

Advanced Function Presentation



# Application Programming Interface: Programming Guide and Reference





Advanced Function Presentation



# Application Programming Interface: Programming Guide and Reference



**Note!**

Before using this information and the product it supports, be sure to read the general information in "Notices" on page ix.

**Third Edition (February 1996)**

This edition applies to Print Services Facility/VM Version 2 Release 1 Modification 1, Print Services Facility/MVS Version 2 Release 2 Modification 0, and Print Services Facility/VSE Version 2 Release 2 Modification 1, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. See the Summary of Changes for the changes made to this publication. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change. Be sure to use the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

The IBM Printing Systems Company welcomes your comments. For your convenience, a form for reader's comments is provided at the back of this publication. You may either send your comments by fax to 1-800-524-1519, or by mail to:

INFORMATION DEVELOPMENT  
THE IBM PRINTING SYSTEMS COMPANY  
DEPARTMENT H7FE BUILDING 003G  
PO BOX 1900  
BOULDER CO 80301-9191

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

© Copyright International Business Machines Corporation 1993, 1994, 1996. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Notices</b> . . . . .	ix
Programming Interface Information . . . . .	ix
Trademarks . . . . .	ix
<b>About this Publication</b> . . . . .	xi
Who Should Read This Publication? . . . . .	xi
How to Use this Publication . . . . .	xi
How Is this Publication Organized? . . . . .	xii
<b>Summary of Changes</b> . . . . .	xiii
<b>Chapter 1. Advanced Function Presentation Concepts</b> . . . . .	3
The Evolution of Printing and Presentation . . . . .	3
What Is Advanced Function Presentation? . . . . .	4
AFP Documents, Pages, and Resources . . . . .	4
How Is Printing with AFP Different? . . . . .	5
AFP Documents and Pages . . . . .	8
Data Objects and AFP Resource Objects . . . . .	11
AFP Data Objects . . . . .	11
AFP Resource Objects . . . . .	12
Indexing AFP Data for Viewing and Archiving . . . . .	14
<b>Chapter 2. Using AFP API</b> . . . . .	17
Creating the Sample Document . . . . .	18
Getting Started . . . . .	23
Program Template . . . . .	24
Putting Data on the Page . . . . .	26
Character String . . . . .	28
Rule . . . . .	31
Resources . . . . .	33
Paragraphs . . . . .	36
Areas . . . . .	42
Tables . . . . .	48
Box . . . . .	62
Include Object . . . . .	63
Introducing Return Codes and Severity Codes . . . . .	65
Setting Up and Defining the Environment for an AFP API Session . . . . .	65
Setting Output Characteristics and Resource Libraries . . . . .	66
Buffering AFP API Output . . . . .	66
Defining Fonts and Using Them with AFP API . . . . .	67
Setting Attributes (and Querying Them) . . . . .	73
Understanding States and Handles . . . . .	75
Understanding States . . . . .	75
Understanding Handles . . . . .	77
Indexing Data for Viewing and Archiving . . . . .	79
What's Involved? . . . . .	79
What Are the Indexing Procedure Calls? . . . . .	79
Determining Page Breaks and Changing Page Layout . . . . .	82
Specifying Presentation Options . . . . .	84
Using AFP API in a CICS/ESA Environment . . . . .	85
Defining the Temporary Storage Queue for AFP Output . . . . .	85

Using IOCA and GOCA Objects	85
Creating VSAM Data Sets for Fonts and Page Segments	86
Using the Error-Checking Routine in APQPERF	86
Link-Editing Your Program with AFP API	86
Improving Performance	87
Coding Tips	88
Troubleshooting Your Program	89
Debugging Errors in Your Application Program	89
Modifying the Error-Checking Routine Supplied with AFP API	90
<b>Chapter 3. Procedure Call Reference</b>	<b>93</b>
The Application Programming Interface Program	93
Format of the AFP API Procedure Call Descriptions	98
AFPBDOC (Begin Document)	100
AFPBFLD (Begin Field)	104
AFPBGRP (Begin Group)	106
AFPBPAG (Begin Page)	108
AFPBPARG (Begin Paragraph)	112
AFPBROW (Begin Row)	116
AFPBTBL (Begin Table)	118
AFPCARE (Create Area)	121
AFPDFLD (Define Field)	124
AFPDFNT (Define Font by Attributes)	128
AFPDROW (Define Row)	131
AFPEARE (End Area)	134
AFPEDOC (End Document)	136
AFPEFLD (End Field)	137
AFPEGGRP (End Group)	138
AFPEND (End AFP API)	140
AFPEPAG (End Page)	141
AFPEPAR (End Paragraph)	143
AFPEROW (End Row)	145
AFPETBL (End Table)	147
AFPGBUF (Get Output Buffer)	149
AFPINIT (Initialize AFP API)	151
AFPINVM (Invoke Medium Map)	152
AFPIOBJ (Include Object)	154
AFPIOVL (Include Page Overlay)	158
AFPIPSG (Include Page Segment)	160
AFPPARE (Put Area)	162
AFPPBOX (Put Box)	164
AFPPCHS (Put Character String)	166
AFPPRUL (Put Rule)	169
AFPPTAG (Put Tag)	171
AFPPTXT (Put Text)	173
AFPQATT (Query Current Attributes)	175
AFPQPOS (Query Current Position)	177
AFPQSTR (Query Character String Size)	179
AFPSCLR (Set Color)	181
AFPSFNT (Set Font)	183
AFPSICS (Set Intercharacter Spacing)	185
AFPSLIB (Set Resource Library Names)	187
AFPSOUT (Set Output Characteristics)	190
AFPSPOS (Set Position)	193
AFPSRTH (Set Rule Thickness)	195

AFPSUNI (Set Units)	197
AFPSWSP (Set Word Spacing)	199
AFPTERM (Terminate AFP API)	201
AFPXARE (Destroy Area)	202
<b>Appendix A. Font Library Indexing Program (FLIP)</b>	<b>203</b>
Invoking the Font Library Index Program in VM	203
Invoking the Font Library Index Program in MVS	204
Invoking the Font Library Index Program in a CICS/ESA Environment	205
Invoking the Font Library Index Program in VSE	205
Font Library Index Program Return Codes	207
<b>Appendix B. Return Codes and Severity Codes</b>	<b>209</b>
AFP API Return Codes	210
<b>Appendix C. Shade Patterns and Types</b>	<b>261</b>
<b>Appendix D. Creating an Executable Program under MVS</b>	<b>265</b>
MVS JCL for Compiling and Link-Editing a COBOL Application	265
MVS JCL for Running a COBOL Application	267
MVS JCL for Compiling and Link-Editing a PL/1 Application	268
MVS JCL for Running a PL/1 Application	271
MVS JCL for Compiling and Link-Editing a COBOL Application in a CICS/ESA Environment	273
<b>Appendix E. Creating an Executable Program under VM</b>	<b>275</b>
VM EXEC for Compiling a COBOL Application	275
VM EXEC for Link-Editing and Running a COBOL Application	276
VM EXEC for Compiling a PL/1 Application	278
VM EXEC for Link-Editing and Running a PL/1 Application	279
<b>Appendix F. Creating an Executable Program under VSE</b>	<b>281</b>
VSE JCL for Compiling and Link-Editing a COBOL Application	281
VSE JCL for Running a COBOL Application	283
<b>Appendix G. AFP API Macros Used as Programming Interfaces</b>	<b>285</b>
General-Use Programming Interfaces	285
<b>Appendix H. Related Publications</b>	<b>287</b>
<b>Glossary</b>	<b>291</b>
Source Identifiers	291
References	291
<b>Index</b>	<b>301</b>





---

## Figures

1.	Billing Statement Produced using Line Data and a Page Definition . . . .	6
2.	Billing Statement Produced using the AFP Data Stream . . . . .	7
3.	Billing Statement Produced using AFP API . . . . .	8
4.	Logical Page Coordinate System . . . . .	9
5.	Medium Coordinate System . . . . .	9
6.	Relationship between the Logical Page and Media Coordinate Systems	10
7.	Sample Document . . . . .	22
8.	Document Elements . . . . .	23
9.	Files Shipped with AFP API . . . . .	27
10.	Character String . . . . .	28
11.	Sample Rule . . . . .	31
12.	Page Segment (Art) . . . . .	33
13.	Paragraph . . . . .	36
14.	Area Containing an Overlay . . . . .	42
15.	Document Elements . . . . .	47
16.	The Header Row of the Table . . . . .	48
17.	One Row of the Table . . . . .	50
18.	Sample Document . . . . .	57
19.	Box . . . . .	62
20.	Example of the IOCAMMR Data Object Shipped with PSF . . . . .	63
21.	Information Flow . . . . .	65
22.	Information Flow Using Buffered Output . . . . .	67
23.	Character String . . . . .	68
24.	Portion of a Sample of Font Library Index Program Listing . . . . .	71
25.	Hierarchy of AFP API States . . . . .	75
26.	An Example Showing the Use of Handles . . . . .	78
27.	Sample Document with Different Page Layouts . . . . .	82
28.	Coding the Sample Document for Page Breaks . . . . .	83
29.	The Hierarchy of States in AFP API . . . . .	93
30.	Format of the AFPBDOC Procedure Call . . . . .	100
31.	Page Orientation of 0° . . . . .	101
32.	Page Orientation of 90° . . . . .	102
33.	Format of the AFPBFLD Procedure Call . . . . .	104
34.	Format of the AFPBGRP Procedure Call . . . . .	106
35.	Format of the AFPBPAG Procedure Call . . . . .	108
36.	Page Orientation of 0° . . . . .	109
37.	Page Orientation of 90° . . . . .	110
38.	Format of the AFPBPAR Procedure Call . . . . .	112
39.	Format of the AFPBROW Procedure Call . . . . .	116
40.	Format of the AFPBTBL Procedure Call . . . . .	118
41.	Format of the AFPCARE Procedure Call . . . . .	121
42.	Format of the AFPDFLD Procedure Call . . . . .	124
43.	Format of the AFPDFNT Procedure Call . . . . .	128
44.	Format of the AFPDROW Procedure Call . . . . .	131
45.	Format of the AFPEARE Procedure Call . . . . .	134
46.	Format of the AFPEDOC Procedure Call . . . . .	136
47.	Format of the AFPEFLD Procedure Call . . . . .	137
48.	Format of the AFPEGRP Procedure Call . . . . .	138
49.	Format of the AFPEND Procedure Call . . . . .	140
50.	Format of the AFPEPAG Procedure Call . . . . .	141
51.	Format of the AFPEPAR Procedure Call . . . . .	143

52.	Format of the AFPEROW Procedure Call	145
53.	Format of the AFPETBL Procedure Call	147
54.	Format of the AFPGBUF Procedure Call	149
55.	Format of the AFPINIT Procedure Call	151
56.	Format of the AFPINVM Procedure Call	152
57.	Format of the AFPIOJB Procedure Call	155
58.	Format of the AFPIOVL Procedure Call	158
59.	Format of the AFPIPSG Procedure Call	160
60.	Format of the AFPPARE Procedure Call	162
61.	Format of the AFPPBOX Procedure Call	164
62.	Format of the AFPPCHS Procedure Call	166
63.	Format of the AFPPRUL Procedure Call	169
64.	Format of the AFPPTAG Procedure Call	171
65.	Format of the AFPPTXT Procedure Call	173
66.	Format of the AFPQATT Procedure Call	175
67.	Format of the AFPQPOS Procedure Call	177
68.	Format of the AFPQSTR Procedure Call	179
69.	Format of the AFPSCLR Procedure Call	181
70.	Format of the AFPSFNT Procedure Call	183
71.	Format of the AFPSICS Procedure Call	185
72.	Format of the AFPSLIB Procedure Call	187
73.	Format of the AFPSOUT Procedure Call	190
74.	Format of the AFPSPOS Procedure Call	193
75.	Format of the AFPSRTH Procedure Call	195
76.	Format of the AFPSUNI Procedure Call	197
77.	Format of the AFPSWSP Procedure Call	199
78.	Format of the AFPTERM Procedure Call	201
79.	Format of the AFPXARE Procedure Call	202
80.	Shade Pattern—STANDARD	262
81.	Shade Pattern—SCREEN	263
82.	JCL to Compile and Link-Edit a COBOL Application in an MVS System	265
83.	JCL to Execute a COBOL Program in an MVS System	267
84.	JCL to Compile and Link-Edit a PL/1 Application in an MVS System	268
85.	JCL to Execute a PL/1 Program in an MVS System	271
86.	JCL to Create an Executable Program in a CICS/ESA Environment	273
87.	VM EXEC to Compile a COBOL Application	275
88.	VM EXEC to Link-Edit a COBOL Application and Run it	276
89.	VM EXEC to Compile a PL/1 Application	278
90.	VM EXEC to Link-Edit a PL/1 Application and Run it	279
91.	JCL to Compile and Link-Edit a COBOL Application in a VSE System	281
92.	JCL to Run a COBOL Application in a VSE System	283

---

## Notices

References in this publication to products or services of IBM do not suggest or imply that IBM will make them available in all countries where IBM does business or that only products or services of IBM may be used. Noninfringing equivalents can be substituted, but the user must verify that such substitutes, unless expressly designated by IBM, work correctly. No license, expressed or implied, to patents or copyrights of IBM is granted by furnishing this document.

---

## Programming Interface Information

This publication is intended to help the customer program AFP applications with high-level programming languages, such as COBOL. This publication documents General-Use Programming Interface and Associated Guidance Information provided by the AFP Application Programming Interface.

General-Use Programming Interfaces allow the customer to write programs that obtain the services of AFP API.

---

## Trademarks

The following terms appear in this publication and are trademarks or registered trademarks of the IBM Corporation:

- Advanced Function Presentation
- AFP
- BookManager
- C/370
- CICS
- CICS/ESA
- GDDM
- IBM
- Print Services Facility
- PSF
- S/370
- System/370

The following terms appear in this publication and are trademarks of other companies:

- ElixirForm for AFP and ElixirImage for AFP are trademarks of Elixir Technologies Corporation.
- ISIS and FormsDesigner are trademarks of ISIS Information Systems.
- Windows is a trademark of the Microsoft Corporation.

The examples in this publication are for purposes of illustration only. Any resemblance to existing businesses or people is unintentional.



---

## About this Publication

This publication describes the Advanced Function Presentation Application Programming Interface (AFP API) product and how to use it.

---

### Who Should Read This Publication?

This publication is for COBOL and PL/1 application programmers who want to use AFP API to:

- Develop new Advanced Function Presentation (AFP) applications
- Revise existing applications so that they can be migrated to AFP
- Insert indexing codes into a document for online viewing

The programming language you can use depends on the operating system:

- On the MVS and VM operating systems, you can use either the PL/1 or the COBOL programming language.
- On the MVS operating system in a CICS/ESA environment, you can use the COBOL programming language.
- On the VSE operating system, you can use the COBOL programming language.

---

### How to Use this Publication

Before reading about AFP API and its code, you need general knowledge about Advanced Function Presentation concepts and terminology. Read Chapter 1, “Advanced Function Presentation Concepts” to familiarize yourself with AFP.

