limit for the current side of the page.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct the positioning OFFSET parameter.

AKQ224E  MORE THAN 254 OVERLAYS ARE SPECIFIED IN A PAGEFORMAT.

Explanation: The maximum number of OVERLAY commands is 254. PPFA can issue this message for the OVERLAY subcommand of the PRINTLINE command.

System action: A page definition is not generated. The syntax check continues.

User response: Specify a valid number of OVERLAY commands.
AKQ225E  CONSTANT SUBCOMMAND PARAMETER (parameter) SPECIFIED IN SIMPLEX PROCESSING

Explanation: The BACK or BOTH parameter is specified for the CONSTANT subcommand within simplex processing.

System action: A form definition is not generated. The syntax check continues.

Operator response: Correct this CONSTANT subcommand or indicate DUPLEX.

AKQ226E  DIRECTION SUBCOMMAND ONLY ALLOWED WITH PRESENT SUBCOMMAND.

Explanation: The DIRECTION subcommand is specified, but the PRESENT subcommand is not.

System action: A form definition is not generated. The syntax check continues.

Operator response: Either add the PRESENT subcommand or remove the DIRECTION subcommand.

AKQ227E  THE ORIGIN OF THE RESOURCE (name) NAMED IN THE PRINTLINE COMMAND IS OFF THE LOGICAL PAGE.

Explanation: The relative position of the PRINTLINE overlay or segment that is named is off the logical page. The origin of the overlay or segment that is specified for the resource that is named in the N_UP subcommand is off the medium.

System action: The page definition that has the overlay or segment in question is not generated. PPF continues the syntax check, ignoring the problem.

Operator response: Correct the x-position and y-position for the OVERLAY or SEGMENT subcommand.

AKQ228E  THE ORIGIN OF THE OVERLAY (overlay name) NAMED IN THE (command) COMMAND IS OFF THE MEDIUM

Explanation: The resource position values position the resource such that at least part of the resource is off the medium (physical page).

System action: The form definition that has the overlay in question is not generated. PPF continues the syntax check, ignoring the problem.

User response: Correct the relative x-position and relative y-position values for the OVERLAY named in the N_UP subcommand.

AKQ229W  SUBGROUPS FOR FRONT AND BACK OF SAME SHEET USED DIFFERENT BINS.

Explanation: In your subgroup command, you specified FRONT and BACK parameters. However, your COPYGROUP has different bins that are specified.

System action: A form definition is generated that specifies the bin that is used for the front side.

Operator response: Check the number of copies and correct the bin setting.

AKQ231E  PRINTLINE OR LAYOUT IS NOT SPECIFIED.

Explanation: No PRINTLINE or LAYOUT command exists in the page format.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify either a PRINTLINE or LAYOUT command.

AKQ232E  REQUIRED SUBCOMMAND TEXT OR LENGTH IS NOT SPECIFIED.

Explanation: A FIELD subcommand must have a TEXT or LENGTH subcommand.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify either a TEXT subcommand or a LENGTH subcommand.

AKQ233E  THE LOGICAL PAGE SIZE IS TOO LARGE OR TOO SMALL.

Explanation: The specified page size is too large or too small. The page size must be 1 - 32767 pels. The HEIGHT and WIDTH subcommands must be 1 - 32767 pels or the same measurements expressed in other units.

System action: A page definition is not generated. The syntax check continues, assuming the defaults.

Operator response: Correct the error.

AKQ234E  POSITION OF LINEONE EXCEEDS THE LOGICAL PAGE BOUNDARY.

Explanation: The TOP or MARGIN position that is specified by the LINEONE subcommand exceeds the logical page boundary. This error message is issued only if TOP or MARGIN is specified.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify a valid position value.
AKQ235E  MORE THAN 127 SEGMENTS ARE SPECIFIED IN ONE PAGEFORMAT.

Explanation: More than 127 segments are specified in a single PAGEFORMAT command. PPF can issue this message for the SEGMENT subcommand of the PRINTLINE command.

System action: No page definition is generated. The syntax check continues.

User response: Correct the error.

AKQ238E  MORE THAN 127 FONTS ARE SPECIFIED IN ONE PAGEFORMAT.

Explanation: More than 127 fonts are specified in one PAGEFORMAT or the specified TRC number exceeds 126. PPF counts each use of a font in more than one direction or rotation as a separate font.

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ239E  PRINT POSITION EXCEEDS THE LOGICAL PAGE BOUNDARY.

Explanation: The print position that is specified by POSITION subcommand exceeds the logical page boundary.

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ240E  NUMBER OF PRINTLINES, FIELDS, AND CONDITIONS EXCEEDS 65,535 IN ONE PAGEFORMAT.

Explanation: The total number of PRINTLINES, FIELDS, and CONDITIONS exceeds 65,535 in one page format.

System action: A page definition is not generated. The syntax check continues.

Operator response: Reduce the number of PRINTLINES, FIELDS, or CONDITIONS in the page format.

AKQ241E  TOTAL LENGTH OF TEXT DATA EXCEEDS 65,534 BYTES.

Explanation: The total length of text can be up to 65,534 bytes.

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ242E  THE VALUE OF THE STARTING POSITION OF A RECORD IS TOO LARGE OR TOO SMALL.

Explanation: The START position of a record exceeds the maximum (65,535) or minimum (1) value.

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ243E  DBCS LENGTH IS NOT A MULTIPLE OF 2.

Explanation: The number of bytes of DBCS must be a multiple of two. Therefore, the value of the LENGTH parameter must be a multiple of two.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify a valid length or a valid DBCS.

AKQ244E  INVALID CODE IS SPECIFIED IN THE TEXT.

Explanation: SBCS text must be within code range X'00' to X'FE'. Valid double-byte character set (DBCS) codes are between X'41' and X'FE' for each byte. PPF checks this range. Code X'4040' (blank) is the only exception. For example, the following are valid DBCS codes: X'4040', X'4141', X'41FE', X'F4E1', X'F1FE'.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify a valid code.

AKQ245E  HEXADECIMAL TEXT IS INVALID.

Explanation: Hexadecimal text is specified in an invalid format. Hexadecimal text must have an even length parameter and be in hexadecimal notation ('0' to 'F').

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify valid hexadecimal text.

AKQ246E  NULL LITERAL IS SPECIFIED.

Explanation: The literal has no string.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify a valid literal.
AKQ247E  KANJI NUMBER TEXT IS INVALID.

Explanation: A Kanji number is specified in invalid format. Kanji number text must be a string of Kanji numbers that are delimited by commas. Each Kanji number must be a decimal number equal to a valid DBCS code, minus X'4000'.

System action: A page definition is not generated. The syntax check continues.

Operator response: Specify valid kanji numbers in a valid format.

AKQ248E  TEXT ATTRIBUTE CONFLICTS WITH FONT.

Explanation: SBCS font is specified for DBCS text (type G, K), or DBCS font is specified for SBCS text (type C).

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ249E  TEXT ATTRIBUTE CONFLICTS WITH TEXT TYPE.

Explanation: The literal type conflicts with text type. SBCS literal is specified as type G or X, and DBCS literal is specified as type C, X, or K.

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ250E  TRC NUMBER IS DUPLICATED.

Explanation: The specified TRC number is duplicated in one page format.

System action: A page definition is not generated. The syntax check continues.

Operator response: Correct the error.

AKQ251W  SPECIFIED LENGTH IS SHORTER THAN THE TEXT AND WAS TRUNCATED.

Explanation: The LENGTH parameter of the TEXT subcommand is shorter than the length of the specified literal, which is truncated to a specified length.

System action: The operation continues, truncating the literal.

Operator response: Check the truncation.

AKQ252E  TEXT IS NOT THE LENGTH SPECIFIED BY THE LENGTH SUBCOMMAND.

Explanation: The length of the comparison text in a WHEN or OTHERWISE subcommand of a CONDITION command is not equal to the length specified by the LENGTH subcommand of that CONDITION command.

System action: A page definition is not generated. The syntax check continues.

Operator response: Change the comparison text or the LENGTH parameter so that they match.

AKQ253E  TEXT IN THE ‘WHEN’ SUBCOMMAND IS TOO LONG.

Explanation: Constant text in a WHEN subcommand of a CONDITION command is too long to fit into an 8150-byte CCP structured field.

System action: A page definition or form definition is not generated. The syntax check continues.

Operator response: Shorten the field to 8000 bytes or fewer, and shorten the comparison text accordingly.

AKQ254E  (text type) LITERAL WAS EXPECTED BUT (text type) WAS FOUND.

Explanation: An SBCS literal occurs where a DBCS one was expected, or vice versa.

System action: A page definition or form definition is not generated. The syntax check continues.

Operator response: In a FIELD command, do not use a DBCS literal without specifying a DBCS font. In a CONDITION command, do not mix SBCS and DBCS literals in the comparison text of a single WHEN subcommand.

AKQ255E  INVOKE SPECIFIES A SIDE FOR WHICH NO PLACE SUBCOMMANDS PUT DATA.

Explanation: The N_UP PLACE subcommand contains an error that makes it incompatible with the value that is specified in the INVOKE subcommand. Either INVOKE BACK was specified, but PLACE n BACK was not specified, or INVOKE FRONT was specified, but PLACE n FRONT was not specified.

System action: No form definition is generated. Processing continues.

User response: Specify the same value (FRONT or BACK) for both the INVOKE and PLACE subcommands.
**AKQ256E**  INCORRECT NUMBER OF PLACE SUBCOMMANDS.

**Explanation:** The required number of PLACE subcommands must be specified.

**System action:** No form definition is generated. Processing continues.

**User response:** When you are using N_UP PLACE subcommands with single-sided printing, the number of PLACE subcommands must equal the value that is specified on N_UP. When you are using duplex printing, the number of PLACE subcommands must equal two times the value that is specified on N_UP.

---

**AKQ257W**  CONSTANT (parameter) FOUND WITH PLACE SUBCOMMAND.

**Explanation:** The CONSTANT (parameter) subcommand cannot be specified when N_UP PLACE subcommands are specified.

**System action:** A form definition is generated without constant forms control. The syntax check continues.

**User response:** Delete the CONSTANT (parameter) from the FORMDEF or COPYGROUP command.

---

**AKQ258W**  MORE THAN 122 OPERATION POSITIONS SPECIFIED FOR A FINISH OPERATION.

**Explanation:** More than 122 finishing operation positions are specified.

**System action:** A form definition is generated with 122 finishing positions. All others are ignored.

**User response:** Move extraneous operator position values.

---

**AKQ259W**  OPCOUNT AND OPPOS VALUES SPECIFIED. OPCOUNT IGNORED.

**Explanation:** Both OPCOUNT and OPPOS are specified.

**System action:** A form definition is not generated.

**Operator response:** If OPCOUNT is specified, OPPOS is ignored. When you are using OPPOS for controlling the position of each operation on the operation axis, OPCOUNT is ignored.

---

**AKQ260E**  (insert-1) not allowed with/on a (insert-2).

**Explanation:** This generic message indicates a contextually incorrect combination of PPF commands or subcommands. If this message indicates that an action is not allowed with or on a CMR of this type, the processing mode is not valid for that type of CMR. LINK is valid only for device link (DL) CMRs and device link (DL) CMRs can have only processing mode LINK. Any other use of LINK processing mode results in error message AKQ260E.

**System action:** A page definition or form definition is not generated.

**Operator response:** Correct the incorrect parameter and rerun the job.

---

**AKQ261E**  (insert-1) requires (insert-2).

**Explanation:** This generic message indicates that a PPF command or subcommand is missing.

**System action:** A page definition is not generated.

**Operator response:** Add the required parameter and rerun the job.

---

**AKQ262E**  (insert-1) specifies a (insert-2) which is not a (insert-3).

**Explanation:** This generic message indicates a contextually incorrect combination of PPF commands or subcommands. For example, an ENDGRAPHIC command is specified or defaulted to a GRAPHID that does not match a floating DRAWGRAPHIC BOX or DRAWGRAPHIC LINE.

**System action:** A page definition is not generated.

**Operator response:** Correct the indicated problem.

---

**AKQ263E**  (insert-1) exceeds (insert-2).

**Explanation:** This generic message indicates an out of bound condition for some parameters. For example, a DRAWGRAPHIC CIRCLE is positioned off the logical page.

**System action:** A page definition is not generated.

**Operator response:** Correct the indicated problem.

---

**AKQ264W**  (insert-1) is ignored (insert-2).

**Explanation:** This generic message indicates that a contextually incorrect combination of PPF commands or subcommands is clearly incorrect and is ignored. For example, if a LINEONE subcommand was coded on a Record Format PAGEDEF (for example, one using LAYOUT), the LINEONE subcommand is ignored.

**System action:** A page definition is generated.

**Operator response:** No action necessary unless the result is not what you wanted.

---

**AKQ265W**  (insert-1) exceeds (insert-2).

**Explanation:** This generic message indicates an out of bounds condition for some parameters, which is not necessarily critical. For example, a DRAWGRAPHIC CIRCLE is positioned outside the margin boundary but is still on the logical page.
**AKQ266E** PAGEDEF CONTAINS BOTH LAYOUT AND PRINTLINE COMMANDS.

**Explanation:** Lines are placed in a record format page definition by using LAYOUT commands or in an XML page definition by using XLAYOUT command; otherwise, lines are placed with PRINTLINE commands. The commands cannot be mixed in the same page definition.

**System action:** A page definition is not generated.

**Operator response:** Remove either the LAYOUT, XLAYOUT, or PRINTLINE commands.

---

**AKQ267E** MORE THAN ONE DEFAULT PAGEHEADER OR PAGETRAILER IN A PAGEFORMAT.

**Explanation:** Only one LAYOUT DEFAULT PAGEHEADER or PAGETRAILER can be coded in a PAGEFORMAT.

**System action:** A page definition is not generated.

**Operator response:** Remove one of the duplicates.

---

**AKQ268E** SPECIFIED MARGINS FOR THIS PAGEFORMAT OVERLAP.

**Explanation:** Either the left margin is defined on or right of the right margin or the top margin is defined on or below the bottom margin.

**System action:** A page definition is not generated.

**Operator response:** Redefine the margins so that they do not overlap.

---

**AKQ269E** A RECORD FORMAT PAGEDEF REQUIRES AT LEAST ONE FONT DEFINITION.

**Explanation:** At least one font must be defined whether one is referenced.

**System action:** A page definition is not generated.

**Operator response:** Define a font.

---

**AKQ270E** PDF417 MACRO DATA BYTE (insert-1), CODEPOINT (insert-2) CANNOT BE TRANSLATED TO GLI 0 ENCODATION

**Explanation:** All code points in an ASCII bar code must ultimately end up as ASCII. The printer translates EBCDIC code points if you tell it, but it translates both the macro data and the regular data.

---

**AKQ271E** THE FONT TYPE AND USER DATA TYPE (UDTYPE) SPECIFIED CAUSES A DATA TRANSFORMATION THAT IS NOT SUPPORTED

**Explanation:** When the User's Data Type (UDType) and font encoding are different, the printer must translate your data to the encoding type of the font. For example, if the UDT is UTF8 and the font is an ASCII font, then the printer must translate your data from UTF8 to ASCII. Therefore, combinations are restricted to these:

- **UDType of UTF8** Fonts can be only ASCII or UNICODE.
- **UDType of UTF16** Fonts can be only UNICODE.

**System action:** PAGEDEF is not generated.

**User response:** Chooses an appropriately encoded font.

---

**AKQ275I** (insert-1)

**Explanation:** This generic informational message contains variable text inserts. Two examples are:

1. **EXTERNAL OBJECT NAME IS DUPLICATED.**
2. **KEYWORD USED AS A NAME. CHECK NAME NOT OMITTED. PROCESSING CONTINUES.**

**System action:** Compilation continues. This definition is generated and stored or replaced.

**Operator response:** Make sure that the situation warned against is what you want. For example, make sure that the keyword that is used as a name is not a missing name, or that the duplicated object name is intended.

---

**AKQ2MMS** NUMBER OF MESSAGES EXCEEDS THE 270 ALLOWED LIMIT. PROGRAM TERMINATES.

**Explanation:** PPFA allows only 270 messages, including this message. When this limit is reached, the messages are printed and the program ends.

**System action:** The program ends.
Operator response: Correct the PPFA code for the messages that are issued and redo.

Explanation: This message accompanies the output listings of all form definitions and page definitions with the maximum return code for that particular object. The object is generated only when the return code is less than 8.

System action: None.

Operator response: None.

AKQ302I NO ERRORS FOUND IN (resource name) DEFINITION.

Explanation: One definition is processed. No statements were flagged in this definition.

System action: This definition is generated, and stored or replaced.

Operator response: None.

AKQ303S NO CONTROL STATEMENT(S) ARE SPECIFIED IN INPUT DATA.

Explanation: No control statements exist in the input data.

System action: The operation ends.

Operator response: Specify a valid PPFA command.

AKQ304S DEFINITION STATEMENT IS NOT SPECIFIED.

Explanation: No FORMDEF or PAGEDEF command exists in the system input command stream.

System action: The operation ends.

Operator response: Specify valid definition commands.

AKQ305S THIS DEFINITION IS NOT STORED BECAUSE MEMBER ALREADY EXISTS.

Explanation: This form definition or page definition is not saved because a file with the same name exists in the directory (REPLACE option is NO).

System action: A page definition or form definition is not generated. The syntax check continues to next definition.

Operator response: Check the specified form definition or page definition name, and specify REPLACE subcommand YES. Specify another form definition or page definition name.
AKQ350T  AN UNRECOVERABLE PROGRAM ERROR OCCURRED.
Explanation:  An error exists in PPFA logic.
System action:  The operation ends.
Operator response:  Use local problem-reporting procedures to report this message.

AKQ360E  FONT COMMAND DOES NOT CONTAIN SUFFICIENT INFORMATION.
Explanation:  The FONT command does not contain enough information to generate a valid MCF. This error is caused by having a CS parameter without a CP parameter, or vice versa.
System action:  A page definition is not generated.
Operator response:  Correct the referenced FONT command.

AKQ361E  FONT COMMAND SPECIFIES CONFLICTING PARAMETERS.
Explanation:  A FONT is specified in more than one way; however, only one of these parameters is allowed:
- Coded Font
- Character Set, Code Page pair (CS and CP parameters)
- GRID
System action:  A page definition is not generated.
Operator response:  Correct the referenced FONT command.

AKQ362E  FONT RATIO SPECIFIED WITHOUT FONT HEIGHT.
Explanation:  To scale a font, both the HEIGHT and RATIO must be specified. If a RATIO subcommand is found without a HEIGHT subcommand, the scaling information cannot be calculated by PPFA.
System action:  A page definition is not generated.
Operator response:  Correct the referenced FONT command.

AKQ363W  HEIGHT SPECIFIED, WIDTH IN GRID IGNORED.
Explanation:  You specified both a HEIGHT and GRID in the FONT command.
System action:  None.
Operator response:  Correct the referenced FONT command.

AKQ364E  INVALID DIRECTION WITH RELATIVE PRINTLINE
Explanation:  You specified an incorrect direction with the relative printline in your page definition source. The field direction must match the direction of the printline. The printline direction must be ACROSS.
System action:  A page definition is not generated.
Operator response:  Correct the referenced DIRECTION subcommand.

AKQ365W  COLOR AND EXTENDED COLOR SPECIFIED
Explanation:  Both COLOR and one of the extended color keywords (RGB, CMYK, HIGHLIGHT, CIELAB) was specified.
System action:  Both requests are placed into the output resource. Output depends on printer function.
Operator response:  If output does not print as expected, remove one of the specifications.

AKQ370E  BARCODE NAME WAS NOT PREVIOUSLY DEFINED.
Explanation:  You attempted to reference a bar code name that was not defined.
System action:  A page definition is not generated.
Operator response:  Correct the referenced BARCODE subcommand of the FIELD command.

AKQ371E  BARCODE NAME WAS PREVIOUSLY DEFINED.
Explanation:  You attempted to define a bar code name that was previously defined.
System action:  A page definition is not generated.
Operator response:  Correct the referenced BARCODE subcommand of the FIELD command.

AKQ372W  BARCODE MODIFICATION UNDEFINED FOR TYPE GIVEN.
Explanation:  You specified a modification for a bar code that is not defined for the type specified.
System action:  A page definition is generated as specified so that, as new bar code types and modifications are introduced, you can create page definitions for them. However, you received this warning because the specification might also be an error.
Operator response:  Correct the referenced BARCODE subcommand of the FIELD command, if appropriate.

AKQ373W  BARCODE TYPE IS UNDEFINED.

**Explanation:** You specified a bar code type that is not defined.

**System action:** A page definition is generated as specified so that, as new bar code types and modifications are introduced, you can create page definitions for them. However, you received this warning because this specification might also be an error.

**Operator response:** Correct the referenced BARCODE subcommand of the FIELD command, if appropriate.

---

AKQ374W  INVALID DATA LENGTH FOR SELECTED BARCODE TYPE AND MODIFICATION.

**Explanation:** You specified a data length for a defined bar code type and modification that is invalid for that combination of type and modification. When extra control characters are used (for example, in QRCode SOSI data), the data after translation might not exceed the limit.


**System action:** A page definition is generated as specified so that, as new bar code types and modifications are introduced, you can create page definitions for them. However, you received this warning because this specification might also be an error.

**Operator response:** Correct the referenced BARCODE subcommand of the FIELD command, if appropriate.

---

AKQ401E  EXEC PARAMETER IS INVALID.

**Explanation:** The program parameter specification is invalid.

**System action:** A page definition or form definition is not generated. The syntax check continues.

**Operator response:** Specify a valid program parameter.

---

AKQ402T  ERROR OCCURRED DURING ATTEMPT TO OBTAIN STORAGE

**Explanation:** These conditions generate this message:

1. Available size to hold the compiled data for the page definition and form definition is exceeded.
2. Insufficient available disk space on the file system to write the output of the compiler.
3. Limit of 269 user errors generated within a PPFA source file is exceeded.

**System action:** The operation ends.

**Operator response:**

---

AKQ403T  ERROR OCCURRED DURING ATTEMPT TO FREE STORAGE.

**Explanation:** A system error occurred while PPFA attempted to free disk space at the end of an execution.

**System action:** The operation ends.

**Operator response:** Use local problem-reporting procedures to report this message.

---

AKQ404T  SYSIPT OPEN FAILURE.

**Explanation:** SYSIPT cannot be opened.

**System action:** The operation ends.

**Operator response:** Assign a valid input data file.

---

AKQ405T  INSUFFICIENT STORAGE TO EXECUTE PPFA.

**Explanation:** The region size is too small to run PPFA.

**System action:** The operation ends.

**Operator response:** Increase the region size available to the job.

---

AKQ410T  (Librarian error message).

**Explanation:** The message describes a librarian error.

**System action:** The operation ends.

**Operator response:** Contact a system programmer.

---

AKQ411T  FORMDEF LIBRARY OPEN FAILURE.

**Explanation:** The FORMDEF library cannot be opened.

**System action:** The operation ends.

**Operator response:** Assign a valid FORMDEF library.

---

AKQ412T  FORMDEF LIBRARY I/O ERROR.

**Explanation:** An I/O error occurred during attempted access of a form definition directory.

**System action:** The operation ends.

**Operator response:** Check the permissions of the directory. If you do not have access, contact the owner of the directory. If this action does not resolve the problem, contact a system programmer.
<table>
<thead>
<tr>
<th>Code</th>
<th>Message Description</th>
<th>Explanation</th>
<th>System Action</th>
<th>Operator Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKQ413T</td>
<td>FORMDEF DIRECTORY CANNOT BE UPDATED.</td>
<td>The FORMDEF member cannot be registered on the directory.</td>
<td>The operation ends.</td>
<td>Contact a system programmer. See the documentation for your operating system.</td>
</tr>
<tr>
<td>AKQ414T</td>
<td>FORMDEF LIBRARY CLOSE FAILURE.</td>
<td>A form definition directory cannot be closed.</td>
<td>The operation ends.</td>
<td>Contact a system programmer.</td>
</tr>
<tr>
<td>AKQ415T</td>
<td>PAGEDEF LIBRARY OPEN FAILURE.</td>
<td>The PAGEDEF library cannot be opened.</td>
<td>The operation ends.</td>
<td>Use local problem-reporting procedures to report this message.</td>
</tr>
<tr>
<td>AKQ416T</td>
<td>PAGEDEF LIBRARY I/O ERROR.</td>
<td>I/O error occurs during attempted access of a page definition directory.</td>
<td>The operation ends.</td>
<td>Assign a valid PAGEDEF library.</td>
</tr>
<tr>
<td>AKQ417T</td>
<td>PAGEDEF DIRECTORY CANNOT BE UPDATED.</td>
<td>A page definition file cannot be registered on the directory.</td>
<td>The operation ends.</td>
<td>Assign a valid PAGEDEF library.</td>
</tr>
<tr>
<td>AKQ418T</td>
<td>PAGEDEF LIBRARY CLOSE FAILURE.</td>
<td>A page definition directory cannot be closed.</td>
<td>The operation ends.</td>
<td>Assign a valid PAGEDEF library.</td>
</tr>
<tr>
<td>AKQ420T</td>
<td>SYSTEM ERROR. ABEND CODE = (ABEND code).</td>
<td>System forces PPFA to end abnormally.</td>
<td>The operation ends.</td>
<td>Assign a valid PAGEDEF library.</td>
</tr>
<tr>
<td>AKQ421T</td>
<td>FORMDEF LIBRARY IS FULL.</td>
<td>The file system into which PPFA attempted to save the form definition is full.</td>
<td>The operation ends.</td>
<td>Increase the size of the file system or specify a directory on a file system that has more disk space.</td>
</tr>
<tr>
<td>AKQ422T</td>
<td>PAGEDEF LIBRARY IS FULL.</td>
<td>The file system into which PPFA attempted to save the page definition is full.</td>
<td>The operation ends.</td>
<td>Increase the size of the file system or specify a directory on a file system that has more disk space.</td>
</tr>
<tr>
<td>AKQ501T</td>
<td>SYSIN OPEN FAILURE.</td>
<td>The PPFA input source file cannot be opened.</td>
<td>The operation ends.</td>
<td>Specify a valid input source file.</td>
</tr>
<tr>
<td>AKQ502T</td>
<td>SPANNED RECORD OF SYSIN IS NOT SUPPORTED.</td>
<td>The spanned record of the PPFA input source file is not supported.</td>
<td>The operation ends.</td>
<td>Specify a valid input record format.</td>
</tr>
<tr>
<td>AKQ503T</td>
<td>UNDEFINED LENGTH RECORD OF SYSIN IS NOT SUPPORTED.</td>
<td>An undefined length record of PPFA input source file is not supported.</td>
<td>The operation ends.</td>
<td>Specify a valid input record format.</td>
</tr>
<tr>
<td>AKQ504T</td>
<td>LOGICAL RECORD LENGTH OF SYSIN EXCEEDS LIMIT.</td>
<td>The logical record length of the PPFA input source file exceeds the limit of 100 bytes or the z/OS variable length of 104.</td>
<td>The operation ends.</td>
<td>Specify a valid input record format.</td>
</tr>
</tbody>
</table>
Operator response: Correct the logical record length of the file.