If you are familiar with AFP, turn to Chapter 2, “Using AFP API” for a step-by-step description of AFP API and an explanation of how to use it.

To determine whether you want to use AFP API, see the following sections:

- “How Is Printing with AFP Different?” on page 5
- “AFP with AFP API” on page 8

---

## How Is this Publication Organized?

This publication introduces key AFP concepts and describes the concepts needed to use AFP API. The publication uses the code for a sample document shipped with the product to illustrate AFP API concepts.

The publication contains the following information:

- Chapter 1, “Advanced Function Presentation Concepts,” which introduces key concepts of AFP
- Chapter 2, “Using AFP API,” which introduces the concepts needed to use AFP API
- Chapter 3, “Procedure Call Reference,” which contains reference information for all AFP API calls
- Appendix A, “Font Library Indexing Program (FLIP),” which contains information about the fonts on your system
- Appendix B, “Return Codes and Severity Codes,” which contains the return codes and severity codes issued by AFP API
- Appendix C, “Shade Patterns and Types,” which contains examples of the shading patterns available for use with AFP API
- Appendix D, “Creating an Executable Program under MVS,” which contains information about printing AFP API output using PSF/MVS
- Appendix E, “Creating an Executable Program under VM,” which contains information about printing AFP API output using PSF/VM
- Appendix F, “Creating an Executable Program under VSE,” which contains information about printing AFP API output using PSF/VSE
- Appendix G, “AFP API Macros Used as Programming Interfaces,” which contains a list of macros that are General-Use Programming Interfaces of AFP API
- Appendix H, “Related Publications,” which contains lists of publications that might be helpful when you are using AFP API

---

## Summary of Changes

In addition to editorial changes, the following changes are included in this edition of the publication, S544-3872-02:

- Changes documented in the *Print Services Facility/MVS: Update Guide*, G544-3984-00, are incorporated.
- Support for output buffering is added. Changes include:
  - The AFPSOUT (Set Output Characteristics) procedure call allows you to request that AFP API write output to a buffer in your program.
  - AFPSOUT also allows you to request that AFP API discard the output instead of returning it to your program.
  - The new AFPGBUF (Get Output Buffer) procedure call returns the AFP buffered output to your application program.
  - New codes 280, 281, and 282 are returned.
- Support for the Customer Information Control System (CICS/ESA) running under the MVS operating system is added. Changes include:
  - The Set Output Characteristics procedure call allows you to name the CICS/ESA temporary storage queue for the AFP output.
  - The Include Object procedure call is not supported.
  - The Set Resource Library Names procedure call is ignored.
  - Fonts and page segments must be in VSAM data sets defined to CICS/ESA.
  - Sample JCL to compile and link edit a CICS/ESA program with AFP API is provided.
- The new Query Character String Size procedure call allows you to determine the size of the area required to print a character string in the current font. New codes 284 and 285 are returned.
- The Set Position procedure call now allows your application program to place data in *any* order on the page.
- You can now issue the Define Row and Define Field procedure calls *only* in the document state. You can no longer issue these calls in the page or area state.
- You can issue the Put Area procedure call *only* in the page state, not in the area state.
- You must set the Concatenation parameter to TRU on the first Put Text procedure call in a field or paragraph.
- New conditions for issuing return codes 0071, 0217, and 0218 are added.
- All fonts are provided in the IBM Font Collection product and are no longer shipped with PSF.
- The trace function is no longer supported on the Initialize AFP procedure call.
- You must include new AFP API modules when link-editing your API application. Also, you no longer need the C/370 libraries when link-editing or running your COBOL programs.

- Tips to improve performance, code your program, and debug your program are included.
- Return codes are now listed with the description of each procedure call.
- Sample JCL to compile, link-edit, and run your PL/1 programs is included.

Technical changes to the text made in this edition are indicated by a vertical line to the left of the changes.

The following changes were included in the previous edition of this publication, S544-3872-01:

- Support for the VSE operating system and the COBOL programming language was added.
- A description of the VM search order was added to the Set Resource Library Names procedure call.
- A new condition for issuing return code 0217 was added: The default coded font was not found in the font library.
- The descriptions of the Include Object POINT-TO-PEL and POINT-TO-PEL with DOUBLE DOT options was changed.
- An appendix containing the macros provided as General-Use Programming Interfaces was added.



---

# Chapter 1. Advanced Function Presentation Concepts

<b>Chapter 1. Advanced Function Presentation Concepts</b> . . . . .	3
The Evolution of Printing and Presentation . . . . .	3
What Is Advanced Function Presentation? . . . . .	4
AFP Documents, Pages, and Resources . . . . .	4
How Is Printing with AFP Different? . . . . .	5
AFP Before AFP API . . . . .	7
AFP with AFP API . . . . .	8
AFP Documents and Pages . . . . .	8
Data Objects and AFP Resource Objects . . . . .	11
AFP Data Objects . . . . .	11
AFP Resource Objects . . . . .	12
Fonts . . . . .	12
Page Segments . . . . .	12
Overlays . . . . .	13
Form Definitions . . . . .	13
Page Definitions . . . . .	13
Indexing AFP Data for Viewing and Archiving . . . . .	14



---

## Chapter 1. Advanced Function Presentation Concepts

### Please Read

This chapter contains Advanced Function Presentation concepts and a description about indexing data for viewing and archiving. Read this chapter first if you are unfamiliar with AFP, because to use AFP API, you need to understand a few things about AFP.

Even if you are familiar with AFP, read “Indexing AFP Data for Viewing and Archiving” on page 14 to see whether indexing is something you can use, then skip the rest of this chapter and turn to Chapter 2, “Using AFP API.”

This chapter describes:

- How printing has evolved into today’s printing and presentation
- AFP concepts
- AFP documents, pages, and resources
- How AFP printing is different from line-oriented printing
- Resource objects
- Data objects
- How to index AFP data for viewing and archiving

When you finish this chapter, you should be ready to use Chapter 2, “Using AFP API” and the rest of this publication and the other AFP API publications to produce AFP output from your COBOL or PL/1 applications.

---

## The Evolution of Printing and Presentation

Printing has evolved from slow, fifteenth-century moveable-type processes to today’s high-speed, high-quality, computer-driven printers. Today’s print technology enables you to combine the high quality of Renaissance printing with the speed and convenience of computer-generated data. Now, in addition to printing lines of text, applications can print data that includes a variety of output such as logos, pie charts, graphs, signatures, and bar codes. Programs can print these types of data today, because AFP printers can place data at any addressable point on the page.

Although all-points-addressability (APA) produces high-quality output, it dramatically increases the complexity of formatting the data for printing. Using early printing technologies, programmers placed each line of data. With APA printers, programmers can place each pel (an abbreviation for picture element) of data, and many printers can address over 50 000 pels in each square inch of paper! Before the availability of APA printing, programmers needed to set the spacing between individual lines of text; with APA, programmers can position multiple types of data anywhere on the page.

Moreover, advances in the resolution of computer displays have enabled an evolution from printing to online viewing of information for some applications. Data formatted for APA printing can also be presented on a display, with the same fidelity as the printed output. Because distributing information in softcopy form is often less expensive than distributing printed output, viewing is emerging as an alternative to printing for information presentation. However, to make

information retrieval as easy to perform with softcopy data as it is with hardcopy, the data must include indexing information comparable to the table of contents and the index found in printed output. Inserting that indexing information in the softcopy output is thus a new requirement for application programmers to consider.

AFP API is a tool that helps application programmers manage the complexity of both placing data for APA printing and including indexing information in viewable output. With AFP API, you don't need to understand the format of AFP data used for either hardcopy or softcopy presentation. Instead, you issue calls from within your application program to access AFP API functions to generate AFP-format output. With AFP API, you can integrate and manipulate other application data, and you can include and reference objects, such as artwork, within a document. Language bindings exist for using AFP API with COBOL and PL/1 programs in the MVS and VM operating systems and with COBOL in the VSE operating system.

Before learning more about AFP API, read the rest of this chapter to understand AFP.

---

## What Is Advanced Function Presentation?

Advanced Function Presentation (AFP) is a collection of hardware and software products that generates high-quality presentation output from data-processing systems, making computer output more readable and attractive. AFP is implemented with an architected data stream (the AFP data stream) consisting of *structured fields* of presentation information. Structured fields are self-identifying, variable-length records containing control information and data. The two major elements in the AFP data stream are the *document* and several types of *resources*.

## AFP Documents, Pages, and Resources

An AFP document consists of data that has been formatted into a series of one or more AFP data stream pages. Each page contains the actual output text characters for that page, as well as information about the placement of each character. In addition to text, the following types of data can be included on a page:

- Graphics and images (line art, pie charts, business graphics, and scanned data such as photographs or facsimile)
- Logos
- Signatures
- Lines (rules) and boxes
- Bar code data
- Indexing information that is ignored when the AFP document is printed but that is useful for navigating through a displayed document

Programs can store all these types of data (except indexing information) outside the document and can use the types of data in multiple pages within a document or in multiple applications. These external files are called AFP resources, which are described more fully in "Data Objects and AFP Resource Objects" on page 11.

## How Is Printing with AFP Different?

Before AFP and APA printers, applications generated output for printing one line at a time; this output was called *line data*. The output lines were created in the order in which they were to appear on the printed page. Limited positioning controls called *carriage control characters* and *channel codes* could be used to place output lines on the paper. The application used carriage control characters to space down 0, 1, 2, or 3 lines on the form, allowing for overstriking (to create the effect of a **bold** font) and for single-, double-, or triple-spacing between lines. The application used channel codes to skip to 1 of 12 different locations on the form, including the top of the form to start a new form.

Interpreting carriage control characters and channel codes included in line data was determined outside the application program. For the early impact line printers, carriage control tapes were punched and loaded on the carriage (form transport) mechanism of the printer. Holes in the tapes corresponded to the locations on the carriage at which the form would be positioned when a carriage control character or channel code was encountered in the print data. Nonimpact line printers (such as the IBM 3800 Printing Subsystem Model 1) provided a software equivalent of the carriage control tape called the *forms control buffer* (FCB).

With AFP, programs can use an external file called a *page definition* to map the positioning controls to locations on the form, with the additional flexibility of addressing more locations and using such enhancements as overlays and typographic fonts. Figure 1 on page 6 shows the application output for a typical billing statement using line data formatted with a page definition.

## AFP Line Data Application (limited formatting)

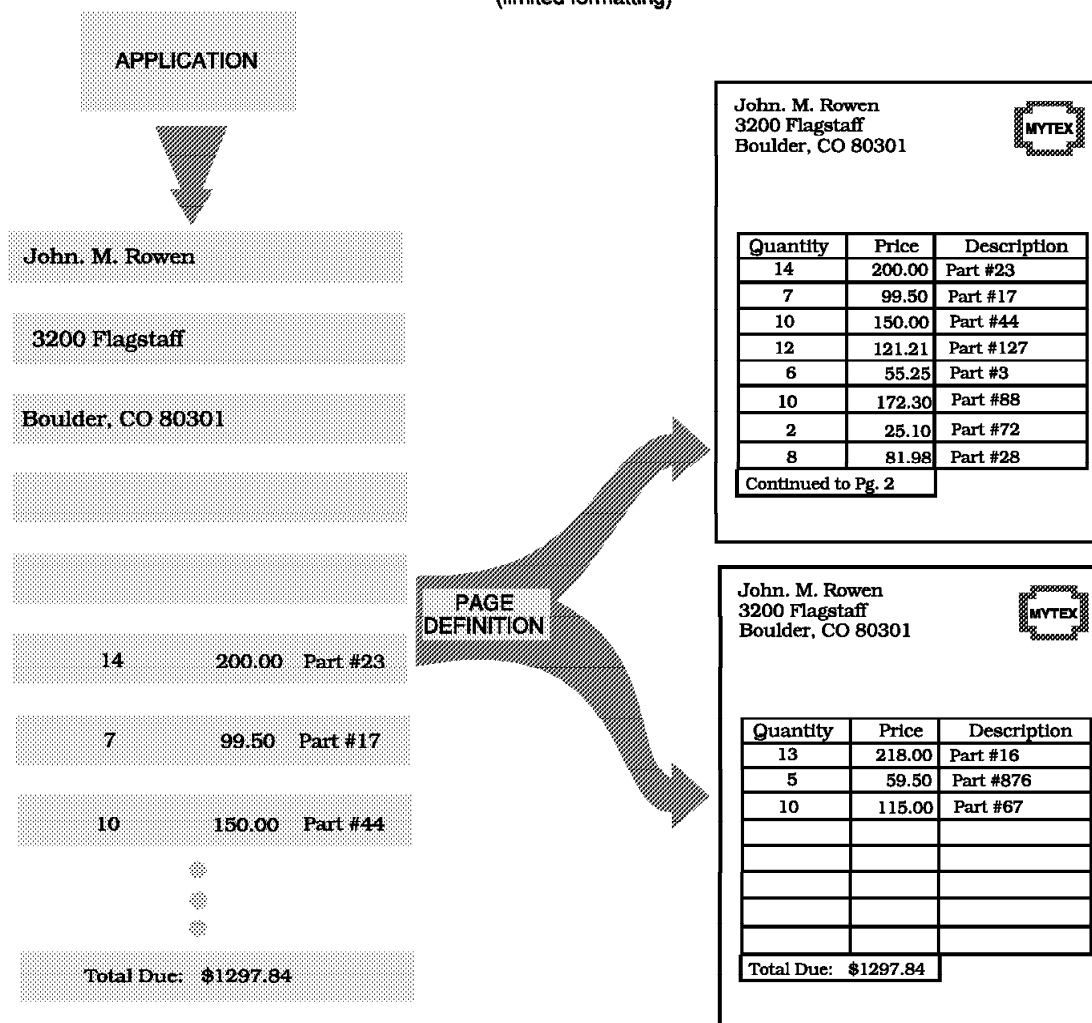


Figure 1. Billing Statement Produced using Line Data and a Page Definition. Notice the unused rows in the table and the use of two sheets of paper.

The page definition also permits applications to format output without the carriage control characters and channel codes. This is because the page definition can place individual parts of the records (fields of data) at any location on the form. For example, an application can produce a single record containing the customer identification information and another record for each customer transaction. The page definition can format the customer identification information into an address label format and format each transaction into columns in a tabular arrangement.

With this use of the page definition, the program can format the printed output separately from generating the output data. However, the functions provided in the page definition require that the application generate uniform data for each customer. In the above example, if the customer transactions are of two different types (such as deposits and withdrawals), and if each customer has a different number of transactions of each type, coding the page definition to stop formatting deposits and start formatting withdrawals at the right time is not easy.

Also, the page definition does not support functions such as flowing text into paragraphs, centering text, or drawing a box around a variable amount of data. To obtain these AFP functions and to handle output, such as the withdrawal and deposit example, you must change the application to produce the AFP data stream.

### AFP Before AFP API

Changing applications to produce the AFP data stream to generate AFP output used to require you to learn the rather complex syntax and semantics of the AFP data stream. For example, the statement shown in Figure 1 on page 6 contains unused rows and requires two sheets to present all the output. As shown in Figure 2, you can eliminate the unused rows, and you can place the data on a single sheet by producing AFP data stream output from the application.