AKQ510T FORMDEF/PAGEDEF LIBRARY OPEN FAILURE.
Explanation: The FORMDEF or PAGEDEF directory cannot be opened.
System action: The operation ends.
Operator response: Specify a valid FORMDEF or PAGEDEF or check to make sure that the directory is correct.

AKQ511T I/O ERROR OCCURRED DURING (FORMDEF/PAGEDEF) DIRECTORY SEARCH. RETURN CODE = (return code) REASON CODE = (reason code)
Explanation: I/O error occurred during the FIND function.
System action: The operation ends.
Operator response: Contact a system programmer.

AKQ512T LOGICAL RECORD LENGTH OF FORMDEF/PAGEDEF EXCEEDS LIMIT.
Explanation: The logical record length exceeds maximum or minimum value.
System action: The operation ends.
Operator response: Specify a file name that has a valid record length.

AKQ513T BLOCK SIZE OF FORMDEF/PAGEDEF EXCEEDS LIMIT.
Explanation: The block size exceeds maximum or minimum value.
System action: The operation ends.
Operator response: Assign a file name that has a valid block size.

AKQ514T UNDEFINED LENGTH RECORD IS NOT SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.
Explanation: An undefined length record is not supported in FORMDEF/PAGEDEF directory.
System action: The operation ends.
Operator response: Assign a valid record format.

AKQ515T FIXED LENGTH RECORD IS NOT SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.
Explanation: The fixed-length record is not supported in the FORMDEF or PAGEDEF library.
System action: The operation ends.
Operator response: Assign a valid record format.

AKQ516T NO CONTROL CHARACTER RECORD IS SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.
Explanation: No control character record is supported in FORMDEF/PAGEDEF directory.
System action: The operation ends.
Operator response: Assign a valid record format.

AKQ517T NO SPACE IN FORMDEF/PAGEDEF DIRECTORY.
Explanation: No space was available in the FORMDEF directory or the PAGEDEF directory to add or replace the resource.
System action: The operation ends.
Operator response: Increase the directory space or specify a directory on another file system that has more disk space.

AKQ518T I/O ERROR OCCURRED WHILE UPDATING FORMDEF/PAGEDEF DIRECTORY. RETURN CODE = (return code). REASON CODE = (reason code).
Explanation: A permanent I/O error was detected, the specified data control block is not opened, or insufficient disk space exists to do the write function.
System action: The operation ends.
Operator response: Contact a system programmer.

AKQ519T I/O ERROR OCCURRED DURING WRITE.
Explanation: The error message is displayed.
System action: The operation ends.
Operator response: Contact a system programmer.

AKQ520T SPANNED RECORD IS NOT SUPPORTED IN FORMDEF/PAGEDEF LIBRARY.
Explanation: The spanned record is not supported in the FORMDEF or PAGEDEF library.
System action: The operation ends.
Operator response: Remove the SPAN attribute and assign a valid data set.

AKQ522T BLOCK SIZE IS NOT SPECIFIED FOR FORMDEF/PAGEDEF DATA SET.
Explanation: A block size is not specified for FORMDEF/PAGEDEF data set.
System action: The operation ends.
Operator response: Specify a BLKSIZE in the DD statement.

AKQ540T SYSTEM ABEND (code) OCCURRED IN PPFA PROCESS.
Explanation: A system ABEND (code) occurred in PPFA. The ESTAE macro instruction ended processing.
System action: The operation ends.
Operator response: Contact a system programmer. See System Messages for your operating system.

AKQ541T USER ABEND (code) OCCURRED IN PPFA/OS/390 PROCESS.
Explanation: A user ABEND (code) occurred in PPFA. The ESTAE macro instruction ended processing.
System action: The operation ends.
Operator response: Use local problem-reporting procedures to report this message.

AKQ600T INPUT FILENAME NOT SPECIFIED.
Explanation: You did not specify an input file name.
System action: The operation ends.
Operator response: Enter the input file name.

AKQ601T INPUT FILETYPE NOT SPECIFIED.
Explanation: You did not specify an input file type.
System action: The operation ends.
Operator response: Enter the input file type.

AKQ602T COMMAND SYNTAX IS NOT VALID.
Explanation: The command syntax that you entered was not accepted.
System action: The operation ends.
Operator response: Enter a valid command.

AKQ603T FILEMODE FOR (FORMDEF/PAGEDEF/LISTING) IS INVALID.
Explanation: You entered an invalid file mode for FORMDEF, PAGEDEF, or LISTING.
System action: The operation ends.
Operator response: Enter a valid file extension.

AKQ604T INVALID PARAMETER IS SPECIFIED IN (FORMDEF/PAGEDEF/LISTING/SIZE) OPTION.
Explanation: You entered an invalid parameter for FORMDEF, PAGEDEF, LISTING, or SIZE.
System action: The operation ends.
Operator response: Enter a valid option parameter.

AKQ605T (FORMDEF/PAGEDEF/LISTING/SIZE) KEYWORD IS DUPLICATED.
Explanation: You entered a duplicate keyword for FORMDEF, PAGEDEF, LISTING, or SIZE.
System action: The operation ends.
Operator response: Enter a unique keyword.

AKQ606T FILETYPE FOR (FORMDEF/PAGEDEF/LISTING) NOT SPECIFIED.
Explanation: The file type for FORMDEF, PAGEDEF, or LISTING was not entered.
System action: The operation ends.
Operator response: Enter an appropriate file type.

AKQ607T INVALID KEYWORD SPECIFIED.
Explanation: The keyword that you entered was not accepted.
System action: The operation ends.
Operator response: Enter a valid keyword.

AKQ608T INVALID SIZE PARAMETER SPECIFIED.
Explanation: The size parameter that is specified is not valid.
System action: The operation ends.
Operator response: Enter a valid size parameter.

AKQ610T SIZE PARAMETER VALUE EXCEEDS THE ALLOWABLE MAXIMUM.
Explanation: The size that is entered exceeds the maximum allowable.
System action: The operation ends.
Operator response: Enter a valid size value.

AKQ61T SIZE PARAMETER VALUE IS TOO SMALL.
Explanation: The size that is entered is too small for running in PPFA/VM.
System action: The operation ends.
Operator response: Enter a valid size value.

AKQ612T INVALID FILE IDENTIFIER "*" SPECIFIED FOR INPUT FILE.
Explanation: "*" is specified for input file name or file type.
System action: The operation ends.
Operator response: Enter a valid file name or file type.

AKQ613T SIZE PARAMETER VALUE IS MISSING.
Explanation: You did not specify a size parameter
System action: The operation ends.
Operator response: Specify a valid size parameter.

AKQ620T INPUT FILE WAS NOT FOUND.
Explanation: The input file name entered was not found.
System action: The operation ends.
Operator response: Correct the input file name.

AKQ621T NO READ/WRITE (file mode) DISK ACCESSED FOR (INPUT/LISTING/FORMDEF/PAGEDEF/OUTPUT).
Explanation: The disk on which the file is saved cannot be read from or written to because it either was not accessed or was accessed with an invalid access mode.
System action: The operation ends.
Operator response: Access the file system by using a valid access mode.

AKQ622T INPUT FILE EXCEEDS THE ALLOWABLE LOGICAL RECORD LENGTH MAXIMUM.
Explanation: The logical record length of the input file exceeds the limit of 100 bytes or the z/OS variable record length of 104.
System action: The operation ends.
Operator response: Correct the logical record length of the file.

AKQ624T I/O ERROR OCCURRED IN (AKQINIO/AKQLBIO/AKQPRIO) MODULE. RC = (return code from FWRITE/FGETS macro instruction).
Explanation: An I/O error occurred during either FGETS or FWRITE processing of module AKQINIO, AKQLBIO, or AKQPRIO.
System action: The operation ends.
Operator response: Contact your system programmer.

AKQ625T DISK (file mode) IS FULL.
Explanation: Not enough space is available on the specified file system to write the file.
System action: The operation ends.
Operator response: Erase some files from the specified file disk and rerun.

AKQ639T ABEND EXIT ROUTINE FAILED TO EXECUTE. RC = (return code from ABNEXIT macro)
Explanation: ABEND exit routine cannot be established.
System action: The operation ends.
Operator response: Contact your system programmer.

AKQ640T SYSTEM ABEND (code) OCCURRED IN PPFA/VM PROCESS.
Explanation: A system ABEND occurred during processing. The ABEND exit routine ended processing.
System action: The operation ends.
Operator response: Use local problem-reporting procedures to report this message.

AKQ641T USER ABEND (code) OCCURRED IN PPFA/VM PROCESS.
Explanation: A user-initiated ABEND occurred during processing. The ABEND exit routine ended the processing.
System action: The operation ends.
Operator response: Use local problem-reporting procedures to report this message.
AKQ700I  SIZE PARAMETER IS NO LONGER NECESSARY IN PPF/370.

Explanation: The storage that is required to contain the messages and control blocks is not automatically set at 32K and 128K. If the control block storage is used up, an additional 128K is obtained and chained to the previous storage. All storage necessary to do the compile is obtained during processing.

System action: The compile process continues.

Operator response: None.
Appendix J. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the “Contact us” web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- z/OS TSO/E Primer
- z/OS TSO/E User’s Guide
- z/OS V2R2 ISPF User’s Guide Vol I

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out
punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* \ FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the
default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

* indicates an optional syntax element that is repeatable
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:
1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included
The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.
Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 1985, 2016
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted.
for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

**Minimum supported hardware**

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS](http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

**Trademarks**

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at [Copyright and Trademark information](http://www.ibm.com/legal/copytrade.shtml).

Adobe and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.
Glossary

This glossary defines technical terms and abbreviations that are used in PSF for z/OS documentation. If you do not find the term that you are looking for, see IBM Glossary of Computing Terms (http://www.ibm.com/software/globalization/terminology/).

These cross-references are used in this glossary:

**Contrast with.**
Refers to a term that has an opposed or substantively different meaning.

**See.**
Refers to preferred synonyms or to defined terms for acronyms and abbreviations.

**See also.**
Refers to related terms that have similar, but not synonymous, meanings, or to contrasted terms that have opposite or substantively different meanings.

**Synonym for.**
Indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the glossary.

**Synonymous with.**
Indicates a backward reference from a defined term to all other terms that have the same meaning.

**A**

**ACIF** See AFP conversion and indexing facility.

**Advanced Function Presentation**
A set of licensed programs, together with user applications, that use the all-points-addressable concept to print data on a wide variety of printers or to display data on various display devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information.

**AFP** See Advanced Function Presentation.

**AFP Conversion and Indexing Facility (ACIF)**
A print server utility program that converts a print file into AFP, MO:DCA-P, creates an index file for input data, and collects resources that are used by an AFP document into a separate file.

**all-points addressability (APA)**
The capability to address, reference, and position data elements at any addressable position in a presentation space or on a physical medium. An example of all points addressability is the positioning of text, graphics, and images at any addressable point on the physical medium. See also picture element.

**all-points-addressable mode**
Synonym for page mode.

**alphanumeric string**
A sequence of characters that consist solely of the letters a - z and the numerals 0 - 9.

**American National Standards Institute (ANSI)**
An organization that consists of producers, consumers, and general interest groups. ANSI establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States. It is the United States constituent body of the International Organization for Standardization (ISO).

**ANSI** See American National Standards Institute.

**APA** See all-points addressability.

**application**
The use to which an information system is put.

A collection of software components that are used to do specific types of work on a computer.

**application program**
A program that is written for or by a user that applies to the user’s work.

**ascender**
The parts of certain lowercase letters, such as b, d, or f, which at zero-degree character rotation rise above the top edge of other lowercase letters such as a, c, and e. Contrast with descender.
aspect ratio
The ratio of the length (or height) of a bar to the length (or width) of the linear bar code symbol.

attribute
A property or characteristic of one or more constructs. For example, character attribute, color attribute, current drawing attributes, default drawing attributes, line attributes, marker attributes, and pattern attributes.

B
bar
In bar codes, the darker element of a printed bar code symbol.

bar code
An array of parallel rectangular bars and spaces that together represent data elements or characters of a particular type. The bars and spaces are arranged in a predetermined pattern that follows unambiguous rules that are defined by the symbology.

bar code command set
In the [IPDS] architecture, a collection of commands used to present bar code symbols in a page, page segment, or overlay.

bar code density
The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is 3-10 cpi.

bar code object area
The rectangular area on a logical page in which a bar code presentation space is mapped.

Bar Code Object Content Architecture (BCOCA)
An architected collection of constructs that are used to interchange and present bar code data.

bar code symbol
A combination of characters that form a complete, scannable entity, including start and stop characters, quiet zones, data characters, and check characters that are required by a particular bar code type.

bar code symbology
A bar code language. Bar code symbologies are defined and controlled by various industry groups and standards organizations. Bar code symbologies are described in public domain bar code specification documents. Synonymous with [symbology]. Examples of bar code symbology include: Canadian Grocery Product Code (CGPC), European Article Numbering (EAN), Japanese Article Numbering (JAN), and Universal Product Code (UPC).

bar height
In bar codes, the bar dimension perpendicular to the bar width. Synonymous with [bar length] and [height].

bar length
In bar codes, the bar dimension perpendicular to the bar width. Synonymous with [bar height] and [height].

bar width
In bar codes, the thickness of a bar measured from the edge closest to the symbol start character to the trailing edge of the same bar.

baseline
A conceptual line in regard to which successive characters are aligned.

baseline direction
The direction in which successive lines of text appear on a logical page.

bearer bars
A bar across the top and bottom edge of a linear bar code. Partial scans of bar code symbologies, such as Interleaved 2 of 5, can produce valid, but incorrect, reads by self-discriminating scanners. Bearer bars help prevent such errors and increase reliability.

Note: A self-discriminating scanner is one that automatically determines which bar code symbology it is reading.

BCOCA
See Bar Code Object Content Architecture.

bin
The standard-size paper source on cut-sheet page printers that have more than one paper source. Each printer is set up with either A4 or letter-size paper as the standard size. Contrast with [cassette].

BITS A data type for architecture syntax, indicating one or more bytes to be interpreted as bit string information.
body  On a printed page, the area between the top and bottom margins that can contain data.
In a book, the portion between the front matter and the back matter.

boldface  A heavy-faced type.
Printing in heavy-faced type.

C

carriage control character  If present, the first character of an output record (line) that is to be printed or spaced; it determines how many lines are skipped before the line. See also control character.

cassette  A removable storage device that is the source for alternative sizes of paper on page printers that have more than one paper source. Contrast with bin.


CGPC  Canadian Grocery Product Code.

CHAR  A data type for architecture syntax, indicating one or more bytes to be interpreted as character information.

character  A member of a set of elements that are used for the organization, control, or representation of data. A character can be either a graphic character or a control character. See also control character and graphic character.
In bar codes, a single group of bars and spaces that represent an individual number, letter, punctuation mark, or other symbol.

character ascender  See ascender.

character attribute  A characteristic that controls the appearance of a character or character string.

character baseline  A conceptual reference line that is coincident with the X axis of the character coordinate system.

character code  An element of a code page or a cell in a code table to which a character can be assigned. The element is associated with a binary value. The assignment of a character to an element of a code page determines the binary value that is used to represent each occurrence of the character in a character string.

character descender  See descender.

character identifier  The unique name for a graphic character. See also graphic character identifier.

character rotation  The alignment of a character in regard to its character baseline, which is measured in degrees in a clockwise direction. Examples are 0°, 90°, 180°, and 270°. Zero-degree character rotation exists when a character is in its customary alignment with the baseline. Character rotation and font inline sequence are related in that character rotation is a clockwise rotation; font inline sequence is a counterclockwise rotation. See also orientation and rotation.

character set  A finite set of different graphic or control characters that is complete for a specified purpose. For example, the character set in ISO Standard 646, 7-bit Coded Character Set for Information Interchange. See also control character and font character set.

character set attribute  An attribute that is used to specify a coded font.

check character  The result of some mathematical combination of the characters in the field that is bar coded. Used as a check of the accuracy of both the input of the data field and the scanning of the bar code. A bar code can have 0, 1, 2, or sometimes more check characters.
check digit
Same as a check character, but limited to decimal digits only.

Codabar
A bar code symbology that is characterized by a discrete, self-checking, numeric code with each character represented by a stand-alone group or four bars and three spaces between them.

Code128
A bar code symbology that is characterized by a variable-length, alphanumeric code with 128 characters.

Code39
A bar code symbology that is characterized by a variable-length, bidirectional, discrete, self-checking, alphanumeric code. Three of the nine elements are wide and six are narrow. It is the standard for LOGMARS (the Department of Defense) and the AIAG.

coded font
A resource that contains elements of a code page and a font character set, and is used for presenting text, graphics character strings, and bar code HRI. See also code page and font character set.

In FOCA, a resource that contains the resource names of a valid pair of font character set and code page resources. The graphic character set of the font character set must match the graphic character set of the code page for the coded font resource pair to be valid.

In the IPDS architecture, a raster font resource that contains code points that are directly paired to font metrics and the raster representation of character shapes, for a specific graphic character set.

In the IPDS architecture, a font resource that contains descriptive information, a code page, font metrics, and a digital-technology representation of character shapes for a specific graphic character set.

coded graphic character set global identifier (CGCSGID)
A four-byte binary or a ten-digit decimal identifier that consists of the concatenation of a CGCSGID and a CPGID. The CGCSGID identifies the code point assignments in the code page for a specific graphic character set, from among all the graphic characters that are assigned in the code page.

code page
A resource object that contains descriptive information, graphic character identifiers, and code points corresponding to a coded graphic character set. Graphic characters can be added over time; therefore, to specifically identify a code page, both a CGCSGID and a CPGID is used.

A set of assignments, each of which assigns a code point to a character. Each code page has a unique name or identifier. Within a specific code page, a code point is assigned to one character. More than one character set can be assigned code points from the same code page. See also coded font.

code page global identifier (CPGID)
A unique 5-digit decimal or 2-byte binary identifier that is assigned to a code page. The range of values is 00001 - 65534 (X'0001' - X'FFE').

code point
A unique bit pattern that can serve as an element of a code page or a site in a code table, to which a character can be assigned. The element is associated with a binary value. The assignment of a character to an element of a code page determines the binary value that is used to represent each occurrence of the character in a character string. Code points are one or more bytes long.

color attribute
An attribute that affects the color values that are provided in a graphics primitive, a text control sequence, or an IPDS command. Examples of color attributes are foreground color and background color.

color model
The method by which a color is specified. For example, the RGB color space specifies color in terms of three intensities for red (R), green (G), and blue (B).

command
A request from a terminal or automated operator for the performance of an operation or service, or a request in a batch-processing job or print file for the
operation or execution of a particular program. In Page Printer Formatting Aid, commands are control statements for major formatting functions. For example, FORMDEF and COPYGROUP are commands. Commands are further specified by subcommands and parameters.

**command stream**
The sequence of Page Printer Formatting Aid commands that is submitted with the job control statements in a Page Printer Formatting Aid execution. The commands and subcommands are the control statements that define the object or objects to be generated.

**compatibility mode**
The use of table reference characters (TRCs) that are acceptable to line printers and page printers and that access page definitions with little or no change to the user's data or to the job command stream. Contrast with page mode.

**composed-text data file**
A file that contains text data and text control information that dictates the format, placement, and appearance of the data to be printed.

**conditional processing**
A page definition function that allows input data records to partially control their own formatting.

**construct**
An architected set of data such as a structured field or a triplet.

**continuous code**
A linear bar code in which each character starts immediately after the preceding character. No space or gap exists between characters. Interleaved 2 of 5 is an example of a continuous bar code.

**control character**
A character that denotes the start, modification, or end of a control function. A control character can be recorded for use in a subsequent action, and it can have a graphic representation. See also character.

A control function the coded representation of which consists of a single code point.

**copy group**
A subset of a form definition that contains a set of controls for the physical pages of a printout. Such functions as the selection of either of two paper sources on the page printer, the use of duplex printing, or the positioning of the reference point for all printing on the sheet are available in the copy group.

**cm**
Centimeters.

**CMS**
Conversational Monitor System.

**cpi**
Characters per inch.

**cut-sheet media**
Unconnected sheets.

**D**

**data map**
An internal object whose structured fields control the formatting of data on a logical page of a printout. Created by a PAGEDEF command or a PAGEFORMAT command.

**data stream**
A continuous stream of data that is a defined format. An example of a defined format is a structured field.

**DBCS**
Double-byte character set.

**default**
Pertaining to an attribute, value, or option that is assumed when none is explicitly specified and one is needed to continue processing.

**density**
A measure of the number of characters per inch or per millimeter represented by the bar code. A high-density bar code represents more characters per inch than a low-density bar code.

The bar code symbology helps determine the density. Within a specific symbology, factors that affect the density of a bar code are: the x-dimension (width of the narrow bar) and the wide-to-narrow ratio (width of a wide bar relative to the narrow bar).

**descender**
In a font, the distance from the baseline to the bottom of the character box. This value might differ for different characters in a specific font. Contrast with ascender.
direction
The print position of data in a logical page, line, or field. In Page Printer Formatting Aid, the ultimate reference point for all direction controls on a page is the hardware origin. Secondary and tertiary reference points are possible as well, allowing more than one print direction on a page.

discrete code
A linear bar code that is constructed with groups of bars and spaces that represent individual characters and having a space or intercharacter gap between each group. This gap is used solely to separate characters and contains no data. Code 3 of 9 is a discrete bar code.

document
A machine-readable collection of one or more objects that represents a composition, a work, or a collection of data.

A publication or other written material.

double-byte character set (DBCS)
A character set, such as a set of Japanese ideographs, requiring 2 bytes to identify each character.

duplex printing
Printing on both sides of a sheet.

y-coordinate
The vertical or baseline position that defines a page origin or the starting point of a line or field.

E
EAN  See European Article Numbering
EAN2SUP
A parameter that specifies a bar code type of European Article Numbering, Two-digit Supplemental.

EAN5SUB
A parameter that specifies a bar code type of European Article Numbering, Five-digit Supplemental.

EAN8  A parameter that specifies a bar code type of European Article Numbering 8 (includes Japanese Article Numbering-short).

EAN13  A parameter that specifies a bar code type of European Article Numbering 13 (includes Japanese Article Numbering-standard).

EBCDIC  See Extended Binary Coded Decimal Interchange Code

electronic overlay
In IBM Print Server Facility, a collection of constant data that is electronically composed in the host processor and can be merged with variable data on a sheet during printing. Contrast with page segment. See also overlay and preprinted form.

European Article Numbering (EAN)
The bar code symbology that is used to code grocery items in Europe.

Extended Binary Coded Decimal Interchange Code (EBCDIC)
A coded character set that consists of 8-bit coded characters.

external library resource (member)
Objects that can be used by other licensed programs while printing jobs; for example, coded fonts, code pages, font character sets, form definitions, page definitions, and page segments. Synonym for resource object.

external object
Synonym for resource object.

F

FCB  Forms control buffer.

field
In a record, a specified area that is used for a particular class of data; for example, a group of character positions that are used to enter or display wage rates on a screen.

In Page Printer Formatting Aid, any area of a record singled out for particular formatting treatment.

field processing
Mapping individual fields to a page of output with special formatting controls.

file
A named set of records that are stored or processed as a unit.

first read rate
The percentage of the bar code scans that read correctly on the first scan of the bar code. A 99% or higher first read is
desirable. Anything below 85% is normally not acceptable.

**fixed medium information**
Information that can be applied to a sheet by a printer or printer-attached device that is independent of data that is provided through the data stream. Fixed medium information does not mix with the data that is provided by the data stream and is presented on a sheet either before or after the text, image, graphics, or bar code data provided within the data stream. Fixed medium information can be used to create preprinted forms, or other types of printing, such as colored logos or letterheads, that cannot be created conveniently within the data stream.

**FOCA**
See [Font Object Content Architecture](#).

**font**
A family or assortment of characters of a specific size and style; for example, 9-point Bodoni Modern.

**font character set**
A FOCA resource that contains descriptive information, font metrics, and the digital representation of character shapes for a specified graphic character set.

**Font Object Content Architecture (FOCA)**
An architected collection of constructs that are used to describe fonts and to interchange those font descriptions.