### AFP Page Data Stream Application

(better appearance than line data)

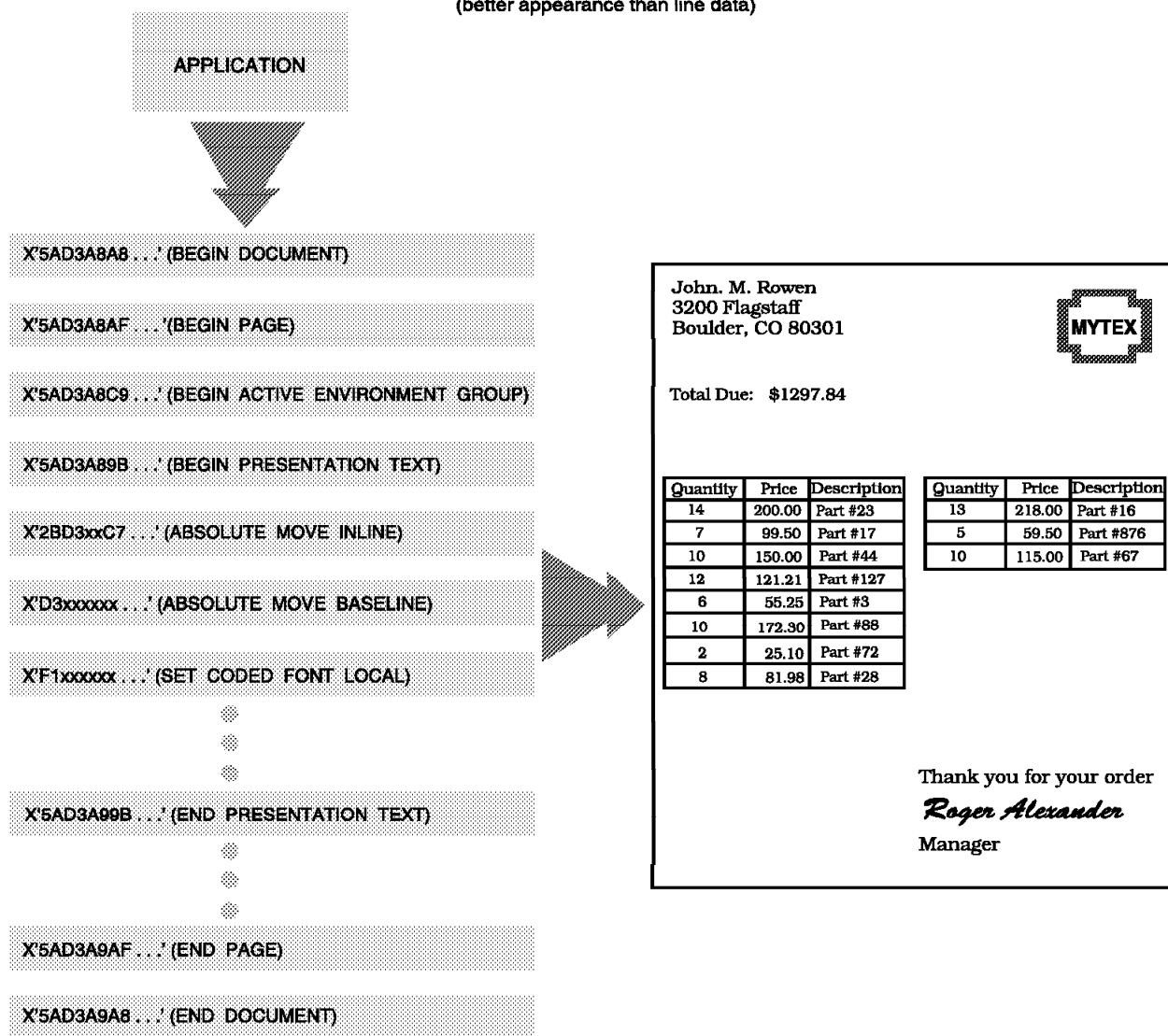


Figure 2. Billing Statement Produced using the AFP Data Stream

## AFP with AFP API

With AFP API, you can use procedures called using the COBOL or PL/1 programming languages to produce the AFP data stream, instead of using page definitions or complicated AFP data stream coding. On the MVS and VM operating systems, you can use either COBOL or PL/1; on the VSE operating system, you can use COBOL. On the MVS operating system, you can write COBOL programs that run under the Customer Information Control System (CICS). Figure 3 shows a portion of AFP API coding to produce the same output for the billing statement shown in Figure 2 on page 7.

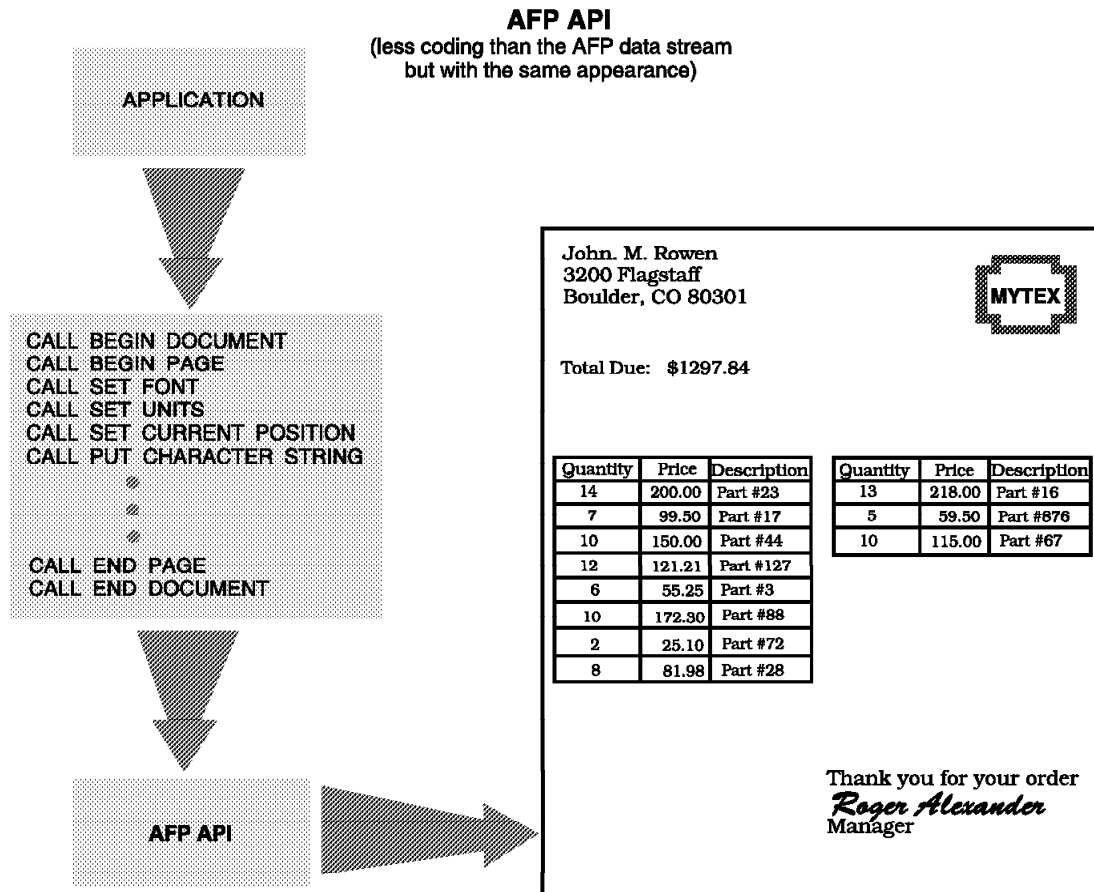


Figure 3. Billing Statement Produced using AFP API. The output is the same as in Figure 2 on page 7, but the AFP API code to generate the AFP data stream is much simpler.

## AFP Documents and Pages

An AFP document consists of one or more pages of AFP data. An AFP page is also called a *logical page* because it is not bound to a particular physical entity such as a certain form or display. The logical page is the electronic representation of the page that will ultimately be presented.

The logical page has dimensions and a coordinate system similar to that of a physical form or medium. The logical page coordinate system is referred to as the Xp, Yp coordinate system in the AFP data stream. The logical page coordinate system is shown in Figure 4 on page 9.



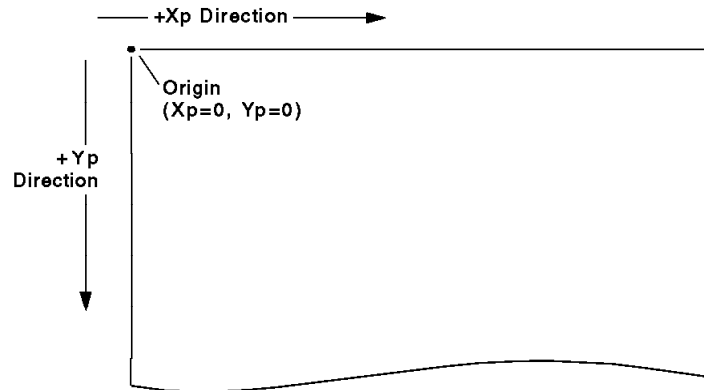


Figure 4. Logical Page Coordinate System

The origin of this system ( $X_p=0$ ,  $Y_p=0$ ) is always defined at the top-left corner of the logical page, at the *logical page origin*. Positive  $X_p$  values begin at the origin and increase along the top of the logical page from left to right. Positive  $Y_p$  values begin at the origin and increase along the left side of the logical page from top to bottom. The size and units of measurement for each logical page of AFP data are specified within the page itself.

The *physical medium*, whether it is a form or display, also has dimensions and a coordinate system. The medium coordinate system is referred to as the  $X_m$ ,  $Y_m$  coordinate system in the AFP data stream. The medium coordinate system is shown in Figure 5.

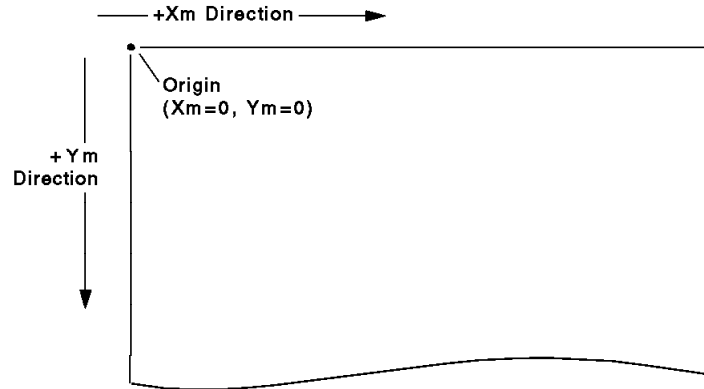


Figure 5. Medium Coordinate System

The origin of this system ( $X_m=0$ ,  $Y_m=0$ ) is always defined at the top-left corner of the medium at the *media origin*. Positive  $X_m$  values begin at the origin and increase along the top of the medium from left to right. Positive  $Y_m$  values begin at the origin and increase along the left side of the medium from top to bottom. The media origin used by a printer can be affected by how forms are fed through the printer. To determine the media origin used by your printer, refer to *Advanced Function Presentation: Printer Information*.

In AFP, data is positioned on a logical page relative to the logical page origin. The logical page is positioned on the medium by aligning the top of the logical page with the top of the medium and positioning the logical page origin relative to the media origin. That is, the logical page does not rotate; the objects and text within the logical page rotate relative to the logical page origin. Figure 6 shows the relationship between the coordinate systems. This relative location, often called the *logical page offset*, is contained in an AFP resource called the *form definition*. The form definition used to present a document is specified when the document is submitted for printing or opened for viewing.

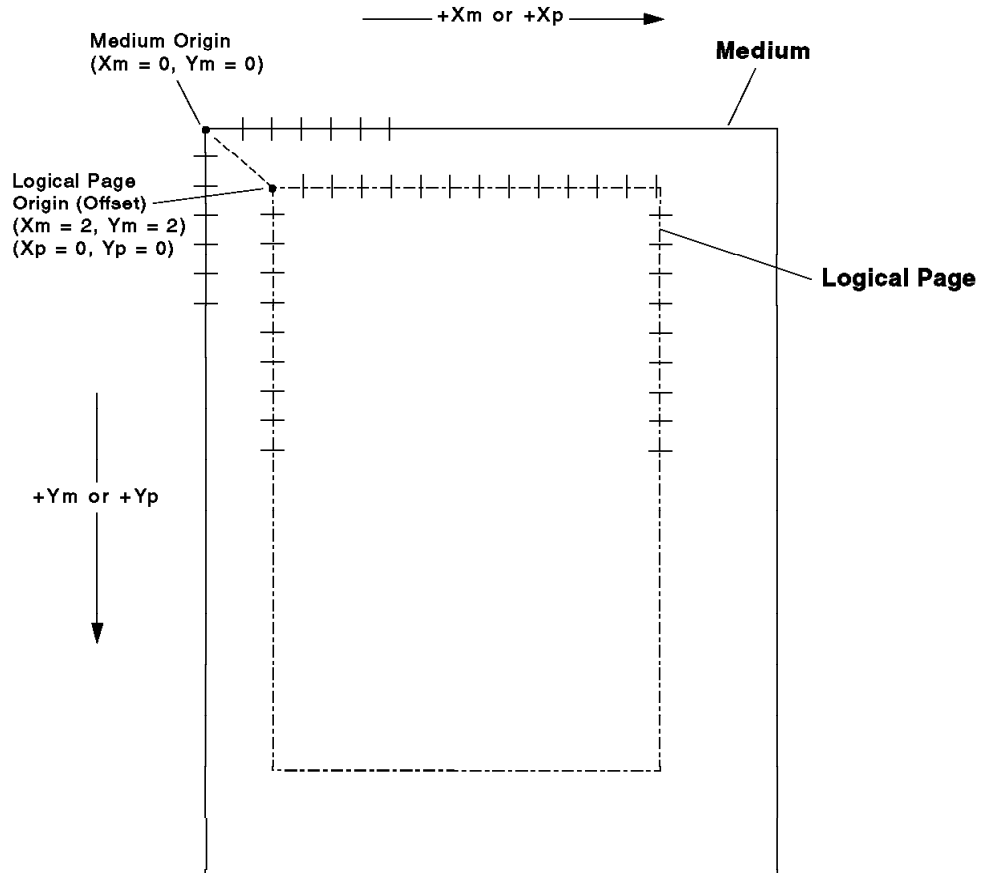


Figure 6. Relationship between the Logical Page and Media Coordinate Systems

Each logical page does not need to contain within itself all the data that will ultimately be presented on a medium. The page can refer to external files (AFP resources), which AFP presentation programs can retrieve when the document is submitted for printing or opened for viewing. The next section describes these AFP resources.

---

## Data Objects and AFP Resource Objects

An AFP data object contains a single type of presentation data; that is, presentation text, vector graphics, raster images, or bar codes, plus all the controls required to present the data. Applications can generate some types of data as AFP data objects and then include the data objects in their output. Programs can format each data object separately from any other data object on a page, and programs can present multiple objects of the same or different types in any sequence on a page.