**font typeface global identifier (FGID)**
See [global resource identifier (GRID)](#).

**form**
A physical piece of paper or other medium on which output data is printed. For cut-sheet printers, a form is one sheet of paper or other medium. For continuous-forms printers, the form is the area of paper (or other medium) defined to the printer as a single physical page, which for fan-fold paper is normally the area between perforations. See also medium, page, and sheet.

**format**
The arrangement or layout of data on a physical medium or in a presentation space.

**formatted data**
In FD:OCA, data whose implied syntax and semantics are represented by architected controls that accompany the data.

**formatted data object (FDO)**
An object that contains formatted data. See also object.

**Formatted Data Object Content Architecture (FD:OCA)**
An architected collection of constructs that are used to interchange formatted data.

**formatter**
A process that is used to prepare a document for presentation.

**formdef**
See [form definition](#).

**form definition**
In IBM Print Server Facility, a resource object that defines the characteristics of the form, which include: overlays to be used, text suppression, position of page data on the form, and modifications and number of copies of a page.

**forms control buffer (FCB)**
A line printer control. In the 3800 Printing Subsystem, a buffer for controlling the vertical format of printed output.

**forms flash**
In the 3800 Printing Subsystem, the function of the printer that allows user-prepared images to be printed with variable page data. An operator must insert the image holder when forms overlay printing is wanted.

The photographic negative of a predefined design to be exposed to the photoconductor by a flash of light. The forms overlay can be merged with variable data during printing. See also electronic overlay.

**G**

**GGCID**
See [graphic character global identifier](#).

**GCSGID**
See [graphic character set global identifier](#).

**GID**
See [global identifier](#).

**global identifier (GID)**
Any of these identifiers:
- Coded character set identifier (CCSID).
• Coded graphic character set global identifier (CGCSGID)
• Code page global identifier (CPGID)
• Font typeface global identifier (FGID)
• Global resource identifier (GRID)
• Graphic character global identifier (GCGID)
• Graphic character set global identifier (GCSGID)
• Object identifier (OID)
• In MO:DCA, an encoded graphic character string that provides a reference name for a document element.

global resource identifier (GRID)
An 8-byte identifier that identifies a coded font resource. A GRID contains the following fields in the order shown:
1. GCSGID of a minimum set of graphic characters that are required for presentation. It can be a character set that is associated with the code page, or with the font character set, or with both.
2. CPGID of the associated code page
3. FGID of the associated font character set
4. Font width in 1440ths of an inch.

GOCA
See Graphics Object Content Architecture (GOCA)

graphic character
A member of a set of symbols that represent data. Graphic characters can be letters, digits, punctuation marks, or other symbols. Synonymous with glyph. See also character.

graphic character global identifier (GCGID)
An alphanumeric character string that is used to identify a specific graphic character. A GCGID can be 4 - 8 bytes long.

graphic character identifier
The unique name for a graphic character in a font or in a graphic character set. See also character identifier.

graphic character set global identifier (GCSCGID)
A unique graphic character set identifier that can be expressed as either a 2-byte binary or a five-digit decimal value.

graphics command set
In the IPDS architecture, a collection of commands that are used to present GOCA data in a page, page segment, or overlay.

graphics object
An object that contains graphics data. See also object.

graphics object area
A rectangular area on a logical page into which a graphics presentation space window is mapped.

Graphics Object Content Architecture (GOCA)
An architected collection of constructs that are used to interchange and present graphics data.

GRID See global resource identifier.

guard bars
The bars at both ends and the center of an EAN, JAN, or UPC symbol, that provide reference points for scanning.

H

height
In Page Printer Formatting Aid, refers to the vertical dimension of a logical page and is controlled by the HEIGHT subcommand.

In bar codes, the bar dimension perpendicular to the bar width. Synonymous with bar height and bar length.

hexadecimal
A number system with a base of sixteen. The decimal digits 0 - 9 and characters A through F are used to represent hexadecimal digits. The hexadecimal digits A-F correspond to the decimal numbers 10 - 15. An example of a hexadecimal number is X’1B’, which is equal to the decimal number 27.

highlighting
The emphasis of displayed or printed information. Examples are increased intensity of selected characters on a display screen and exception highlighting on an IPDS printer.

host
In the IPDS architecture, a computer that drives a printer.

In IOCA the host is the controlling environment.
HRI  See human-readable interpretation

human-readable interpretation (HRI)
The printed translation of bar code characters into equivalent Latin alphabetic characters, Arabic numeral decimal digits, and common special characters that are typically used for printed human communication.

I

image  An electronic representation of a picture that is produced by using sensing light, sound, electron radiation, or other emanations that come from the picture or reflected by the picture. An image can also be generated directly by software without reference to an existing picture.

image content  Image data and its associated image data parameters.

Image Object Content Architecture (IOCA)
An architected collection of constructs used to interchange and present images.

in  Inches.

IND2OF5  A parameter that specifies a bar code type of Industrial 2-of-5.

inline  In printing, the direction of successive characters in a line of text. Synonymous with inline direction.

inline direction  Synonym for inline.

Intelligent Printer Data Stream (IPDS)
An architected host-to-printer data stream that contains both data and controls that define how the data is to be presented.

intercharacter gap  The space between characters in a discrete bar code symbology.

International Organization for Standardization (ISO)
An organization of national standards bodies from various countries that are established to promote development of standards to facilitate international exchange of goods and services, and develop cooperation in intellectual, scientific, technological, and economic activity.

Invoke Data Map
A control record that is placed in the user's data to begin a new page format.

Invoke Medium Map
A control record that is placed in the user's data to begin a new copy group.

IOCA  See Image Object Content Architecture

IPDS  See Intelligent Printer Data Stream

ISO  See International Organization for Standardization

italics  A typeface with characters that slant upward to the right. In FOCA, italics is the common name for the defined inclined typeface posture attribute or parameter.

ITL2OF5  A parameter that specifies a bar code type of Interleaved 2-of-5, Automatic Identification Manufacturers Uniform Symbol Specification-I 2/5.

J

JAN  See Japanese Article Numbering

Japanese Article Numbering (JAN)
The bar code symbology that is used to code grocery items in Japan.

jog  Offset stacking of individual sheets or sets of sheets in the output hopper of a page printer or copy mark in a continuous forms printer.

K

kanji  A graphic character set consisting of symbols that are used in Japanese ideographic alphabets. Each character is represented by 2 bytes.

keyword  A two-part self-defining parameter that consists of a 1-byte identifier and a 1-byte value.

L

ladder orientation  Linear bar code orientation where the bars are parallel to the base of the document (like the rungs of a ladder). Sometimes called vertical orientation (because that is the direction of the scan).
landscape presentation
The position of a printed sheet that has its long edges at the top and bottom and its short edges at the sides. Contrast with portrait presentation.

language
A set of symbols, conventions, and rules that is used for conveying information.

leading
A printer’s term for the distance between lines of type that is measured in points. It refers to the lead slug placed between lines of type in traditional typesetting.

library
System storage for generated form definitions and page definitions.

library resource (member)
A named collection of records or statements in a library.

library resource name
A name by which an object can be called from a library by IBM Print Server Facility as part of a print job. Includes the two-character prefix for the type of object, such as P1 for page definitions, F1 for form definitions, or O1 for overlays (also known as resource name).

line attributes
Those attributes that pertain to straight and curved lines. Examples of line attributes are line type and line width.

line data files
Files that are formatted for printing on line printers.

line printer
A device that prints a line of characters as a unit. Synonymous with line-at-a-time printer. Contrast with page printer.

line type
A line attribute that controls the appearance of a line. Examples of line types are dashed, dotted, and solid. Contrast with line width.

line width
A line attribute that controls the appearance of a line. Examples of line width are light, medium, and bold. Contrast with line type.

lines per inch (lpi)
On a printer, a measurement of the number of lines per vertical inch of paper. A unit of measure for specifying the baseline increment.

local name
A name for a suppression, an overlay, or a font that is used only within the Page Printer Formatting Aid command stream. Contrast with user-access name.

location
A site within a data stream. A location is specified in terms of an offset in the number of structured fields from the beginning of a data stream, or in the number of bytes from another location within the data stream.

logical page
The area on a surface of a form that is formatted for printing. A collection of data that can be printed on one side of a sheet of paper. See also form and page.

logical page origin
The user-defined point that acts as a reference for all positioning of printed material on the page. The point nearest the hardware origin where printing can occur.

logical unit (L-unit)
A unit of linear measurement that is expressed with a unit base and units per unit-base value. For example, in Page Printer Formatting Aid, 1 logical unit = 1/240 inch (unit base = 10 inches, units per unit base = 2400).

lpi Lines per inch.

lowercase
Pertaining to small letters as distinguished from capital letters. Examples of small letters are a, b, and g. Contrast with uppercase.

L-unit A unit of linear measurement that is expressed with a unit base and units per unit-base value. In other words, the number of units in a linear inch. Synonymous with logical unit.
M

MAT2OF5
A parameter that specifies a bar code type of Matrix 2-of-5.

media origin
The first hardware addressable point on the physical page. The point from which the logical page origin is positioned by the medium map.

medium
The physical material (for example, paper) on which data is printed. See also form.

medium map
An internal object whose structured fields control the physical sheets of a printout, including the choice of duplex printing, the beginning print position, and the paper source to use. Controlled by a COPYGROUP command in a Page Printer Formatting Aid command stream.

medium overlay
Synonym for overlay.

mixed data files
Files consisting of composed and uncomposed portions.

mm
Millimeters.

MOD
A parameter that specifies more processing information about the bar code symbol to be generated. See Bar Code Object Content Architecture Reference for more information.

Mixed Object Document Content Architecture (MO:DCA)
An architected, device-independent data stream for interchanging documents.
Print data that is composed into pages. Text formatting programs can produce composed text data that consists entirely of structured fields.

MO:DCA
See Mixed Object Document Content Architecture.

MO:DCA-P
Mixed Object Document Content Architecture for Presentation.

module
In a bar code symbology, the nominal width of the smallest element of a bar or space. Actual bar code symbology bars and spaces can be a single module wide or some multiple of the module width. The multiple need not be an integer.

MODWIDTH
A parameter that specifies the width of the smallest defined bar code element by using mils (thousandths of an inch).

MSI
A parameter that specifies a bar code type of modified Plessey code.

multiple up
The printing of more than one page on a single side of a sheet of paper.

N
name
A table heading for architecture syntax. The entries under this heading are short names that give a general indication of the contents of the construct.

noncompatibility mode
The use of table reference character (TRC) numbers not compatible with a line printer.

normal duplex printing
Duplex printing for sheets that are to be bound on the long edge of the paper, regardless of whether the printing is portrait or landscape. Contrast with tumble duplex printing.

N_UP
The printing of more than one logical page on a single side of a medium.

O
object
A collection of data that is referred to by a single name. Form definitions and page definitions that are stored in a library are resources.

offset
A table heading for architecture syntax. The entries under this heading indicate the numeric displacement into a construct. The offset is measured in bytes and starts with byte zero. Individual bits can be expressed as displacements within bytes.

order
In GOCA, a graphic construct that the controlling environment builds to instruct a drawing processor about what to draw and how to draw it.

orientation
The angular distance a presentation space
or object area is rotated in a specified coordinate system, expressed in degrees and minutes. For example, the orientation of printing on a physical medium, relative to the $X_m$ axis of the $X_m,Y_m$ coordinate system.

**origin** A picture element (pel).

**outline font**
A shape technology in which the graphic character shapes are represented in digital form by a series of mathematical expressions that define the outer edges of the strokes. The resultant graphic character shapes can be either solid or hollow.

**overlay**
A collection of predefined data such as lines, shading, text, boxes, bar codes, or logos, that can be merged with variable data on a page during printing. See electronic overlay.

**Overlay Generation Language (OGL)**
A programming language that is used to produce electronic overlays.

**P**

**page** A collection of data that can be printed on one side of a sheet of paper or a form.

The boundary for determining the limits of printing. See also logical page and physical page.

**page definition**
A resource that contains a set of Page Printer Formatting Aid formatting controls for printing pages of data. Includes controls for number of lines per printed sheet, font selection, print direction, and mapping of individual fields in the data to positions on the printed sheets.

**page ejection**
The point at which the printer finishes printing on one sheet and moves to the beginning of the next sheet.

**page format**
A subset of a page definition, containing all the same controls for formatting printed output as a page definition. Includes controls for number of lines per printed sheet, font selection, print direction, and mapping of individual fields in the data to positions on the printed sheets.

**page mode**
The mode of operation in which an AFP printer can accept a page of data from a host processor to be printed on an all-points-addressable output medium. Printed data can consist of pages that are composed of text, images, overlays, and page segments. Contrast with compatibility mode.

**page printer**
A device that prints a page at a time. Contrast with line printer.

**Page Printer Formatting Aid (PPFA)**
An IBM licensed program that creates and stores form definitions and page definitions, which are resource objects for print-job management. By writing a command stream that specifies form definitions, page definitions, or both, for running Page Printer Formatting Aid, you can store the objects that are specified in the library. These objects can then be used to format printed output.