An AFP resource object is a collection of presentation instructions and data consisting entirely of AFP structured fields. Resource objects are referenced by name in the presentation data stream and can be stored in system libraries so that multiple applications and the print server can use them. Some types of resource objects can also be included (inline) in the presentation file, so that library access is not required when printing or viewing the file.

### AFP Data Objects

The AFP data objects supported by AFP API are:

- Graphics objects (in Graphics Object Content Architecture or GOCA format) that contain the type of vectored data used for line art drawing
- Image objects (in Image Object Content Architecture or IOCA format) that contain raster data such as that produced by a facsimile or scanning device

Unlike resources such as overlays and page segments, AFP API can manipulate image and graphics objects to change the size and orientation of the object on the presentation medium.

For more information about each type of data object, refer to the architecture publication for each object type listed in Appendix H, "Related Publications."

## AFP Resource Objects

The five types of AFP resource objects are:

- **Fonts**, which are collections of graphics characters of a given size and style used to present text.
- **Page segments**, which are collections of image graphics or text data objects that can be presented at any location on a page. Examples of items that can be page segments include logos, signatures, bar charts, and engineering drawings.
- **Overlays**, which are collections of predefined data objects, such as boxes, lines, shading, text, logos, and graphics, that can be merged with application data for presentation. Overlays are often used as electronic forms.
- **Form definitions**, which contain information that defines the presentation of the page on the medium, such as where the page should be placed on the medium and whether the data should be printed on one or both sides of the paper.
- **Page definitions**, which contain information that formats line data into AFP pages.

The following sections provide additional information about using these resources.

### Fonts

Fonts present text characters requested for presentation in a page, an overlay, or a page segment. Local font identifiers are used in the data stream to select fonts. These identifiers are equated to the resource files containing the actual font data. For more information about AFP fonts and how to use them, see “Defining Fonts and Using Them with AFP API” on page 67.

### Page Segments

Page segments can contain a mixture of text, image objects, and graphics data objects and can be placed anywhere on a presentation page. Programs can request page segments for presentation in a page or overlay. Because page segments inherit the environment (such as the local font identifiers) defined by the page or overlay that includes them, you can think of them as pieces of pages. Page segments are used for logos, signatures, and boilerplate.

## Overlays

Overlays can contain a mixture of text, image, graphics, and bar code data objects and also can include page segments. Unlike page segments, overlays contain all of the environment information required for their presentation.

AFP defines two types of overlays: *medium overlays* and *page overlays*. Medium overlays are referenced by a form definition and are positioned on the medium relative to the media origin. Page overlays are referenced by a page and are positioned on the medium relative to the page origin.

## Form Definitions

Form definitions contain instructions about how to map pages of data to a physical medium. A form definition can request that a medium overlay be presented on the medium with the page of data. You must use a form definition whenever you print or display an AFP document.

Each form definition contains one or more *copy groups*, which can be changed between pages of a document to dynamically select the form-mapping controls for subsequent pages. Using copy groups (also called *medium maps*), the application can specify the following form-mapping controls:

- Position the logical page on the physical medium
- Print on both sides of a sheet of paper (*duplex printing*)
- Include medium overlays
- Select the number of copies of any page of data (to replace traditional multipart forms)
- Suppress selected fields (to replace the use of spot carbons)
- Specify offset stacking of cut-sheet output or marking the edges of continuous-forms output
- Select among input bins on a printer
- Select the level of print quality
- Specify the page presentation (*portrait* or *landscape*)

## Page Definitions

Page definitions contain instructions for formatting data intended for a line printer into pages that can be printed on an AFP printer. A page definition is used whenever you print a line-data file on an AFP printer, but a page definition is not needed when you print or display an AFP file. With AFP API, page definitions are not necessary, because AFP API produces an AFP file rather than a line-data file. The information contained in the page definition is already contained in the output that AFP API produces from the application.

---

## Indexing AFP Data for Viewing and Archiving

As described in “The Evolution of Printing and Presentation” on page 3, documents designed for viewing on a workstation should contain indexing information to facilitate navigating through the document. Archival and retrieval applications can use indexing information to identify separate parts of a large print file for saving or restoring.

Using indexing, you can logically segment a large print file into uniquely-identifiable “logical documents.” Bank-statement applications, for example, can create large print files, with each print file containing thousands of individual statements. Each of these statements can be thought of as a “logical document” and can be uniquely identified by an attribute such as an account number. Other attributes, such as date and type of account, can further identify a specific customer statement.

AFP API provides procedure calls that generate indexing information, with which you can:

- Define boundaries of logical documents (called *groups*) in the AFP API output document, for example, the start and end of the statement for each customer in the file containing many customer statements.
- Identify a group of pages with an indexing *tag* containing an attribute name and value, for example, an Account Number attribute associated with the account number of each customer. This type of tag is called a *group-level* tag because it is associated with a group of pages.
- Identify a single page with a tag containing an indexing attribute name-value pair, for example, a Total Charges attribute associated with the charges for each specific customer. This type of tag is called a *page-level* tag because it is associated with a single page.

Using AFP Workbench for Windows, the Viewer Application, you can locate a group of pages using the indexing attribute names and values defined for each group. You can navigate through a large file to locate a single customer statement more quickly than by performing a string search on, for example, a customer’s name. For more information about using groups for navigation in AFP Workbench for Windows, the Viewer Application, refer to the help screens provided with the Viewer application.

For the Viewer application or an archival and retrieval program to take full advantage of the indexing information in a document, you can create an AFP *index object file* that identifies the location of all of the groups and tags in the document. You cannot create an index object file with AFP API, but you can use AFP Conversion and Indexing Facility (ACIF) to create it from an indexed document created by AFP API. ACIF is provided with PSF/MVS Version 2.1.1, PSF/VM Version 2.1.1, and PSF/VSE Version 2.2.1. For more information about creating an index object file, refer to *AFP Conversion and Indexing Facility: Application Programming Guide*.

“Indexing Data for Viewing and Archiving” on page 79 describes the AFP API procedure calls to use to insert indexing information in your output document.

---

## Chapter 2. Using AFP API

<b>Chapter 2. Using AFP API</b> .....	17
Creating the Sample Document .....	18
Getting Started .....	23
Program Template .....	24
Putting Data on the Page .....	26
Character String .....	28
Rule .....	31
Resources .....	33
More About Resources .....	34
Paragraphs .....	36
More About Paragraphs .....	40
Areas .....	42
More About Areas .....	47
Tables .....	48
More About Tables .....	58
Step 1. Sketch the Row .....	59
Step 2. Define all of the Fields in the Row .....	59
Step 3. Form a Grid .....	59
Step 4. Determine the Number of Columns and Their Widths .....	59
Step 5. Determine the Number Subrows and Their Depths .....	60
Step 6. Determine the Arrangement of Each Field Within Each Subrow .....	60
Step 7. Define The Row .....	60
Box .....	62
Include Object .....	63
Introducing Return Codes and Severity Codes .....	65
Setting Up and Defining the Environment for an AFP API Session .....	65
Setting Output Characteristics and Resource Libraries .....	66
Buffering AFP API Output .....	66
Defining Fonts and Using Them with AFP API .....	67
Selecting the Font You Want .....	68
Font Library Indexing Program .....	69
Setting Attributes (and Querying Them) .....	73
Understanding States and Handles .....	75
Understanding States .....	75
Understanding Handles .....	77
Indexing Data for Viewing and Archiving .....	79
What's Involved? .....	79
What Are the Indexing Procedure Calls? .....	79
Determining Page Breaks and Changing Page Layout .....	82
Specifying Presentation Options .....	84
Using AFP API in a CICS/ESA Environment .....	85
Defining the Temporary Storage Queue for AFP Output .....	85
Using IOCA and GOCA Objects .....	85
Creating VSAM Data Sets for Fonts and Page Segments .....	86
Using the Error-Checking Routine in APQPERF .....	86
Link-Editing Your Program with AFP API .....	86
Improving Performance .....	87
Coding Tips .....	88
Troubleshooting Your Program .....	89
Debugging Errors in Your Application Program .....	89
Modifying the Error-Checking Routine Supplied with AFP API .....	90





---

## Chapter 2. Using AFP API

---

### General-Use Programming Interfaces

---

The macros identified in this chapter in Figure 9 on page 27 are provided as programming interfaces for customers by AFP API.

**Warning:** Do not use as programming interfaces any AFP API macros other than those identified in this chapter.

---

### End of General-Use Programming Interfaces

---

These macros are listed separately in Appendix G, “AFP API Macros Used as Programming Interfaces.”

#### Please Read

If you are not familiar with Advanced Function Presentation, read Chapter 1, “Advanced Function Presentation Concepts” before reading this chapter. You must understand AFP concepts before you can understand and use AFP API.

This chapter describes the components of AFP API and how they work together, using an example as a model. The publication uses COBOL to illustrate the code for the example. PL/1 programmers can translate the COBOL explanations and code to PL/1 language bindings to run the samples.<sup>1</sup>

The sequence of this chapter is as follows:

1. First is a “template” with key initialization calls. You can insert code into the template to put data on the page. The actual initialization code for the sample includes calls for defining the environment, which will be described later in the chapter.
2. Next is the code for putting various types of data on the page; that is, the data to insert into the template. In this publication, the term *page* means logical page, not the physical medium or sheet of paper, unless otherwise noted.
3. Finally, the chapter describes the initialization calls for defining the environment (such as setting up output characteristics and defining fonts) and describes *handles* and *states*.

---

<sup>1</sup> The PL/1 programming language is supported only on the VM and MVS operating systems. PL/1 is not supported in a CICS/ESA environment.

The files that contain the COBOL and PL/1 source code for the example used in this chapter are shipped with the product. The source code for the examples are printed in *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference*. You can insert the code for various parts of the example (character string, rule, paragraph, area, and table) into the template in “Getting Started” on page 23 and print each part. You can put the code for all of the parts in the template and print the entire example. The code shown uses copy files that contain constants and variables. These copy files are provided with AFP API.

When you finish studying this chapter, you should be ready to follow the source code in *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference* and modify it to suit your needs.

---

## Creating the Sample Document

The following sections describe the tasks to create the sample document shown on the next three pages. The sample document has three pages, but a document can have many pages.

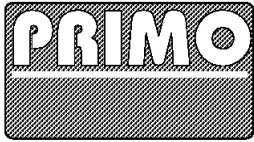


**Susan B. Ames**  
 98765 N. Frontage Road  
 Tin Cup, Colorado 80000

ACCOUNT NUMBER	9999 8888 7777 6666
Payment Due Date	October 15, 1992
New Balance	\$3,572.15
Minimum Payment Due	\$357.22
Enter Amount Enclosed	\$
Make Check Payable to <i>The Primo Card</i> . Please return this portion with your payment.	

**CONGRATULATIONS, Susan B. Ames!** Because of your excellent credit rating, you are now eligible for free credit insurance which protects you in case your Primo card is ever lost or stolen. Call NOW for more information!

Date	Transaction Description	Amount
08/20	CNTRL RSRV -Lodging Winter Park CO	\$173.12
08/24	The Last Dance Winter Park CO	\$28.95
08/25	Jenos Cheese House Winter Park CO	\$44.08
08/25	Ticket Sales Winter Park CO	\$34.00
08/25	Ticket Sales Winter Park CO	\$34.00
08/25	Ski School Winter Park CO	\$90.61
08/25	Gas-N-Go Winter Park CO	\$25.00
08/25	Winter Park Restaurant Winter Park CO	\$55.22
09/01	Alex's Restaurant Boulder CO	\$22.03
09/03	Foot and Ankle Clinic Boulder CO	\$90.61
09/03	Coconut Brewery Boulder CO	\$59.07
09/05	Suds Your Duds Boulder CO	\$69.00
09/06	Fay CO Boulder CO	\$167.89
09/06	Kids Etc Tiny Town CO	\$55.02
09/09	Torrence Restaurant Boulder CO	\$20.15
09/10	Boulder SKIs Boulder CO	\$264.52
09/12	Pelican Jake's Boulder CO	\$90.59
09/17	FAPA Auto Parts Boulder CO	\$167.71
09/17	Harold's Restaurant Boulder CO	\$30.98
09/18	Clothes Line Boulder CO	\$260.53
09/18	Jake's Boulder CO	\$60.50
09/18	Walt's Jewelry Boulder CO	\$1,200.00
09/18	Rutger's Cards and Books Boulder CO	\$13.92
09/19	Fill-R-Up Denver CO	\$18.50
09/19	Best Buy Air Denver CO	\$299.95
09/20	Fred's Dance School Denver CO	\$75.43
09/20	Joe's Grill Denver CO	\$15.77



---

Date	Transaction Description	Amount
09/20	Smith Gas Denver CO	\$10.00
09/20	Pete's Pony Rides Gunbarrel CO	\$95.00
<b>Total Amount</b>		<b>\$3,572.15</b>



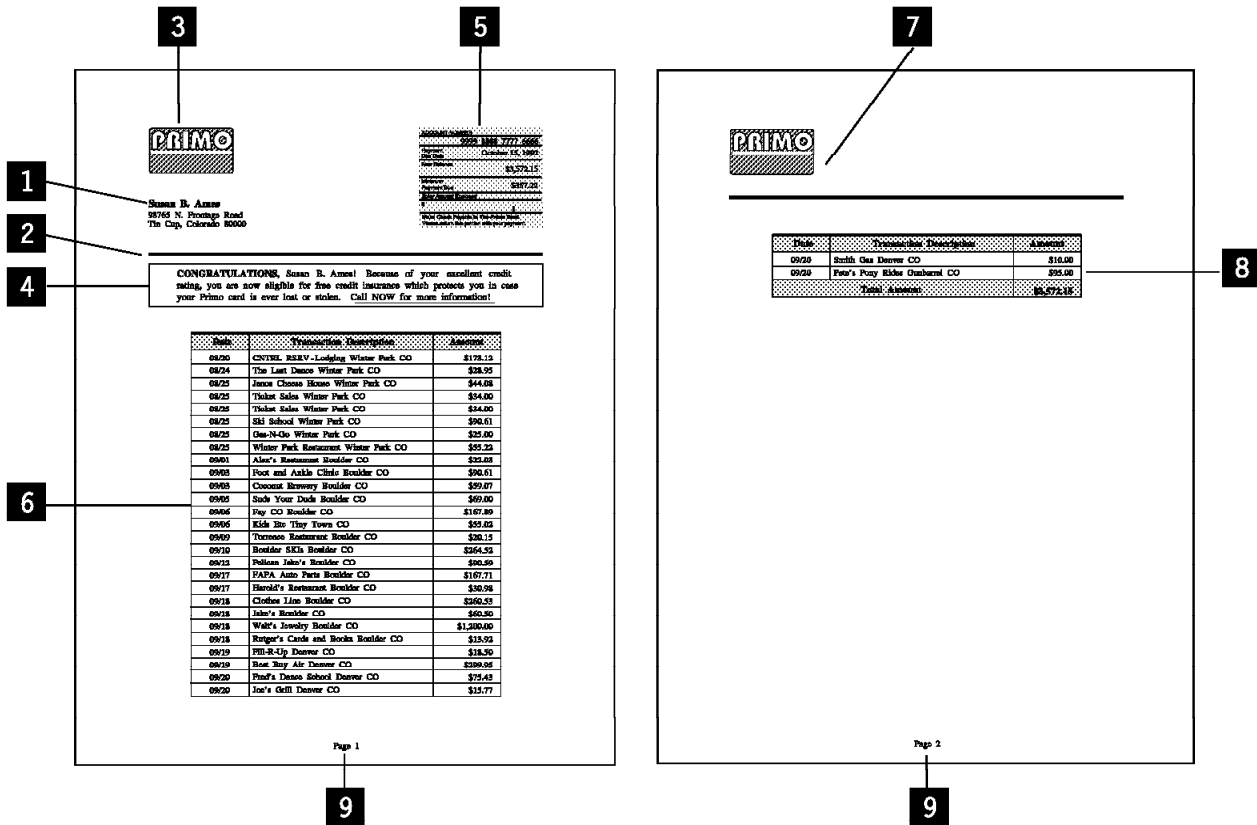
ACCOUNT NUMBER	1111 2222 3333 4444
Payment Due Date	October 15, 1992
New Balance	\$378.14
Minimum Payment Due	\$37.81
Enter Amount Enclosed	\$
Make Check Payable to <i>The Primo Card</i> . Please return this portion with your payment.	

**Lawrence M. Browning, Jr.**  
6 Lafayette Street  
Niwot, Colorado 81050

**CONGRATULATIONS,** Lawrence M. Browning, Jr.! Because of your excellent credit rating, you are now eligible for free credit insurance which protects you in case your Primo card is ever lost or stolen. Call NOW for more information!

Date	Transaction Description	Amount
08/05	Kummins Lighting Co Denver CO	\$209.24
08/12	PKG S Cincinnati OH	\$90.95
08/14	Exploratorium Store San Francisco CA	\$19.48
08/17	Shorty's Drug Store Oakland CA	\$9.25
08/17	Monty's Sporting Inc Oakland CA	\$30.26
08/18	McCrackin Hardware Boulder CO	\$18.96
<b>Total Amount</b>		<b>\$378.14</b>

Figure 7 shows where various parts of the first two pages of the sample are described.

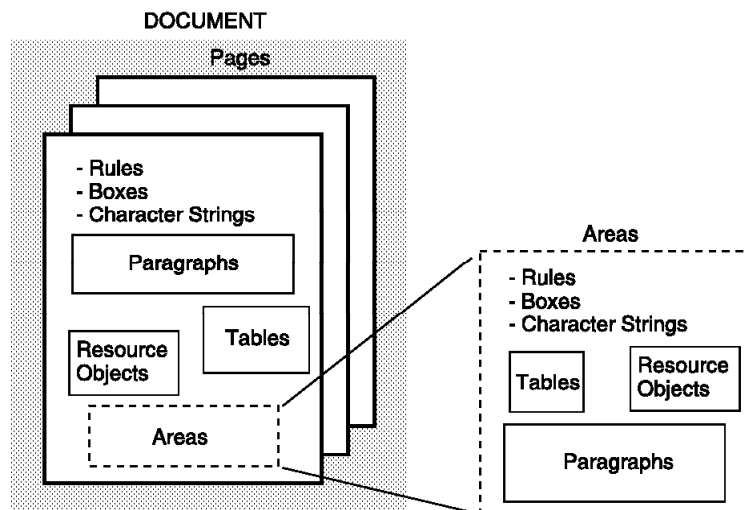


- 1 Character string. See "Character String" on page 28.
- 2 Rule. See "Rule" on page 31.
- 3 Page segment. See "Resources" on page 33.
- 4 Paragraph. See "Paragraphs" on page 36.
- 5 Overlay and Area. See "Areas" on page 42.
- 6 Table. See "Tables" on page 48.
- 7 Page segment, new page layout. See "Determining Page Breaks and Changing Page Layout" on page 82.
- 8 Continue data after page break. See "Determining Page Breaks and Changing Page Layout" on page 82.
- 9 Footer. See "Determining Page Breaks and Changing Page Layout" on page 82.

Figure 7. Sample Document

Documents contain pages, which can contain the elements shown in Figure 8, all of which are optional. “AFP Documents and Pages” on page 8 describes the components of a document in more detail.

Figure 8. Document Elements. This is a conceptual drawing; it does not reflect the layout of the sample document.



## Getting Started

The approach to this sample (or any document) is:

1. Initialize AFP API to begin the AFP API session. The AFP API session initialization *must successfully complete before you invoke any other call*.
2. If necessary, override the default environment of the AFP API session. “Setting Up and Defining the Environment for an AFP API Session” on page 65 describes calls for doing this.
3. Begin the document.
4. If necessary, define fonts. “Defining Fonts and Using Them with AFP API” on page 67 describes calls for defining fonts.
5. If necessary, define table rows and fields. “Tables” on page 48 describes calls for defining table rows and fields.
6. Begin the page.
  - a. Specify where to put data on the page.
  - b. Specify the attributes of the data.
  - c. Put data on the page. In the example, data consists of strings of text (the name and address), a horizontal rule, a paragraph, a page segment (artwork), an overlay (the shaded summary in the upper right of the sample), an area (includes the overlay), and a table. Start with the character string.
  - d. Repeat steps a through c until the page is complete.
7. End the page.

## Program Template

8. If you requested that AFP API write the output to a buffer, retrieve the output for the page. “Buffering AFP API Output” on page 66 describes calls for retrieving buffered output.
9. Begin a new page. Repeat steps 6– 8 above.
10. End the document.
11. End the AFP API session.

To help you keep track of where you are in the above procedure, AFP API has named the elements of the process. Each element (AFP API session, document, page, and so on), is called a *state*, and each state has an ID called a *handle*. “Understanding Handles” on page 77 describes handles in more detail. Don’t be concerned with states yet; they are described later in this chapter under “Understanding States” on page 75.

## Program Template

To get started in the AFP API sample, use the template shown on page 25. By inserting the code for the various parts of the sample, as described in “Putting Data on the Page” on page 26, you can print either part or all of the sample. The AFP API procedure calls shown in the sample code (for example, AFPBDOC) include all the parameters. This chapter does not describe all of the parameters; see “Format of the AFP API Procedure Call Descriptions” on page 98 for a detailed description of the parameters.

You must begin and end every document and every page within the document. The code in this template is only part of the total initialize code for the example. It does not contain calls for setting output characteristics, defining fonts, and defining table defaults. The complete code for the example is in *AFP Application Programming Interface: COBOL Language Reference* and in *AFP Application Programming Interface: PL/1 Language Reference*. Basically, the code is as follows for initializing AFP API, beginning a document, beginning a page, and then ending each:

Code	Description
<pre>SETUP-AFPAPI. CALL "AFPINIT" USING     BY REFERENCE     AFPAPI-HANDLE     BY CONTENT     FALS     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPINIT (Initialize AFP API): Establishes the AFP API session and <i>must successfully complete before you invoke any other call</i>. FALS means do not generate a trace file. AFP API returns the AFPAPI handle, which must appear in all subsequent calls to the AFP API in this session. See “Understanding Handles” on page 77 for a description of handles.</p>
<pre>MOVE 215 TO AFP-DOC-PAGE-WIDTH MOVE 280 TO AFP-DOC-PAGE-DEPTH CALL "AFPBDOC"     USING     BY CONTENT     AFPAPI-HANDLE     MM     AFP-DOC-PAGE WIDTH     AFP-DOC-PAGE DEPTH     ORIENTO     BY REFERENCE     AFP-DOCUMENT-HANDLE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPBDOC (Begin Document): Begins a document and establishes the default unit of measure, page size, and page orientation for all pages in this document. The page width is 215 mm, and the page depth is 280 mm. The begin page call can override the page size and orientation defaults. AFP API returns the document handle. AFPBDOC must include the AFP API handle returned from the AFPINIT call.</p>



Code	Description
<pre> MOVE AFP-DEFAULT TO AFP-PAGE-WIDTH MOVE AFP-DEFAULT TO AFP-PAGE-DEPTH CALL  "AFPBPAG" USING       BY CONTENT       AFP-API-HANDLE       AFP-DOCUMENT-HANDLE       AFP-PAGE WIDTH       AFP-PAGE DEPTH       ORIENTDOC       BY REFERENCE       AFP-PAGE-HANDLE       AFP-RET-CODE       AFP-SEVERITY-CODE.           </pre>	<p><b>AFPBPAG (Begin Page):</b>            Begins a page within a document and establishes the page size and orientation. If you use default values, the page inherits them from the document (AFPBDOC call). The page width is 215 mm, which is the default set in the AFPBDOC call. The page depth is 280 mm, which is the default set in the AFPBDOC call. AFP API returns a page handle. AFPBPAG must include the AFP API handle returned from the AFPINIT call and the document handle returned from the AFPBDOC call. Only one page can be open at a time.</p>
<pre> ===== (Put data on the page....) =====           </pre>	<p>See "Putting Data on the Page" on page 26.</p>
<pre> CALL  "AFPEPAG" USING       BY CONTENT       AFP-API-HANDLE       BY REFERENCE       AFP-PAGE-HANDLE       AFP-RET-CODE       AFP-SEVERITY-CODE.           </pre>	<p><b>AFPEPAG (End Page):</b>            Ends the page. AFPEPAG must include the AFP API handle returned from the AFPINIT call and the page handle returned from the AFPBPAG call. At this point, you can either begin a new page in this document or end the document.</p>
<pre> CALL  "AFPEDOC" USING       BY CONTENT       AFP-API-HANDLE       BY REFERENCE       AFP-DOCUMENT-HANDLE       AFP-RET-CODE       AFP-SEVERITY-CODE.           </pre>	<p><b>AFPEDOC (End Document):</b>            Ends the document. AFPEDOC must include the AFP API handle returned from the AFPINIT call and the document handle returned from the AFPBDOC call. At this point, you can begin a new document in this session. AFPEDOC is required before ending an AFP API session with the AFPEND call (normal end).</p>
<pre> CALL "AFPEND" USING       BY CONTENT       AFP-API-HANDLE       BY REFERENCE       AFP-RET-CODE       AFP-SEVERITY-CODE.           </pre>	<p><b>AFPEND (End AFP API):</b>            Ends the AFP API session and frees all AFP API storage. AFPEND must include the AFP API handle returned from the AFPINIT call.</p>

AFPINIT establishes a session and assigns a handle for that session. You cannot issue any other AFP API calls until you have successfully initialized AFP API.

AFPBDOC begins a document, and AFPBPAG begins a page within the document. Both AFPBDOC and AFPBPAG contain the logical page width, logical page depth, and logical page orientation parameters. If the values for these parameters are different, the values in AFPBPAG override those in AFPBDOC until an AFPEPAG is received, when the values for these parameters return to those specified in AFPBDOC. Definitions for logical page and physical page are:

**Physical Page** The top of the physical page is the short side of the paper.

**Logical Page** The top of the logical page is the side associated with the width on the AFPBDOC or AFPBPAG procedure calls.

For example, if you want to format for landscape presentation, specify either 90° or 270° orientation and set the logical page width equal to the dimension of the short side of the paper. See " AFPBDOC (Begin Document)" on page 100 for a description of logical page orientation.

AFPBDOC also specifies the unit of measure to use for the entire document (inches, millimeters, centimeters, 240ths of an inch or 1440ths of an inch), which

## Putting Data on a Page

remains in effect until overridden by the AFPSUNI (Set Units) procedure call. “Setting Attributes (and Querying Them)” on page 73 describes units of measure (AFPSUNI).

After you end all pages and end the document, end the session. You can end a session with either of two procedure calls. One is a normal termination, and one is an abnormal termination.

AFPEND (End AFP API):	Normal end. Ends the AFP API session and frees all AFP API storage.
AFPTERM (Terminate AFP API):	Abnormal end. Ends an AFP API session, frees all AFP API storage, and creates a partial page (if one exists) that you can print. This is useful for assisting you in debugging your program.

---

## Putting Data on the Page

After you initialize, begin the document, and begin a page, you can use procedure calls to put data in a page (see item 4c under “Getting Started” on page 23). Before putting data on a page, you must specify its position on the page with the AFPSPOS procedure call. You can put these types of data on a page:

- Character strings
- Rules
- Resources (art and overlays)
- Paragraphs
- Areas
- Tables
- Boxes (not shown in the example)
- Data objects (not shown in the example)

The example shown here is one document with three pages; however, you can put the code for each piece of data shown in this section into the template shown in “Program Template” on page 24 and print just that piece of data. The example is intended to illustrate concepts and should not interfere with your understanding of the complete code for the example.

The example code on the pages that follow produces the output shown, if you use the copy files for variables and constants and the other files that are shipped with the product. These files are listed in Figure 9 on page 27. Refer to *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference* for a description of the copy files for COBOL and PL/1.

APQSAMP	COBOL source (performs)
APQSAMP2	COBOL source (calls)
APQCISMP	COBOL source for sample program (CICS)
APQCISMB	COBOL source for sample program (CICS, buffered output)
APQRCS	COBOL return codes
APQCONST	COBOL constants
APQPERF	COBOL performs
APQVARS	COBOL variables
APQTRIM	COBOL trim subprogram
APQSTRL	COBOL string length subprogram
APQPSAMP	PL/1 source (performs)
APQPSMP2	PL/1 source (calls)
APQPRCS	PL/1 return codes
APQPCON	PL/1 constants
APQPPRF	PL/1 performs
APQPVAR	PL/1 variables
APQDATA	Data file for APQSAMP, APQSAMP2, APQPSAMP, and APQPSMP2
APQCOCOB	JCL to compile & link COBOL
APQCOPLI	JCL to compile & link PL/1
APQIVCOB	JCL to run APQSAMP
APQIVPLI	JCL to run APQPSAMP
APQCOSMB	JCL to translate, compile & link APQCISMB (CICS)
APQCOSMP	JCL to translate, compile & link APQCISMP (CICS)
APQCIFON	JCL to copy font PDS into VSAM data set (CICS)
APQCISEG	JCL to copy page segment PDS into VSAM data set (CICS)
APQPSEG	Page segment (PRIMO artwork)
O1APQL2	Overlay (shaded summary box)
IOCAMMR	Image object used for describing AFPIOBJ (Include Object)

**Note:** IOCAMMR is shipped with PSF but not on the AFP API distribution tape.

Figure 9. Files Shipped with AFP API

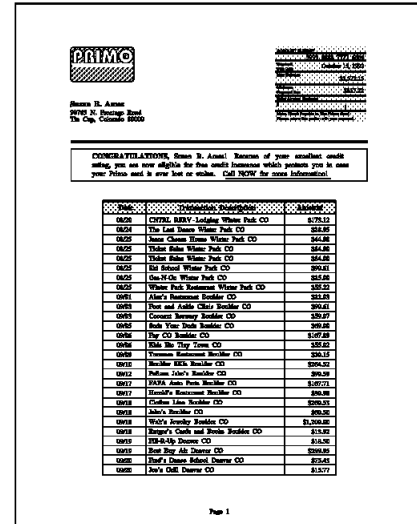
# Put Character String

## Character String

A character string is printed exactly as entered; that is, with no formatting, except to align it right, left, center, or at a character. The example has three character strings in the name and address, and all are aligned left. The example, however, shows code only for Susan’s name. Code for the street address and the city are similar, except for the font used and the position on the page.