**page segment**
An object that can contain text and images and be included at any addressable point on a page or electronic overlay. It assumes the environment of an object it is included in.

A library resource that contains the definition of a page segment. Contrast with electronic overlay.

**parameter**
A variable that is given a constant value for a specified application and that can denote the application.

In Page Printer Formatting Aid, the values specified for a subcommand.

**partition**
Dividing the medium presentation space into a specified number of equal-sized areas in a manner that is determined by the current physical media.

In FD:OCA, a conceptual subdivision of a string of data fields. A partition can be further divided into subpartitions.

**pel** Picture element. The smallest printable or displayable unit on a physical medium. In
computer graphics, the smallest element of a physical medium that can be independently assigned color and intensity. Synonymous with picture element and pixel.

PELS In Page Printer Formatting Aid, a unit of measure under the SETUNITS command. See also logical unit.

physical page A single surface (front or back) of a sheet. See also form and page.

picket fence orientation Linear bar code orientation where the bars are perpendicular to the base of the document (like the pickets in a picket fence). Sometimes called horizontal orientation (because that is the direction of the scan).

picture element In computer graphics, the smallest element of a display space that can be independently assigned color and intensity. The smallest area that can be individually toned by the printer.

pixel The smallest printable or displayable unit on a physical medium. Synonymous with pel and picture element.

PMF Print Management Facility.

point In printing, a unit of about 1/72 of an inch used in measuring typographical material; for example, 10-point Helvetica. 12 points equal a pica.

portrait presentation The position of a printed sheet that has its short edges at the top and bottom and its long edges at the sides. Contrast with landscape presentation.

position The location specified for a line or field on the output page.

POSTNET A parameter that specifies a bar code type of POSTal Numeric Encoding Technique (United States Postal Service), and defines specific values for the BSD module width, element height, height multiplier, and wide-to-narrow ratio fields.

PPFA Page Printer Formatting Aid.

preprinted form A sheet of paper that contains a preprinted design of constant data. Variable data can be merged with the constant data on such a form. See also electronic overlay and forms flash.

print line A single line of text. In the formatting of line data, it refers to the output generated by one data record. Governed by the PRINTLINE command.

Print Management Facility (PMF) A program that can create fonts, segments, page definitions, and form definitions.

Print Server Facility (PSF) A program that produces printer commands from the data sent to it.

printer–attached device Either a preprocessor or postprocessor that is attached to the printer.

PSF Print Server Facility.

quiet zone A blank area before and following a bar code. This required space enables the scanner to differentiate the start and stop of a bar code. The size of the quiet zone is usually 10 times the x-dimension or 1/4 inch, whichever is larger.

R

range A table heading for architecture syntax. The entries under this heading give numeric ranges applicable to a construct. The ranges can be expressed in binary, decimal, or hexadecimal. The range can consist of a single value.

raster In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space. In AFP printers, an on-or-off pattern of electrostatic images produced by the laser print head.

RASTER / NORASTER subcommand A subcommand that specifies whether an overlay is to be kept in the printer (3800 only) as raster data.
RATIO
A parameter that specifies the ratio of the wide-element dimension to the narrow-element dimension whenever two different size elements exist.

ratio
The relationship in quantity, amount, or size between two or more things.

record
In programming languages, an aggregate that consists of data objects, possibly with different attributes, with identifiers typically attached to them. In some programming languages, records are called structures.

A set of data that is treated as a unit.

A set of one or more related data items that are grouped for processing.

RM4SCC
A parameter that specifies a 4-state customer code that is defined by the Royal Mail Postal Service of England for bar coding postal code information. See Royal Mail 4 State Customer Code.

resource
A collection of printing instructions, and sometimes data to be printed, that consists entirely of structured fields. A resource object is stored as a member of a library and can be called for by IBM Print Server Facility when needed. The different resource objects are page segments, overlays, form definitions, and page definitions.

resource object
In AFP, a collection of printing instructions, and sometimes data to be printed, that consists entirely of structured fields. A resource object is stored as a member (or file) of a library and can be called for by PSF when needed. The different resource objects include coded font, font character set, code page, page segment, overlay, form definition, and page definition.

RNORMAL
Rotated normal. A Page Printer Formatting Aid parameter that specifies the type of duplex printing. The tops of both sides of a duplex-printed sheet are toward the same physical edge of the sheet, for side binding of the document. Used with landscape-presentation pages.

rotation
The orientation of the characters of a font in regard to the baseline.

Royal Mail 4 State Customer Code (RM4SCC)
A two-dimensional bar code symbology that is developed by the United Kingdom’s Royal Mail postal service for use in automated mail-sorting processes.

RTUMBLE
Rotated tumble. A Page Printer Formatting Aid parameter that specifies a type of duplex printing. It means the top of one side of a duplex-printed sheet and the bottom of the other are toward one physical edge of the sheet, for top binding of the document. Used with landscape-presentation pages.

rule
A solid line of any line width.

x-dimension
The width (usually in thousandths of an inch) of the narrow bar or space of the bar code symbology.

S
SBCS Single-byte character set.

scanner
In bar codes, an electronic device that converts optical information into electrical signals. Sometimes called a reader or decoder.

segment
A collection of composed text and images, which are prepared before formatting and included in a document when it is printed. See page segment.

The resource that contains the structured-field definition of a page segment.

sheet
A single piece of paper. For cut-sheet printers, a synonym for form.

shift-in and shift-out characters (SOSI)
Characters that are used to delimit literals in Page Printer Formatting Aid command streams: X’0E’ and X’0F’.

c simplex printing
A method that is used to print data on one side of a sheet; the other side is left blank. Contrast with duplex printing.

single-byte character set
A character set whose codes require a
single byte of data. The character set used for English is an example.

**skip-to-channel control**
A line printer control that appears in line data. Allows space to be left between print lines. Compatible with page printers when the data is formatted by page definitions.

**space**
In bar codes, the lighter element of a printed bar code symbol, which is typically formed by the background between bars.

**space width**
In bar codes, the thickness of a bar code symbol space measured from the edge closest to the symbol start character to the trailing edge of the same space.

**SSASTERISK**
A parameter that specifies whether an asterisk is to be generated as the HRI for CODE39 bar code start and stop characters.

**start-stop character or pattern**
In bar codes, a special bar code character that provides the scanner with start and stop reading instructions and a scanning direction indicator. The start character is normally at the left end and the stop character at the right end of a horizontally oriented bar code symbol.

**structured field**
A self-identifying string of bytes and its data or parameters.

**subcommand**
In Page Printer Formatting Aid, the next level of control under commands.
A request for an operation that is within the scope of work that is requested by a previously issued command.

**subgroup**
A subset of a form definition that is used to reprint the same page of data more than once. Subgroups provide for variations in the same page of data within one print job. Modifications that distinguish one subgroup from another are number of copies, type of duplex printing, inclusion of overlays, inclusion of suppressions, and (only for the 3800 printer) forms flash. A set of modifications within a copy group that applies to a certain number of copies of a form. A copy group can contain more than one subgroup.

**subpage**
A part of a logical page on which line data can be placed. In the page definition, multiple subpages can be placed on a physical page based on changes in the print data.

**suppression**
The electronic equivalent of a spot carbon, preventing selected data from being printed on certain copies.

**symbology**
A bar code language. Bar code symbologies are defined and controlled by various industry groups and standards organizations. Bar code symbologies are described in public domain bar code specification documents. Synonymous with [bar code symbology]. See also [European Article Numbering (EAN)], [Japanese Article Numbering (JAN)], and [Universal Product Code (UPC)].

**syntax**
The rules that govern the structure of a construct.

**table reference character (TRC)**
Usually, the second byte on a line in the user’s data. This byte contains a value (0 - 126) that is used to select a font to be used to print that line.

**tate**
The Japanese word for top-to-bottom, as applied to the formatting of writing and printing. The traditional arrangement of Japanese kanji characters on the page. Pronounced “ta-tay”.

**text**
A graphic representation of information on an output medium. Text can consist of alphanumeric characters and symbols that are arranged in paragraphs, tables, columns, and other shapes.

**TRC**
Table reference character.

**truncation**
Planned or unplanned end of a presentation space or data presentation.

**tumble duplex printing**
Duplex printing for sheets that are to be bound on the top, as is often done for legal documents. The top of one side of
each sheet is at the same edge as the bottom of the other side. Contrast with normal duplex printing.

**triplet** A three-part self-defining variable-length parameter that consists of a length byte, an identifier byte, and one or more parameter-value bytes.

**type** A table heading for architecture syntax. The entries under this heading indicate the types of data present in a construct. Examples include: BITS, CHARCODE, SBIN, UBIN, UNDF.

**TYPE** A parameter that specifies the kind of barcode symbol to be generated. For example, CODE39, MSI, UPC, and UPCE.

**type font** See font.

**type weight** A parameter that indicates the degree of boldness of a typeface. A character's stroke thickness determines its type weight. Examples are light, medium, and bold.

**type width** A parameter that indicates a relative change from the font's normal width-to-height ratio. Examples are normal, condensed, and expanded.

**U**

**unformatted print data** Data that is not formatted for printing. A page definition can contain controls that map unformatted print data to its output format.

**Uniform Symbol Specification (USS)** A series of barcode symbology specifications that are published by AIM. The specifications that are currently included are USS-Interleaved 2 of 5, USS-39, USS-93, USS-Codabar, and USS-128.

**Universal Character Set (USC)** A printer feature that permits the use of various character arrays. Synonymous with font.

**Universal Product Code (UPC)** A standard barcode symbology commonly used to mark the price of items in stores, that can be read and interpreted by a computer.

**unprintable area** The area of a sheet of paper on which no printing can be done because of printer and hardware limitations.

**UPC** See Universal Product Code.

**UPCA** A parameter that specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version A.

**UPCE** A parameter that specifies a bar code type of Universal Product Code (United States) and the Canadian Grocery Product Code, Version E.

**UPC2SUPP** A parameter that specifies a bar code type of Universal Product Code (United States) two-digit Supplemental (periodicals).

**UPC5SUPP** A parameter that specifies a bar code type of Universal Product Code (United States) five-digit Supplemental (paperbacks).

**uppercase** Pertaining to capital letters. Examples of capital letters are A, B, and C. Contrast with lowercase.

**user-access name** The library resource name of a font or an overlay, less its two-character prefix. Contrast with local name.

**USS** See Uniform Symbol Specification.

**W**

**wide-to-narrow ratio** The ratio of the width of the wide bar or space to the narrow bar (x-dimension) or space in a two-width symbology. This ratio is usually in the range of 2:1 to 3:1.

**width** In Page Printer Formatting Aid, refers to the horizontal dimension of a logical page, is specified in the page definition, and is controlled by the WIDTH subcommand.

**X**

**x-coordinate** The horizontal or inline position that defines a page origin or the starting point of a line or field.
**x-dimension**

The width (usually in thousandths of an inch) of the narrow bar or space of the bar code symbology.
Index

Numerics
2DMATRIX  500, 511
2DMAI  501, 511
2DPDF417  502, 511
3900 Printing Subsystem
restrictions on printing area of sheet  22
A
absolute inline positioning
  in XML page definition  89
  with XLAYOUT command  448
access a new copy group  20
accessibility  587
  contact IBM  587
features  587
AFP print systems
color management  163
AFP Resource installer
  installing data objects  172
AFP Resource Installer
  copying and pasting a name  176
  installing CMRs  163
  storing CMRs  163
alphanumeric characters  202
alphabetical characters  202
maximum number allowed  202
APOSTAL  503, 511
Appendix A
  System Dependencies for PPFA  467
  arrangement of N_UP partitions  141
  assistive technologies  587
  asymmetrical page placement  159
  audit processing mode, CMR  168
Australia Post Bar Code  478, 481
Australia Postal Bar Code  486
B
BACK subcommand
duplexing pages  25
  rules  26
bar code data
  BCOCA characters  477
  BCOCA code points  477
  characters  477
  code 128 code page
    CPGID  477
  code pages  477
  type styles  477
bar codes
  Bar Code Object Content Architecture (BCOCA)  477
  BARCODE subcommand
    supplemental information  477
baseline direction
description  9
basic N_UP printing
  compared to enhanced N_UP  141
description  148
  (continued)
  examples
    normal duplex  152
    tumble duplex  153
    using INVOKE and OVERLAY  150
  list of printers  141
  subcommands and parameters enabled  149
best practices
color resources  173
  images  173
blank characters  202
blank lines in command streams  202
blank truncation, conditional processing  130
body records
  LAYOUT Command  59
  page definition  59
  record format  59
BOTH subcommand
duplexing pages  25
BOTHMARGIN subcommand
  PAGEDEF (record format) Command  61
  PAGEFORMAT (record format) Command  61
bounded-box fonts
description  10
boxes
  create with LAYOUT Command  63
C
carriage control characters  6, 12
carriage control characters (CC)
  relationship with START subcommand  127
change copy group
  with conditional processing  117
change overlays when ejecting new partition  149
change page format
  with conditional processing  117
character rotation  10
Character Set
  four types  202
character sets
description  202
types  202
characters
  alphabetic  202
  blank  202
  number allowed  202
  numeric  202
  shift-out/shift-in codes  202
  special  202
choose copy group
  with conditional processing  117
choose page format
  with conditional processing  117
circles
  create with LAYOUT Command  63
CMR (Form Definition)
syntax diagram  177
CMR subcommand (COPYGROUP)
  Code Example  182
  syntax diagram  181
CMR subcommand (Form Definition)
  Code Example 178
CMR subcommand (PAGEFORMAT)
  syntax diagram 184
CMRs 163
  audit processing modes 168
  color conversion 164
  creating 169
  generic halftone 166
  halftone 165
  installing 169
  instruction processing modes 168
  link color conversion 164
  link processing mode 169
  storing 163
  tone transfer curve 167
Codabar 477, 479, 486, 492
Code 128 477, 479, 484, 486, 492
Code 39 477, 478, 486
Code 93 481, 511
CODE 93 486
Code Example
  for CMR subcommand (COPYGROUP) 182
  for CMR subcommand (Form Definition) 178
  for CMR subcommand (Form Definition) (continued)
    form definition
      RENDER subcommand (COPYGROUP) 184
color conversion CMR 164
color management
  AFP print systems 163
color management resources (CMRs) 163
command delimiters
  description 202
Command Delimiters 202
command stream, examples of
  defining literals 204
  defining logical page size 35, 64
  for form definitions 19
  for page definitions 33, 55
  for record formatting 55
  programmer comments 203
  units of measurement 205
commands
  CONDITION 274
  COPYGROUP 208
  DEFINE COLOR 280
  DEFINE QTAG 282
  description 12
  DOFONT 283
  DRAWGRAPHIC - BOX 289
  DRAWGRAPHIC - CIRCLE 301
  DRAWGRAPHIC - ELLIPSE 307
  DRAWGRAPHIC - LINE 296
  ENDGRAPHIC 313
  ENDSUBPAGE 314
  EXTREF 315
  FIELD 319
  FONT 360
  FORMDEF 20, 234
  LAYOUT 365
  nesting rules
    form definition 20
    page definition 34
  OBJECT 385
  OVERLAY 264, 402
  PAGEDEF 404
  PAGEFORMAT 415
  PRINTLINE 423
  commands (continued)
    rules 201
    SEGMENT 442
    SETUNITS 265, 443
    SUBGROUP 266
    SUPPRESSION 269
    syntax 201
    token rules 201
    TRCREF 445
    XLAYOUT 446
  comments in command streams
    delimiters 203
    location 203
  complex printouts
    creating 107
    example 107
    field processing 107
    overlay, electronic 107
  CONDITION command
    subcommands 275
    syntax diagram 274
    using with enhanced N_UP 160
  CONDITION Command
    blank truncation, consideration 130
    interaction with CHANNEL subcommand 125
    interaction with REPEAT subcommand 125
    selecting copy groups and page formats 128
    variable-length records, consideration 129
  condition processing
    change copy group 117
    change page format 117
    choose copy group 117
    choose page format 117
  conditional processing
    blank truncation 130
    considerations 124
    copy group, selection 118, 128
    description 117
    duplex output example 130
    examples 130
    in PRINTLINE Command 119
    offset stacking (jog) example 130
    page format, selection 119, 128
    paper (bin) selection example 132
    record reprocessing 121
    record reprocessing example 131
    repeated printlines examples 136
    restrictions 124
    rules 124
    setting the environment 118
    subpage, description 121
    using multiple conditions, examples 133
    variable-length records 129
    versus normal line data processing 117
    WHEN CHANGE always false 127
  conditional processing considerations
    LAYOUT (record format) Command 63
    PAGEDEF (record format) Command 63
  constant forms
    description 24
    example 24
  constant overlays
    placement 154
  CONSTANT subcommand
    enhanced N_UP printing example 156
    example 24
contact 587
continuous forms 29
example 30
narrow 30
rules 30
specifying page presentation 29
wide 30
continuous forms printers 30
specifying page presentation for
control information
print data 9
control page ejects 62
control record
Invoke Data Map (IDM) structured field
in PAGEFORMAT command 416
controlling page formatting
LAYOUT (record format) Command 61
COPIES subcommand
example 23
using with enhanced N_UP 160
copy group
accessing 20
description 19
purpose 19
selection, conditional processing 118, 128
starting or restarting 128
COPYGROUP command
subcommands 211
syntax diagram 208
COPYGROUP Command 154
specifying overlays 22
specifying the N_UP subcommand
basic N_UP printing 148
copying and pasting a name
AFP Resource Installer 176
create
boxes
LAYOUT Command 63
circles
LAYOUT Command 63
eclipses
LAYOUT Command 63
lines
LAYOUT Command 63
creating CMRs 169

D
data
positioning 65
positioning first line 35
data file types
line 6
mixed 7
MOD:DCA-P 7
data lengths 477
data map
description 14
invoke 14
PAGEFORMAT command 416
Data Matrix 480
Data Matrix (2D barcode) 478
Data Object fonts
definition support
DOFONT command 283
DOFONT command 283
data objects
description 170
Data Record Types 58
DataMatrix two-dimensional Bar Code 486
default x-pos
body records 59
default y-pos
body records 59
DEFINE COLOR command
subcommands 281
syntax diagram 280
DEFINE QTAG Command
syntax diagram 282
defining color models
LAYOUT (record format) Command 64
PAGEDEF (record format) Command 64
delimiters
description 202
diagram
shorthand 273
diagram shorthand 205, 273
differences in measurements and REPEATs 475
dimensions of logical page 34
direction
description 9
change print direction of logical page 37, 65
inline, description 9
of fonts 49, 75
relationship to duplex 29
specifying
for fields 48, 72
for lines 48, 72
in a page definition 37, 65
dIRECTION subcommand
changing logical page print direction 37, 65
description 30
element 37, 48, 65, 72
LAYOUT Command 59
using with enhanced N_UP 160
when to use 30
DOFONT command
non-PPFA requirements 284
PPFA requirements 284
subcommands 285
syntax diagram 283
double-byte font 329
DRAWGRAPHIC - BOX command
subcommands 290
syntax diagram 289
DRAWGRAPHIC - CIRCLE command
subcommands 302
syntax diagram 301
DRAWGRAPHIC - ELLIPSE command
subcommands 308
syntax diagram 307
DRAWGRAPHIC - LINE command
subcommands 297
syntax diagram 296
duplex 19
duplex printing
description 130
conditonal processing example 130
example of basic N_UP printing 152
in landscape presentation 27

Index 615
duplex printing (continued)
in portrait presentation 27
normal duplex 14
possible combinations 29
rotated normal duplex 14
rotated tumble duplex 14
specifying in form definition 25
tumble duplex 14
using BACK subcommand 25
using BOTH subcommand 25
using FRONT subcommand 25
DUPLEX subcommand
example 25, 27
NORMAL parameter 28
RNORMAL parameter 28
RTUMBLE parameter 28
TUMBLE parameter 28
DUTCH KIX 510
Dutch KIX bar code 480

E
EAN 477, 479, 490, 494, 495
EAN 13 (includes JAN-standard) 486
EAN 8 (includes JAN-short) 486
EAN Five-digit Supplemental 486
EAN Two-digit Supplemental 486
EBCDIC data
blank characters 202
shift-out/shift-in codes 202
ellipses
create with LAYOUT Command 63
Encapsulated PostScript (EPS)
image data object 170
ENDGRAPHIC command
subcommands 314
syntax diagram 313
ENDSUBPAGE command
syntax diagram 314
enhanced N_UP printing 154
compared to basic N_UP 141
examples
asymmetric pages 158
using CONSTANT and OVERLAY 156
using PLACE 155
list of printers 141
subcommands and parameters enabled 153
error messages 567
examples for Data Object fonts
DOFONT command 288
examples for OpenType fonts
DOFONT command 288
examples for TrueType fonts
DOFONT command 288
EXTREF command
subcommands 315
syntax diagram 315

F
field
direction of 48, 72
Field (record format) Command
LAYOUT Command 60
FIELD command
bar code, supplemental information 477

FIELD command (continued)
subcommands 324
SUPPRESSION subcommand
name 331
syntax diagram 319
FIELD Command
example 43, 49, 69, 72
LENGTH parameter 44, 71
nesting in LAYOUT Commands 72
nesting in PRINTLINE Commands 49
specifying location 44, 71
START parameter 44, 71
field processing
combining data 113
combining with overlay 107
positioning fields 44, 71
rule 69
rules 43
selection of fields 44, 71
use of fixed text with 110
fields, printing
in two directions 48, 72
first line of data
positioning 35
fixed text
example 110
in page definition 110
FONT command
example 45, 49, 73, 75
rotating fonts 49, 75
subcommands 362
syntax diagram 360
FONT subcommand
example 50, 76
rotating data 50, 76
fonts
bounded-box 10
in tate presentation 51, 77
naming in a page definition 45, 73
rotation of 49, 75
specified
for PRINTLINE Command 46
unbounded-box 10
varying on a page 45, 73
fonts, examples for Data Object
DOFONT command 288
fonts, examples for OpenType
DOFONT command 288
fonts, examples for TrueType
DOFONT command 288
fonts, support for Data Object
DOFONT command 283
fonts, support for Open Type
DOFONT command 283
fonts, support for TrueType
DOFONT command 283
form definition
CMR subcommand (COPYGROUP) 181
CMR subcommand (Form Definition) 177
CMR subcommand (PAGEFORMAT) 184
command nesting 20
command reference 207
contents of 4
copy groups in 19
COPYGROUP command 208
defining overlays 23
description 4
form definition (continued)
duplex printing
  using NORMAL 27
  using RTUMBLE 28
  using subgroups 25
  using TUMBLE 28
example command streams 19
examples
  asymmetric pages 159
  normal duplex 152
  tumble duplex 153
  using CONSTANT and OVERLAY 157
  using INVOKE and OVERLAY 150
  using PLACE 155
FORMDEF command 234
logical pages 21
OVERLAY command 264
page definition 3
print jobs requiring 6
RENDER subcommand (COPYGROUP) 182
RENDER subcommand (FORMDEF) 179
RENDER subcommand (in a PAGEFORMAT) 185
sequence of commands for 207
SETUNITS command 265
specifying the N_UP subcommand 148
steps for creating 3
storage location 3
SUBGROUP command 266
SUPPRESSION command 269
using commands 19
formatting multiple applications pages
on a single sheet 161
FORMDEF command
  subcommands 237
  syntax diagram 234
FORMDEF Command
  specifying DIRECTION DOWN 31
  specifying the N_UP subcommand 148
  basic N_UP printing 148
  enhanced N_UP printing 154
FORMDEF Parameters
  PPFA system dependencies 471
  VM 471
FRONT subcommand
duplexing pages 25
  rules 26

G
generic halftone CMR 166
graphical objects subcommand
  LAYOUT (record format) Command 63
  PAGEDEF (record format) Command 63
Graphics Interchange Format (GIF)
  image data object 170
Group Headers
  LAYOUT Command 60
guidelines about creating and managing
  color resources 173
  images 173

H
halftone CMR 165
HEIGHT subcommand
  example 35, 64

I
identify filed in Data Record 60
IDM structured field
  and the PAGEFORMAT command 416
image data object
  Encapsulated PostScript (EPS) 170
  Graphics Interchange Format (GIF) 170
  Image Object Content Architecture (IOCA) 171
  Portable Document Format (PDF) 171
  Portable Network Graphics (PNG) 171
  Tagged Image File Format (TIFF) 171
image data objects in print jobs 171
Image Object Content Architecture (IOCA)
  image data object 171
  in COPYGROUP Command 154
Industrial 2-of-5 477, 479, 486, 491
inline direction
  description 9
installing CMRs 169
installing data objects
  AFP Resource Installer 172
instruction processing mode, CMR 168
Interleaved 2-of-5 477, 479, 486, 491
invoke
  new copy group 149
Invoke Data Map (IDM) structured field
  and the PAGEFORMAT command 416
INVOKE subcommand
  basic N_UP printing example 150

J
Japan Postal Bar Code 477, 480, 486
job control language (JCL) for z/OS 469
job control statements (JCS) for VSE 467
jog (offset stacking)
  conditional processing example 130
JPOSTAL 497, 511

K
kanji print presentation
  example 51, 77
  tate 51, 77
keyboard
  navigation 587
  PF keys 587
  shortcut keys 587

L
landscape presentation
  description 10
  specifying on continuous-forms printers 29
  with duplex printing 27
  with OFFSET subcommand 22
layout
  description 9
LAYOUT (record format) Command
  conditional processing considerations 63
  defining color models 64
  graphical objects subcommand 63
  logical page eject processing 63
  PAGE NUMBERING subcommand 62
  record formatting examples 77
LAYOUT command
   subcommands 367
   syntax diagram 365

LAYOUT Command
   example 72
   Field (record format) Command 60
   GROUP Headers 60
   in field processing 69
   Page Headers and Trailers 59
   printing direction 72
   types of Data Records 58