The steps for placing this string on the page are:

1. Specify the font to use for Susan’s name.
2. Specify where to put Susan’s name.
3. Put Susan’s name on the page.



The character strings in the sample look like this:

**Susan B. Ames**  
 98765 N. Frontage Road  
 Tin Cup, Colorado 80000

Figure 10. Character String

For Susan’s name only, here is the code to put in the template:

Code	Description
*-----*	*
* Write the customer name.	*
*-----*	*
CALL "AFPSFNT" USING	AFPSFNT (Set Font):
BY CONTENT	Specifies a font for Susan’s name. Prior to specifying
AFPAPI-HANDLE	a font, you must define it to AFP API. Do this with a
TIM12BOLD	AFPDEFNT (Define Font) procedure call. "Defining
AFP-PAGE-HANDLE	Fonts and Using Them with AFP API" on page 67
BY REFERENCE	contains more detail.
AFP-RET-CODE	
AFP-SEVERITY-CODE.	

Code	Description
<pre> MOVE 29 TO AFP-X-COORDINATE. MOVE 56 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING   BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-X-COORDINATE     XABS     AFP-Y-COORDINATE     YABS   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>AFPSPOS (Set Position):</p> <p>Specifies the position for placing the baseline of the character string on the page in the current unit of measure. In this case, the current unit of measure is millimeters, which was established in the AFPBDOC call. Susan's name is at X-coordinate 29 and Y-coordinate 56. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>
<pre> CALL "TRIM" USING CUST-NAME,   BY CONTENT LENGTH OF CUST-NAME,   BY REFERENCE AFP-CHARACTER-STRING,   AFP-STRING-LENGTH. </pre>	<p>TRIM:</p> <p>Uses Susan's name in the CUST-NAME identifier, strips off unnecessary leading and trailing blanks in the data for correct formatting, counts the characters (length) of CUST-NAME, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>

## Put Character String

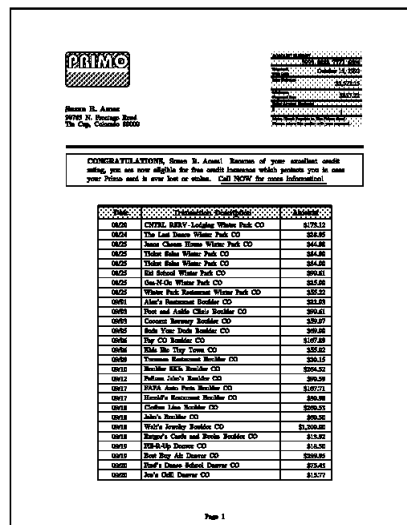
Code	Description
<pre>CALL "AFPPCHS" USING   BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-STRING-LENGTH     AFP-CHARACTER-STRING     L-FT     AFP-ALIGNMENT-CHAR     FALS     FALS   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPPCHS (Put Character String):          Puts a character string (Susan's name) on the page and aligns the left character (S) in the string at the current position. AFPPCHS can put a character string in a page, in an area, and in the field of a table with the following alignments relative to the current position:</p> <p><b>Left</b>      First character begins at the current position.</p> <pre>Current position =                      A B C                    A B C</pre> <p><b>Right</b>     Last character ends at the current position.</p> <pre>Current position =                      A B C                    A B C</pre> <p><b>Center</b>    Characters are centered around the current position.</p> <pre>Current position =                      A B C                    A B C</pre> <p><b>Character</b> Specified character (for example, a decimal point) is placed at the current position.</p> <pre>Current position =                      A B C.                    .A B C</pre> <p>The current values for intercharacter spacing, word spacing, color, and font apply, and you can underline the string. See "Setting Attributes (and Querying Them)" on page 73 for a description of these concepts.</p> <p>FALS (first one) means move the current position to the end of the string.</p> <p>FALS (second one) means don't underline the string.</p>

The code is similar for the street address and city and state, except for using a different font and a different position. Refer to *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference* for the code.

## Rule

You can draw vertical and horizontal rules and can specify their position, length, and thickness. Here are the steps for placing this rule on the page:

1. Set the rule thickness.
2. Set the position for where the rule begins.
3. Draw the rule from the specified position, in the specified direction, at the specified thickness, for the specified length.



The rule in the sample looks like this:

Figure 11. Sample Rule

For the rule only, here is the code to put in the template:

Code	Description
*-----*	
* Draw a rule underneath the address *	
*-----*	
<pre> MOVE 1.5 TO AFP-RULE-THICKNESS. CALL "AFPSRTH" USING     BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-RULE-THICKNESS     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.         </pre>	<p>AFPSRTH (Set Rule Thickness): Specifies the thickness for the rule in the unit of measure currently in effect, in this case, 1.5 millimeters.</p>
<pre> MOVE 29 TO AFP-X-COORDINATE. MOVE 73 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-X-COORDINATE     XABS     AFP-Y-COORDINATE     YABS     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.         </pre>	<p>AFPSPOS (Set Position): Sets the position for placing the top-left corner of the rule at 29 millimeters in the X-direction and 73 millimeters in the Y-direction, relative to the logical page origin. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>

## Put Rule

Code	Description
<pre>MOVE 158 TO AFP-RULE-LENGTH. CALL "AFPPRUL" USING   BY CONTENT     AFPAPI-HANDLE     AFP-PAGE-HANDLE     XDIRECTION     AFP-RULE-LENGTH   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPPRUL (Put Rule):</p> <p>Draws a rule 158 millimeters long across the page from the current position. The current values for rule thickness and color apply. See "AFPSCLR (Set Color)" on page 181 for specifying color. The rule thickness extends below (for a horizontal rule) or to the right (for a vertical rule) of the rule. The current position is unchanged after the rule is drawn. That is, X is 29 millimeters, and Y is 73 millimeters.</p>

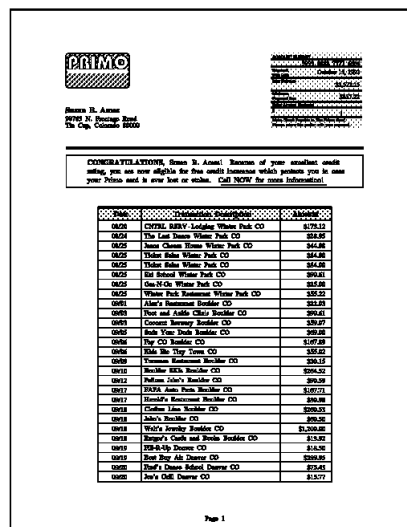


## Resources

You can put art on a page at a specified position by using a resource called a page segment, which is created by using another application program. "More About Resources" on page 34 describes other resources and their use.

Here are the steps for placing this page segment on the page:

1. Set the position for where to place the upper-left corner of the page segment.
2. Put the page segment at the specified position.



The page segment in the example looks like this:



Figure 12. Page Segment (Art)

For the page segment only, here is the code to put in the template:

Code	Description
*-----*	
* Include the Page Segment	
*-----*	
<pre> MOVE 29 TO AFP-X-COORDINATE. MOVE 23 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-X-COORDINATE     XABS     AFP-Y-COORDINATE     YABS     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.                     </pre>	<p>AFPSPOS (Set Position):</p> <p>Sets the position for placing the page segment at 29 millimeters in the X-direction and 23 millimeters in the Y-direction, relative to the logical page origin. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>

Code	Description
<pre>MOVE "APQPSEG" TO AFP-PSEG-NAME. CALL "AFPIPSG" USING   BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-PSEG-NAME     TRU     FALS   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPIPSG (Include Page Segment):</p> <p>Gets the page segment (art), called APQPSEG, from the page segment library and brings it inline at the current position specified in the AFPSPPOS call. "More About Resources" on page 34 has more information about including page segments, objects, and overlays in documents. The current position is unchanged after the page segment is drawn. That is, the position is X=29 millimeters and Y=23 millimeters.</p>

### More About Resources

As described in "Data Objects and AFP Resource Objects" on page 11, resources are objects that are created with other applications and placed in a library. AFP API must be able to access these libraries.<sup>2</sup> AFP API can either reference a resource that the printer later integrates with the AFP API output or retrieve the resource and integrate it with the output pages produced by AFP API. The example uses three resources:

- The *PRIMO* logo in the upper-left corner of the first page is a page segment (artwork) called APQPSEG. The sample code above shows how to include this page segment in a document.
- The shaded summary box in the upper-right corner of the first page uses an overlay. The overlay is included in an area, which is described in "Areas" on page 42.
- The fonts are used throughout the document.

Resources also include form definitions, images, and graphics. All resources must be available to AFP API, the print server, or both. The approach in the code for including overlays and objects is similar to the approach used for including page segments. The approach to referencing form definitions is described below under AFPINVM.

You'll get an error at print time but not necessarily during AFP API execution for either of the following conditions:

- If you reference a resource that doesn't exist or is in a resource library that isn't available to AFP API
- If AFPSPPOS places an overlay or page segment at a valid position on the page, but the object runs off the page

---

<sup>2</sup> On the VM operating system, AFP API searches in alphabetical order for fonts, page segments, and included objects on the first minidisk on which the resource is stored. For example, AFP API searches for a page segment named RUFUS as RUFUS PSEG3820 \*.

You can use, for example, the following licensed programs to create these resources:

- **Overlays**
  - Overlay Generation Language/370 (OGL/370)
  - PSF/2 and IBM AFPDS Windows Driver
  - FormsDesigner by ISIS Information Systems, Inc.
  - ElixirForm by Elixir Technologies Corporation
- **Form definitions**
  - Page Printer Formatting Aid/370 (PPFA/370)
  - Application Builder for AFP by Elixir Technologies Corporation
  - FormsDesigner by ISIS Information Systems, Inc.
- **Page segments, images, and graphics**
  - Graphical Data Display Manager (GDDM)
  - PSF/2 and IBM AFPDS Windows Driver
  - AFP Workbench for Windows (the clipping function)
  - AFP Workbench for Windows and IBM AFPDS Windows Driver
  - ElixirImage by Elixir Technologies Corporation
  - FormsDesigner by ISIS Information Systems, Inc.

Four other procedure calls involve resources. See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions for the following procedure calls.

AFPINVM (Invoke Medium Map):	References a medium map (also called a copy group) from the form definition being used for printing the AFP API output. You can use this procedure call in the document between pages; it is valid only in document state and always forces a new physical page. You can use AFPINVM to change medium overlays, to switch bins, and to control duplexing options.
AFPIOBJ (Include Object):	Gets and puts an image or graphic object either in a page or area.
AFPIOVL (Include Page Overlay):	References an overlay either in a page or area.
AFPIPSG (Include Page Segment):	Either gets a page segment from the library specified in the AFPSLIB call and puts it inline in either a page or area, or references the page segment for the printer to include in the document at print time.

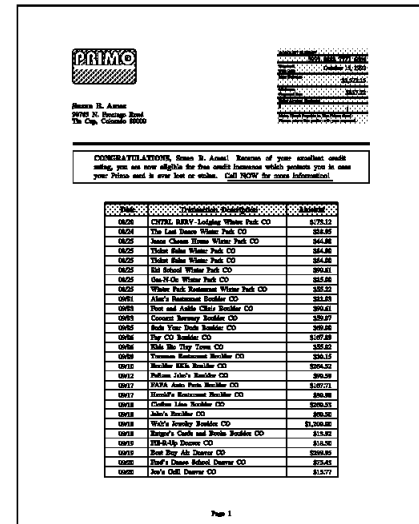
# Paragraphs

## Paragraphs

You put paragraphs in a page or area, and you can control the layout and appearance of that paragraph.

Here are the steps for placing a paragraph on the page:

1. Set the position for where to place the top-left corner of the paragraph.
2. Set the rule thickness.
3. Begin the paragraph.
4. Set the font you want (you'll do this several times).
5. Put text in the paragraph (you'll do this several times).
6. End the paragraph.



The paragraph in the example looks like this:

**CONGRATULATIONS, Susan B. Ames! Because of your excellent credit rating, you are now eligible for free credit insurance which protects you in case your Primo card is ever lost or stolen. Call NOW for more information!**

Figure 13. Paragraph

For the paragraph only, here is the code to put in the template:

Code	Description
*-----*	
* PROCESS THE PARAGRAPH. *	
*-----*	
PROCESS-THE-PARAGRAPH.	
MOVE 29 TO AFP-X-COORDINATE.	
MOVE PARAGRAPH-WHITE-SPACE TO AFP-Y-COORDINATE.	
CALL "AFPSPOS" USING	
BY CONTENT	AFPSPOS (Set Position):
AFP-API-HANDLE	Sets the position for placing the paragraph at
AFP-PAGE-HANDLE	29 millimeters in the X-direction and after the
AFP-X-COORDINATE	designated white space in the Y-direction. The Data
XABS	Division in the COBOL program contains the value for
AFP-Y-COORDINATE	PARAGRAPH-WHITE-SPACE. XABS and YREL mean
YREL	the X-absolute and Y-relative reference coordinate
BY REFERENCE	systems, respectively. See "Setting Attributes (and
AFP-RET-CODE	Querying Them)" on page 73 for a description of
AFP-SEVERITY-CODE.	absolute and relative coordinate systems.

Code	Description
<pre> MOVE 0.5 TO AFP-RULE-THICKNESS. CALL "AFPSRTH" USING     BY CONTENT         AFP-API-HANDLE         AFP-PAGE-HANDLE         AFP-RULE-THICKNESS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	<p>AFPSRTH (Set Rule Thickness): Sets the thickness for the rule (frame) around the paragraph and overrides any rule thickness set earlier.</p>
<pre> MOVE 0 TO AFP-FIRST-LINE-INDENT. MOVE AFP-DEFAULT TO AFP-FIRST-LINE-OFFSET. MOVE 10.0 TO AFP-LEFT-MARGIN. MOVE 145.0 TO AFP-LINE-LENGTH. MOVE AFP-DEFAULT TO AFP-LINE-SPACING. MOVE 158.0 TO AFP-RT-RULE-OFFSET. MOVE 0.0 TO AFP-BOT-RULE-OFFSET. MOVE 0 TO AFP-SHADING-INTENSITY. CALL "AFBPAR" USING     BY CONTENT         AFP-API-HANDLE         AFP-PAGE-HANDLE         AFP-FIRST-LINE-INDENT         FOJUSTIFY         AFP-FIRST-LINE-OFFSET         AFP-LEFT-MARGIN         AFP-LINE-LENGTH         AFP-LINE-SPACING         TRU         AFP-RT-RULE-OFFSET         AFP-BOT-RULE-OFFSET         NOSHADE         AFP-SHADING-INTENSITY     BY REFERENCE         AFP-PARAGRAPH-HANDLE         AFP-RET-CODE         AFP-SEVERITY-CODE. CALL "AFPSFNT" USING     BY CONTENT         AFP-API-HANDLE         AFP-PARAGRAPH-HANDLE         TIM12BOLD     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	<p>AFBPAR (Begin Paragraph): Begins a paragraph at the current position. With the parameters in this call, you can specify the first-line offset, left margin, line length and line spacing, the amount of indentation for the first line (0, positive, or negative) and can frame (TRU) or shade (or both) the paragraph. To shade the paragraph without a frame, use 0 for AFPSRTH just prior to the AFBPAR in page or area state, but remember to reset the AFPSRTH after ending the paragraph.</p> <p>AFPSFNT (Set Font): Specifies a font previously defined in the AFPDFNT call. You'll use several AFPSFNT calls in this example.</p>
<pre> MOVE LOW-VALUES TO AFP-CHARACTER-STRING. STRING "CONGRATULATIONS, " DELIMITED BY SIZE     INTO AFP-CHARACTER-STRING. CALL "STRING-LENGTH" USING AFP-CHARACTER-STRING,     BY CONTENT LENGTH OF AFP-CHARACTER-STRING,     BY REFERENCE AFP-STRING-LENGTH. </pre>	<p>LOW-VALUES is a reserved COBOL word. It has the lowest ordinal position in the collating sequence. Moving it to AFP-CHARACTER-STRING removes all residual data from any previous AFP-CHARACTER-STRING.</p> <p>STRING-LENGTH determines the length of a character string and puts the length in AFP-CHARACTER-STRING. APQSTRL is the name of the copy book that contains the string-length subprogram that is shipped with the example code.</p>
<pre> CALL "AFPPTXT" USING     BY CONTENT         AFP-API-HANDLE         AFP-PARAGRAPH-HANDLE         AFP-STRING-LENGTH         AFP-CHARACTER-STRING         TRU         FALS     BY REFERENCE         AFP-REMAINING-LENGTH         AFP-REMAINING-STRING         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	<p>AFPPTXT (Put Text): Places text in the paragraph and underscores it, if you want. Here, TRU means concatenate (format) lines of text, and FALS means no underscore. You can use multiple AFPPTXT calls in a paragraph, and the program merges and formats the lines to fit the space, as shown in the two paragraphs in "More About Paragraphs" on page 40 for Susan and Lawrence. If the text exceeds the page or area depth, AFP API returns a WARNING severity code and returns the overflow characters to the program. This example contains several AFPPTXT calls.</p>

## Paragraphs

Code	Description
<pre>CALL "TRIM" USING CUST-NAME,                   BY CONTENT LENGTH OF CUST-NAME,                   BY REFERENCE AFP-CHARACTER-STRING,                   AFP-STRING-LENGTH.</pre>	<p><b>TRIM:</b> Uses Susan's name in the CUST-NAME identifier, strips off unnecessary leading and trailing blanks in the data for correct formatting, counts the characters (length) of CUST-NAME, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>
<pre>CALL "AFPSFNT" USING                   BY CONTENT                   AFPAPI-HANDLE                   AFP-PARAGRAPH-HANDLE                   TIM12MED                   BY REFERENCE                   AFP-RET-CODE                   AFP-SEVERITY-CODE.</pre>	<p><b>AFPSFNT (Set Font):</b> Specifies a font defined in the AFPDFNT call. In this case, it is the font for Susan's name and the rest of the text in the paragraph.</p>
<pre>CALL "AFPPTXT" USING                   BY CONTENT                   AFPAPI-HANDLE                   AFP-PARAGRAPH-HANDLE                   AFP-STRING-LENGTH                   AFP-CHARACTER-STRING                   TRU                   FALS                   BY REFERENCE                   AFP-REMAINING-LENGTH                   AFP-REMAINING-STRING                   AFP-RET-CODE                   AFP-SEVERITY-CODE.</pre>	<p><b>AFPPTXT (Put Text):</b> Writes Susan's name in the paragraph. CUST-NAME in the TRIM call identifies the character string (Susan) to be printed.</p>
<pre>MOVE LOW-VALUES TO AFP-CHARACTER-STRING. STRING "! Because of your excellent credit rating, you are - " now eligible for free credit insurance which"   DELIMITED BY SIZE INTO AFP-CHARACTER-STRING. CALL "STRING-LENGTH" USING AFP-CHARACTER-STRING,   BY CONTENT LENGTH OF AFP-CHARACTER-STRING,   BY REFERENCE AFP-STRING-LENGTH.</pre>	<p><b>LOW-VALUES</b> is a reserved COBOL word. It has the lowest ordinal position in the collating sequence. Moving it to AFP-CHARACTER-STRING removes all residual data from any previous AFP-CHARACTER-STRING.</p> <p><b>STRING-LENGTH</b> determines the length of a character string and puts the length in AFP-CHARACTER-STRING. APQSTRL is the name of the copy book that contains the string-length subprogram shipped with the example code.</p>
<pre>CALL "AFPPTXT" USING                   BY CONTENT                   AFPAPI-HANDLE                   AFP-PARAGRAPH-HANDLE                   AFP-STRING-LENGTH                   AFP-CHARACTER-STRING                   TRU                   FALS                   BY REFERENCE                   AFP-REMAINING-LENGTH                   AFP-REMAINING-STRING                   AFP-RET-CODE                   AFP-SEVERITY-CODE.</pre>	<p><b>AFPPTXT (Put Text):</b> Writes the next text string in the paragraph.</p>

Code	Description
<pre>MOVE LOW-VALUES TO AFP-CHARACTER-STRING. STRING " protects you in case your Primo card is ever lost "or stolen. " - DELIMITED BY SIZE INTO AFP-CHARACTER-STRING. CALL "STRING-LENGTH" USING AFP-CHARACTER-STRING,     BY CONTENT LENGTH OF AFP-CHARACTER-STRING,     BY REFERENCE AFP-STRING-LENGTH.</pre>	<p>LOW-VALUES is a reserved COBOL word. It has the lowest ordinal position in the collating sequence. Moving it to AFP-CHARACTER-STRING removes all residual data from any previous AFP-CHARACTER-STRING.</p> <p>STRING-LENGTH determines the length of a character string and puts the length in AFP-CHARACTER-STRING. APQSTRL is the name of the copy book that contains the string-length subprogram that is shipped with the example code.</p>
<pre>CALL "AFPPTXT" USING     BY CONTENT         AFP-API-HANDLE         AFP-PARAGRAPH-HANDLE         AFP-STRING-LENGTH         AFP-CHARACTER-STRING         TRU         FALS     BY REFERENCE         AFP-REMAINING-LENGTH         AFP-REMAINING-STRING         AFP-RET-CODE         AFP-SEVERITY-CODE.</pre>	<p>AFPPTXT (Put Text): Writes the next text string in the paragraph.</p>
<pre>MOVE LOW-VALUES TO AFP-CHARACTER-STRING. STRING " Call NOW for more information!" DELIMITED BY SIZE INTO AFP-CHARACTER-STRING. CALL "STRING-LENGTH" USING AFP-CHARACTER-STRING,     BY CONTENT LENGTH OF AFP-CHARACTER-STRING,     BY REFERENCE AFP-STRING-LENGTH.</pre>	<p>LOW-VALUES is a reserved COBOL word. It has the lowest ordinal position in the collating sequence. Moving it to AFP-CHARACTER-STRING removes all residual data from any previous AFP-CHARACTER-STRING.</p> <p>STRING-LENGTH determines the length of a character string and puts the length in AFP-CHARACTER-STRING. APQSTRL is the name of the copy book that contains the string-length subprogram that is shipped with the example code.</p>
<pre>CALL "AFPPTXT" USING     BY CONTENT         AFP-API-HANDLE         AFP-PARAGRAPH-HANDLE         AFP-STRING-LENGTH         AFP-CHARACTER-STRING         TRU         TRU     BY REFERENCE         AFP-REMAINING-LENGTH         AFP-REMAINING-STRING         AFP-RET-CODE         AFP-SEVERITY-CODE.</pre>	<p>AFPPTXT (Put Text): Writes the last text string in the paragraph. The first TRU means concatenate the lines and the second TRU means underline the text string.</p>
<pre>CALL "AFPEPAR" USING     BY CONTENT         AFP-API-HANDLE     BY REFERENCE         AFP-PARAGRAPH-HANDLE         AFP-PARAGRAPH-DEPTH         AFP-RET-CODE         AFP-SEVERITY-CODE.</pre>	<p>AFPEPAR (End Paragraph): Ends the paragraph, returns to the program the final depth of the paragraph in the units of measure currently in effect, resets the font to what it was before the AFPBPAP call, and establishes the current position as the bottom-left corner of the paragraph.</p>

### More About Paragraphs

While in a paragraph, you can set fonts (AFPSFNT), put text (AFPPTXT), set color (AFPSCLR), control intercharacter spacing (AFPSICS), and set word spacing (AFPSWSP). Attributes specified for a paragraph override those specified in the page or area in which that paragraph appears, until an AFPEPAR is issued.

The most significant feature of a paragraph is that text flows within it. That is, words and sentences flow according to the formatting parameters set in the AFPBPAR call, regardless of how the words and sentences were entered on the AFPPTXT call. For example, here are two versions of the same paragraph with only the name changed:

A paragraph formatted with Susan's name. Notice where each line breaks. Also notice that the paragraph uses two different fonts.

**CONGRATULATIONS**, Susan B. Ames! Because of your excellent credit rating, you are now eligible for free credit insurance, which protects you in case your Primo card is ever lost or stolen. Call NOW for more information!

The same paragraph formatted with Lawrence's name. Notice where each line breaks, and that the breaks are different than for Susan's paragraph above. Both were entered exactly the same way, but the power of AFP API caused the text to "flow" into the allotted space.

**CONGRATULATIONS**, Lawrence M. Browning, Jr! Because of your excellent credit rating, you are now eligible for free credit insurance, which protects you in case your Primo card is ever lost or stolen. Call NOW for more information!

The four formatting options you can specify look like this:

Ragged Right

Paragraphs are wonderful  
because you can change them  
in so many ways.

Ragged Left

Paragraphs are wonderful  
because you can change them  
in so many ways.

Center

Paragraphs are wonderful  
because you can change them  
in so many ways.

Justify

Paragraphs are wonderful  
because you can change  
them in so many ways.



Because of this text-formatting ability of the paragraph, the AFPPCHS (Put Character String) call is not allowed in a paragraph. Remember that an AFPPCHS is printed exactly as entered, with no formatting. The example doesn't use all the allowable procedure calls. Other procedure calls that are permitted in a paragraph are as follows. (See "Format of the AFP API Procedure Call Descriptions" on page 98 for detailed parameter descriptions.)

AFPSCLR (Set Color)	Specifies the color for the text.
AFPSICS (Set Intercharacter Spacing)	Specifies additional spacing between characters. The font determines the original spacing between characters.
AFPSWSP (Set Word Spacing)	Specifies the spacing between words.

# Areas

## Areas

Areas are great for specifying data that you want to reuse in a document or for constructing variable portions of a page, as described in “Determining Page Breaks and Changing Page Layout” on page 82. Create the area, store it, then call it in where you want it—several times on the same page or on different pages. Having the data already formatted saves time in reformatting the same data every time you use it.

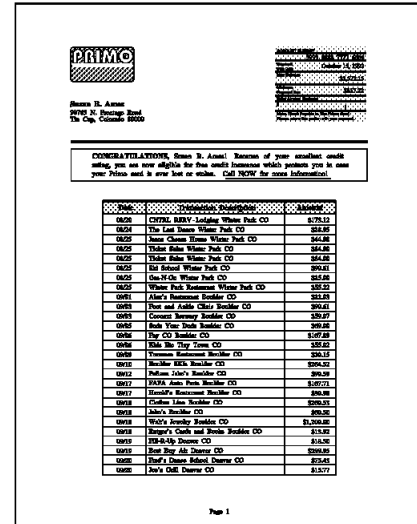
In areas, you can set attributes, define and set fonts, put rules and boxes, write paragraphs, build tables, and invoke resources and objects. See “More About Areas” on page 47 for additional information about areas.

Here are the steps for creating this area and placing it on the page:

1. Create the area.
2. Include the overlay.
3. Set the position for where to place the character string (account number) relative to the area origin.
4. Place the character string.
5. Repeat the previous two steps until all of the data is placed.
6. End the area.
7. Set the position for where to place the area on the page.
8. Place the area.
9. Destroy the area if it is no longer needed.

The area in the example contains an overlay and several character strings and looks like this:

Figure 14. Area Containing an Overlay. You can place the area anywhere on the page by changing only the AFPSPoS call.



ACCOUNT NUMBER	
9999	8888 7777 6666
Payment Due Date	October 15, 1992
New Balance	\$3,572.15
Minimum Payment Due	\$357.22
Enter Amount Enclosed	\$
Make Check Payable to <b>The Prime Card</b> . Please return this portion with your payment.	

For the area only, here is the code to put in the template:

Code	Description
<pre> *-----* *   PROCESS THE AREA.   * *-----*  PROCESS-THE-AREA.   MOVE 50.0 TO AFP-AREA-WIDTH.   MOVE 65.0 TO AFP-MAX-AREA-DEPTH.   MOVE 0 TO AFP-SHADING-INTENSITY.   CALL "AFPCARE" USING     BY CONTENT       AFP-API-HANDLE       AFP-PAGE-HANDLE       AFP-AREA-WIDTH       AFP-MAX-AREA-DEPTH       AFP-AREA-FRAME       NOSHADE       AFP-SHADING-INTENSITY     BY REFERENCE       AFP-AREA-HANDLE       AFP-RET-CODE       AFP-SEVERITY-CODE.  *-----* * Include the Page Overlay * *-----*    MOVE "O1APQL2" TO AFP-OVLY-NAME.   CALL "AFPIOVL" USING     BY CONTENT       AFP-API-HANDLE       AFP-AREA-HANDLE       AFP-OVLY-NAME     BY REFERENCE       AFP-RET-CODE       AFP-SEVERITY-CODE.  *-----* * Write the account number * *-----*    CALL "AFPSFNT" USING     BY CONTENT       AFP-API-HANDLE       AFP-AREA-HANDLE       TIMIOMED     BY REFERENCE       AFP-RET-CODE       AFP-SEVERITY-CODE.    MOVE 49 TO AFP-X-COORDINATE.   MOVE 7 TO AFP-Y-COORDINATE.   CALL "AFPSPOS" USING     BY CONTENT       AFP-API-HANDLE       AFP-AREA-HANDLE       AFP-X-COORDINATE       XABS       AFP-Y-COORDINATE       YABS     BY REFERENCE       AFP-RET-CODE       AFP-SEVERITY-CODE. </pre>	<p><b>AFPCARE (Create Area):</b> Creates the area to be filled with elements and returns the area handle for that area. You use this area handle to identify the area when you place data in the area with subsequent calls and when you place the area on a page with the AFPPARE call.</p> <p>With the parameters in this call, you can specify the area width as 50 millimeters, the maximum depth as 65, and a shading intensity of 0. The area in this example isn't framed.</p> <p><b>AFPIOVL (Include Page Overlay):</b> References an overlay called O1APQL2 for use in the area. Notice that because AFPSPOS wasn't issued, AFP API placed the overlay at the area origin.</p> <p><b>AFPSFNT (Set Font):</b> Specifies a font defined in the AFPDFNT call. This example uses several AFPSFNT calls.</p> <p><b>AFPSPOS (Set Position):</b> Sets the position for placing the account number at 49 millimeters in the X-direction and 7 millimeters in the Y-direction, relative to the area origin. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>