LAYOUT Commands
   in page definition 58
   layout position 71
   library-resource name
   description 23
   line data
   description 6
   printing, print server printer 39
   record format 7
   structured fields 13
   traditional 7
   line data processing
   versus conditional processing 117

LINEONE subcommand
   example 35
   positioning first line of data 35
   lines, create with LAYOUT Command 63
   lines, printing
   in four directions 72
   in two directions 48

LINESP subcommand
   positioning the first line of data 36
   link color conversion CMR 164
   link processing mode, CMR 169
   literals
   description 204
   syntax 204
   used in TEXT subcommand 204
   used in WHEN subcommand 204
   local name
   description 23
   logical page
   defining size 34, 64
   description 8
   height 64
   positioning 21
   size 34, 64
   specifying the origin 21
   width 64
   logical page dimensions 34
   logical page eject processing
   LAYOUT (record format) Command 63
   PAGEDEF (record format) Command 63
   logical page origin
   printline position 44
   logical page position 19

M

managing resource library 172
mapping fields
   to printed sheets 69
   mapping fields to printed sheets 43
   margins 67
   Matrix 2-of-5 477, 479, 486, 491

MaxiCode 480
MaxiCode (2D barcode) 478
MaxiCode two-dimensional Bar Code 486
measurement
   differences in repeated lines 475
   units, described 204
media origin 8
medium map
   description 14
   invoke 14
medium overlay
   description 160
   using with N_UP 160
messages and codes 567
mixed data
   description 7
MO:DCA-P data
   description 7
MOD parameter
   bar code type 485
   MOD value 485
modifications
   description 11
MSI 477, 478, 486, 487
multiple conditions, conditional processing examples 133
multiple-up function 161
multiple-up printing
   compared to N_UP printing 8, 161
   conditional processing 121
   description 51
   example 51

N

N_UP considerations
   CONDITION subcommand 160
   COPIES subcommand 160
   DIRECTION subcommand 160
   OVERLAY subcommand 160
   PRESENT subcommand 160
   SUPPRESSION subcommand 160

N_UP partitions
   arrangement 141
   description 11, 141
   N_UP printing
   basic description 141
   basic N_UP printing 141
   compared to multiple-up printing 8, 161
   enhanced N_UP printing 141
   examples
      asymmetric pages 158
      normal duplex 152
      tumble duplex 153
      using CONSTANT and OVERLAY 156
      using INVOKE and OVERLAY 150
      using PLACE 155
   list of printers 141
   partition arrangement 141
   partitions 141
   N_UP subcommand 154
   basic N_UP printing 149
   in COPYGROUP Command 148
   in FORMGROUP Command 148, 154
   name
      SUPPRESSION subcommand
      for FIELD command 331
names
  character length allowed 202
library-resource 23
local 23
overlay 23
resource 23
user-access 23
NAMES in PPF 202
narrow forms
definition 30
navigation
  keyboard 587
nesting rules
  commands
    form definition 20
    page definition 34
NEWFORM parameter
  using with enhanced N_UP 160
NEWSIDE parameter
  using with enhanced N_UP 160
No Operation (NOP) 14
normal duplex
definition 14
element 152
normal line data processing
  versus conditional processing 117
NORMAL parameter
description 27
Notices 591
numeric characters 202
numeric values description 204

O
object
  include 14
OBJECT command
  subcommands 387
  syntax diagram 385
offset stacking 19
  example, conditional processing 130
OFFSET subcommand
  example 21, 22
  landscape presentation 22
positioning a logical page 21
  rotated print directions 22
Open Type fonts
  support
    DOFONT command 283
OpenType fonts
  examples
    DOFONT command 288
origin
  logical page, definition 21
  specifying with OFFSET subcommand 21
OTHERWISE subcommand
  interaction with REPEAT subcommand 125
output
  formatting different data types 6
overlay
  invoked by a form definition 160
  invoked by a page definition 160
OVERLAY command
  subcommands 264, 403
  syntax diagram 264, 402
OVERLAY Command
  example 22

OVERLAY subcommand
  basic N_UP printing example 150
  enhanced N_UP printing example 156
  using with enhanced N_UP 160
overlays
  combining with field processing 107
  examples of invoking 160
  form definition example 23
  local name 23
  names 23
  system name 23
overlays in different subgroups 22

P
page definition
  command nesting 34, 57
CONDITION command 274
  contents of 5
  defining font rotation 50, 76
  defining individual lines 40
description 5
  example command streams 33, 55
  field processing 44, 70
  fixed text 110
  FONT command 360
  formatting lines 48, 72
  incorporating fixed text into 110
  multiple-up printing 52
  naming fonts 45, 73
OBJECT command 385
OVERLAY command 402
page formats in 33, 56
  page sequence, alteration of 53
PAGEDEF command 404
PAGEFORMAT command 415
  positioning of data 65
  positioning the first line of data 35
  print jobs requiring 6
PRINTLINE command 423
SEGMENT command 442
  sequence of commands for 271
SETUNITS command 443
  size of logical pages 34, 64
steps for creating 3
tasks 33
  TRCREF command 445
  using commands 33, 55
page definitions, sequence of commands for 272
page ejects 62
page format
  purpose 33, 56
  selecting with conditional processing 119, 128
  starting or restarting 128
Page Headers and Trailers
  LAYOUT Command 59
page margins 61, 67
page numbering subcommand
  LAYOUT (record format) Command 62
  PAGEDEF (record format) Command 62
page overlay
  description 160
  include 14
page presentation
  example 29
page printers, use of line data with 39
PAGEDEF command
  subcommands 406
  syntax diagram 404

PAGEDEF Parameters
  PPFA system dependencies 470
  VM 470

PAGEFORMAT command
  and the IDM structured field 416
  subcommands 417, 425
  syntax diagram 415

paper source
  selection, conditional processing example 132

parameters
  description 12
  entry order 201
  for CMR subcommand (COPYGROUP) 181
  for CMR subcommand (FORMDEF) 177
  for DEFINE CMRNAME subcommand (FORMDEF and all
  PAGEDEF types) 175
  for RENDER subcommand 182, 185
  for RENDER subcommand (FORMDEF) 179

PDF417 480
PDF417 (2D barcode) 478
PDF417 two-dimensional Bar Code 486

physical page
  description 8
  place pages in partitions
  any sequence 153

PLACE subcommand
  enhanced N_UP printing 153
  example 155

Portable Document Format (PDF)
  image data object 171

Portable Network Graphics (PNG)
  image data object 171

portrait presentation
  description 10
  specifying on continuous-forms printers 29
  with duplex printing 27

position first line of data
  logical page 35
  position of logical page 19

POSITION subcommand
  first line of data 35
  in PRINTLINE Command 44
  processing fields 44, 71
  positioning data 65

POSTNET 477, 480, 486, 495

PPFA
  basic terms 8
  concepts 8

PPFA command stream
  rules for creating 201

PPFA Commands and Syntax 199
  predefined a color 64

PRESENT subcommand
  description 30
  example 30
  producing readable output 31
  specifying LANDSCAPE 31
  using with enhanced N_UP 160
  when to use 30

presentation
  description 10
  example 29
  landscape 22
  specifying for continuous-forms printers 29

print data
  control information 9

print direction
  baseline 9
  inline 9

print directions
  changing 37

print jobs
  including image data objects 171

print quality
  selection 32
  specifying level 32

printers
  used in N_UP printing 141

printings
  BACK subcommand 25
  basic N_UP 149
  basic N_UP example
    normal duplex 152
    tumble duplex 153
    using INVOKE and OVERLAY 150

BOTH subcommand 25
  constant forms 24
  controlling direction 48, 72
duplex 14
  landscape presentation 27
  portrait presentation 27
duplicate example 25
  enhanced N_UP 153
  enhanced N_UP example
    asymmetric pages 158
    using CONSTANT and OVERLAY 156
    using PLACE 155

FRONT subcommand 25
  line data 39
  lines in four directions 72
  lines in two directions 48
  multiple up 51
  on both sides 25
two versions of same data 109
  using form definitions 3
  using page definitions 3
duplex area
  for 3900 22

printing area
  printing page numbers 62
duplex area
  description 8

PRINTLINE command
  syntax diagram 423

PRINTLINE Command
  conditional processing 119
  defining individual lines 40
  example 40, 48
  in field processing 43
  printing direction of 48
  specifying fonts for 46

printline position
  logical page origin 44
subcommands (continued)
  for OVERLAY command 264, 403
  for PAGedef command 406
  for SETUNIT command 265
  for SETUNITS command 444
  for SUBGROUP command 266
  for TRCREF command 445
  for XLYOUT command 449
  PAGEFORMAT command 417, 425
  RENDER (COPYGROUP) 182
  RENDER (FORMDEF) 179
  RENDER (in a PAGEFORMAT) 185

SUBGROUP command
  subcommands 266
  syntax diagram 266

subgroups
  description 20
  use in duplex printing 25

subcommands
  for DOFONT command 285
  subpage
    conditional processing 121
    description 8
  support for Data Object fonts
    DOFONT command 283
  support for Open Type fonts
    DOFONT command 283
  support for TrueType fonts
    DOFONT command 283

suppression
  description 109
  example 109
  SUPPRESSION command
    syntax diagram 269
  SUPPRESSION subcommand
    name
      for FIELD command 331
      using with enhanced N_UP 160
  syntax
    literals 204
  syntax diagram
    for DEFINE QTAG command 282
    for DRAWGRAPHIC - BOX command 289
    for DRAWGRAPHIC - CIRCLE command 301
    for DRAWGRAPHIC - ELLIPSE command 307
    for DRAWGRAPHIC - LINE command 296
    for ENDGRAPHIC command 313
    for LAYOUT command 365
    for XLYOUT command 446

form definition
  CMR (Form Definition) 177
  CMR subcommand (COPYGROUP) 181
  COPYGROUP command 208
  for FORMDEF command 234
  OVERLAY command 264
  RENDER subcommand 179
  RENDER subcommand (COPYGROUP) 182
  RENDER subcommand (in a PAGEFORMAT) 185
  SETUNITS command 265
  SUBGROUP command 266
  SUPPRESSION command 269

page definition
  CMR subcommand (PAGEFORMAT) 184
  CONDITION command 274
  DEFINE COLOR command 280
  DOFONT command 283
  ENDSUBPAGE command 314

syntax diagram (continued)
page definition (continued)
  EXTREF command 315
  FIELD command 319
  FONT command 360
  OBJECT command 385
  OVERLAY command 402
  PAGedef command 404
  PAGEFORMAT command 415
  PRINTLINE command 423
  SEGMENT command 442
  SETUNITS command 443
  TRCREF command 445

syntax diagrams
  fragment elements xvii
  how to interpret xv
  optional parameters xvi
  reading order xv
  repeating parameters xvii
  required parameters xv
  style rules xv
  symbols xv
  SYSIN data definition, z/OS 469
  System Dependencies for PPFA
    Appendix A 467
  system name
    description 23

T
  table reference characters (TRC)
    relationship with START subcommand 127
  table-reference characters 6, 12
  Tagged Image File Format (TIFF)
    image data object 171
  tate 51, 77
  test input data
    with conditional processing 117
  token rules 201
  tokens
    creating 202
    definition 201
  tone transfer curve CMR 167
  TOPMARGIN subcommand
    PAGedef (record format) Command 61
    PAGEFORMAT (record format) Command 61
  traditional line data
    basic controls 12
    carriage control characters 12
    description 7
    record ID characters 13
    table-reference characters 12
  TRCREF command
    subcommands 445
    syntax diagram 445
  TrueType fonts
    examples
      DOFONT command 288
      support
      DOFONT command 283
  tumble duplex
    definition 15
    example 153
  TUMBLE parameter
    description 27
  tumble printing
    example of basic N_UP printing 133
two sides, printing on 25

U

UCC/EAN 128 479, 492, 493, 494
unbounded-box fonts
  description 10
units of measurement
  description 204
  specifying 204
unprintable area
  for 3900 22
UPC 477, 478, 486, 488, 489
UPC - Five-digit Supplemental 486
UPC - Two-digit Supplemental 486
UPC/CGPC Version E 486
user interface
  ISPF 587
  TSO/E 587
user-access name
  description 23
USPS Four-State 481, 486

V

variable-length records, conditional processing 129
VM
  FORMDEF Parameters 471
  PAGEDEF Parameters 470
  PPFA execution 469
  PPFA system dependencies 469, 470, 471
VSE
  PPFA execution 467
  Rules 468
  VSE Environment
    PPFA system dependencies 467

W

WHEN subcommand
  at start of a page format 127
wide forms
  definition 30
WIDTH subcommand
  example 35, 64

X

X LAYOUT command
  absolute inline positioning 448
  relative inline positioning 448
  subcommands 449
  syntax diagram 446
XML command sequence
  for XML page definitions 272
XML commands
  command sequence 272
XML data elements 90
XML page definition
  absolute inline positioning 89
  formatting function 88
  relative inline positioning 88
  sequence of commands for 272