## Areas

Code	Description
<pre> MOVE ACCOUNT-NUM-IN TO ACCOUNT-NUM-OUT. MOVE 19 TO AFP-STRING-LENGTH. MOVE ACCOUNT-NUM-OUT TO AFP-CHARACTER-STRING. CALL "AFPPCHS" USING     BY CONTENT         AFPAPI-HANDLE         AFP-AREA-HANDLE         AFP-STRING-LENGTH         AFP-CHARACTER-STRING         R-GHT         AFP-ALIGNMENT-CHAR         FALS         FALS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	
<pre> *-----* * Write the due date. *-----* </pre>	
<pre> MOVE 49 TO AFP-X-COORDINATE. MOVE 12 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT         AFPAPI-HANDLE         AFP-AREA-HANDLE         AFP-X-COORDINATE         XABS         AFP-Y-COORDINATE         YABS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	<p>AFPSPOS (Set Position): Sets the position for placing the due date on the logical page at 49 millimeters in the X-direction and 12 millimeters in the Y-direction, relative to the area origin. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>
<pre> MOVE DUE-DATE TO AFP-CHARACTER-STRING. MOVE 11 TO AFP-STRING-LENGTH. CALL "AFPPCHS" USING     BY CONTENT         AFPAPI-HANDLE         AFP-AREA-HANDLE         AFP-STRING-LENGTH         AFP-CHARACTER-STRING         R-GHT         AFP-ALIGNMENT-CHAR         FALS         FALS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	
<pre> *-----* * Write the customer balance. *-----* </pre>	
<pre> MOVE 49 TO AFP-X-COORDINATE. MOVE 19 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT         AFPAPI-HANDLE         AFP-AREA-HANDLE         AFP-X-COORDINATE         XABS         AFP-Y-COORDINATE         YABS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	

Code	Description
<pre>CALL "TRIM" USING CUSTOMER-BALANCE-OUT,                 BY CONTENT LENGTH OF CUSTOMER-BALANCE-OUT,                 BY REFERENCE AFP-CHARACTER-STRING,                 AFP-STRING-LENGTH.</pre>	<p>TRIM:</p> <p>Uses the customer balance out in the CUSTOMER-BALANCE-OUT identifier, strips off unnecessary leading and trailing blanks in the data for correct formatting, counts the characters (length) of the data, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>
<pre>CALL "AFPPCHS" USING                 BY CONTENT                 AFP-API-HANDLE                 AFP-AREA-HANDLE                 AFP-STRING-LENGTH                 AFP-CHARACTER-STRING                 R-GHT                 AFP-ALIGNMENT-CHAR                 FALS                 FALS                 BY REFERENCE                 AFP-RET-CODE                 AFP-SEVERITY-CODE.</pre>	
<pre>*-----* * Write the customer payment. *-----*</pre>	
<pre>MOVE 49 TO AFP-X-COORDINATE. MOVE 24 TO AFP-Y-COORDINATE. CALL "AFSPPOS" USING                 BY CONTENT                 AFP-API-HANDLE                 AFP-AREA-HANDLE                 AFP-X-COORDINATE                 XABS                 AFP-Y-COORDINATE                 YABS                 BY REFERENCE                 AFP-RET-CODE                 AFP-SEVERITY-CODE.</pre>	
<pre>MULTIPLY .1 BY CUSTOMER-BALANCE-IN GIVING MIN-AMOUNT-DUE-COMP ROUNDED. MOVE MIN-AMOUNT-DUE-COMP TO MIN-AMOUNT-DUE-OUT. CALL "TRIM" USING MIN-AMOUNT-DUE-OUT,                 BY CONTENT LENGTH OF MIN-AMOUNT-DUE-OUT,                 BY REFERENCE AFP-CHARACTER-STRING,                 AFP-STRING-LENGTH.</pre>	<p>TRIM:</p> <p>Uses the minimum amount due out in the MIN-AMOUNT-DUE-OUT identifier, strips off unnecessary leading and trailing blanks in the data for correct formatting, counts the characters (length) of the data, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>
<pre>CALL "AFPPCHS" USING                 BY CONTENT                 AFP-API-HANDLE                 AFP-AREA-HANDLE                 AFP-STRING-LENGTH                 AFP-CHARACTER-STRING                 R-GHT                 AFP-ALIGNMENT-CHAR                 FALS                 FALS                 BY REFERENCE                 AFP-RET-CODE                 AFP-SEVERITY-CODE.</pre>	

## Areas

Code	Description
<pre>CALL "AFPEARE" USING   BY CONTENT   AFP-API-HANDLE   AFP-AREA-HANDLE   BY REFERENCE   AFP-AREA-DEPTH   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	<p><b>AFPEARE (End Area):</b> Ends the area and returns the actual area depth in the unit of measure currently in effect.</p>
<pre>*-----* * Place the area on the page. *-----* MOVE 137 TO AFP-X-COORDINATE. MOVE 23 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING   BY CONTENT   AFP-API-HANDLE   AFP-PAGE-HANDLE   AFP-X-COORDINATE   XABS   AFP-Y-COORDINATE   YABS   BY REFERENCE   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	<p><b>AFPSPOS (Set Position):</b> Sets the position for placing the top-left corner of the area on the logical page at 137 millimeters in the X-direction and 23 millimeters in the Y-direction, relative to the logical page origin. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>
<pre>CALL "AFPPARE" USING   BY CONTENT   AFP-API-HANDLE   AFP-PAGE-HANDLE   AFP-AREA-HANDLE   ORIENTO   BY REFERENCE   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	<p><b>AFPPARE (Put Area):</b> Puts the area onto the page at the position specified in the AFPSPOS procedure call.  You can rotate the area around the current position; however, page segments and overlays in the area will not rotate. AFPPARE is valid only in page state.</p>
<pre>*-----* * Destroy the area from AFP API storage. *-----* CALL "AFPXARE" USING   BY CONTENT   AFP-API-HANDLE   AFP-AREA-HANDLE   BY REFERENCE   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	<p><b>AFPXARE (Destroy Area):</b> Deletes the area contents from storage to free storage when the area is no longer needed. AFPXARE is valid only in document state or page state.</p>

## More About Areas

Attributes, including fonts, specified in the area override attributes specified in the document or page but do not affect the attributes of the document or page. All the AFP API elements shown in Figure 15 are valid in an area.

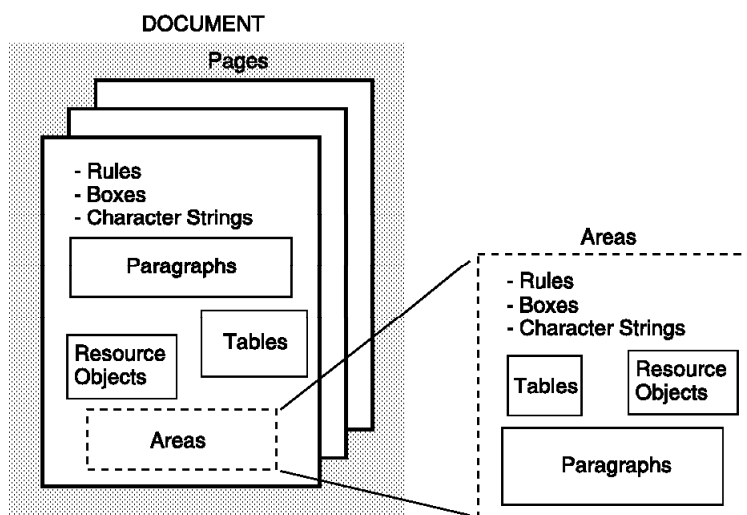


Figure 15. Document Elements

Data is placed in an area relative to the area origin. When placing data within an area, you must place the data in a top to bottom order. After being built, the formatted data remains in storage until used.

**Note:** Vertical rules that are part of an area cannot be enclosed with an area frame, because the current position is unchanged when a vertical rule is drawn. See “AFPPARE (Put Area)” on page 162 and “AFPSPOS (Set Position)” on page 193 for placing an area and for the current position after an area is drawn. You can place an area on a page with the AFPPARE procedure call.

See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions for the procedure calls described in this section, and see “Setting Attributes (and Querying Them)” on page 73 for more information about placing data on a page with the AFPSPOS procedure call.

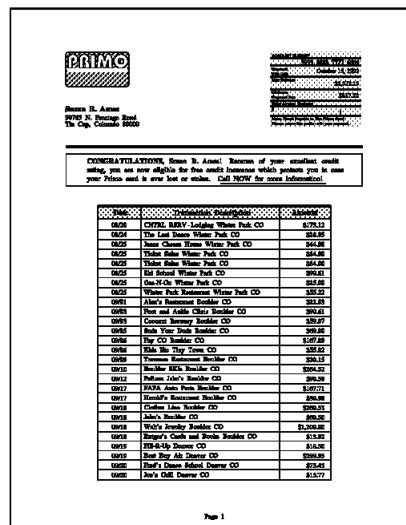
### Tables

The details of Susan’s statement are in a table. The following will be provided:

- Descriptive information about tables
- Definition of terms
- Code for the first, shaded row of the table
- Tips on reading the source code for the other rows in the table

Here are the steps for defining the table and placing it on the page:

1. Begin the document (AFPBDOC).
2. Define the format of the fields for the row (AFPDFLD).
3. Define the format of the row (AFPDROW).
4. Repeat steps 2 and 3 until all unique rows are defined.
5. Begin the page (AFPBPAG).
6. Begin the table (AFPBTBL).
7. Begin the row (AFPBROW).
8. Begin the field (AFPBFLD).
9. Put the data in the field (AFPPCHS).
10. End the field (AFPEFLD).
11. Repeat steps 8, 9, and 10 until all fields in the row are filled.
12. End the row (AFPEROW).
13. Repeat steps 7–12 until all rows in the table are complete or until the table is full. See “Determining Page Breaks and Changing Page Layout” on page 82 for determining when the table is full.
14. End the table (AFPETBL).
15. End the page (AFPEPAG).
- ⋮
16. End the document (AFPEDOC).



The first row of the table in the example looks like this:

Date	Transaction Description	Amount
------	-------------------------	--------

Figure 16. The Header Row of the Table

Tables can present information in a logical manner. By understanding some basic principles and with some planning, you can create tables of any complexity.

A table has one or more rows; each row consists of one or more columns, and each column consists of one or more subrows. AFP API constructs tables with three types of procedure calls: table, row, and field. This chapter will define the procedure calls and then describe how they fit together to form a table.



This is the basic, simplest table. It has one row (horizontal), one column (vertical), one subrow, and no text:

--

This is a table with one row, one subrow, two columns, and two fields with no text:

--	--

This is a table with one row, one subrow, two columns, and two fields with text:

This is a <i>field</i> of text within a row.	This is another field of text in another column within a row.
--	---

This is a table with one row, two subrows, three columns, and five fields of text.

<i>This is a field of text in column one, subrows 1 and 2.</i>	This is a field of text in column 2, subrow 1.	This is a field of text in column 3, subrow 1.
	This is a field of text in column 2, subrow 2.	This is a field of text in column 3, subrow 2.

This is the basic approach to creating a table:

1. Define the format of the unique fields for all rows of the table (AFPDFLD).

This includes text positioning, text orientation, line spacing, shading, and so on. The field is the place where you put text.

2. Define the format of the unique rows of a table (AFPDRROW).

This includes arranging fields in the row, column width, minimum subrow depth, and the thickness of the top and bottom rules for the row. The row is the horizontal element in a table; it is divided into vertical elements called columns and can be divided horizontally into elements called subrows.

3. Begin the page (AFPBPAG).

4. Begin the table (AFPBTBL).

This includes specifying table rotation, width, maximum depth, and the thickness of the rules that surround the table.

5. Begin a row (AFPBROW), which was already defined in step 2.

6. Begin a field (AFPBFLD), which was already defined in step 1.

7. Place data in the field using either:

AFPPTXT, which puts text into a paragraph in the field to flow as defined in the AFPDFLD procedure call for text alignment.

AFPPCHS, which puts text in the field as entered (character alignment). The Alignment Position parameter in AFPDFLD is specified if you want character alignment (AFPPCHS alignment option).

8. End the field (AFPEFLD).

9. Repeat steps 6– 8 until all fields in the row are filled.

10. End the row (AFPEROW).

## Tables

11. Repeat steps 5–10 until data for the table is finished or until the end of the page is reached.
12. End the table (AFPETBL).
13. End the page (AFPEPAG).

Again, here is the first row of the table in the example:

Date	Transaction Description	Amount
------	-------------------------	--------

Figure 17. One Row of the Table

For the first row only of the table, here is the code to put in the template:

Code	Description
*-----*	*-----*
*   PROCESS THE TABLE.	*
*-----*	*-----*
/-----*	/-----*
*   THIS IS THE START OF THE FIELD AND ROW DEFINITIONS	*
*-----*	*-----*
MOVE 18 TO AFP-SHADING-INTENSITY.	
MOVE 0 TO AFP-ALIGNMENT-POSITION.	
MOVE 0.0 TO AFP-LEFT-MARGIN.	
MOVE 0.0 TO AFP-RIGHT-MARGIN.	
MOVE AFP-DEFAULT TO AFP-LINE-SPACING.	
MOVE .5 TO AFP-TOP-THICKNESS.	
MOVE .5 TO AFP-BOTTOM-THICKNESS.	
MOVE .5 TO AFP-LEFT-THICKNESS.	
MOVE .5 TO AFP-RIGHT-THICKNESS.	
CALL "AFPDFLD" USING	
BY CONTENT	
AFPAPI-HANDLE	
AFP-DOCUMENT-HANDLE	
FOCENTER	
AFP-ALIGNMENT-POSITION	
VERCENTER	
AFP-LEFT-MARGIN	
AFP-RIGHT-MARGIN	
AFP-LINE-SPACING	
TXTOR0-0	
SCREEN	
AFP-SHADING-INTENSITY	
AFP-TOP-THICKNESS	
AFP-BOTTOM-THICKNESS	
AFP-LEFT-THICKNESS	
AFP-RIGHT-THICKNESS	
BY REFERENCE	
FIELDH1	
AFP-RET-CODE	
AFP-SEVERITY-CODE.	

### AFPDFLD (Define Field):

Defines characteristics for the field for row 1 column 1.

- 18 is the shading pattern intensity from Appendix C, "Shade Patterns and Types" on page 261.
- MOVE 0 TO AFP-ALIGNMENT-POSITION means, if character alignment is specified in the AFPPCHS procedure call within this field, the position of the character in the field.
- MOVE 0.0 TO AFP-LEFT-MARGIN means no left margin within the field.
- MOVE 0.0 TO AFP-RIGHT-MARGIN means no right margin within the field.
- FOCENTER means center the text in the column horizontally.
- VERCENTER means center the text in the column vertically.
- TXTOR0-0 means no text rotation.
- SCREEN means use the specified shading pattern (18 in the example).
- All table rules are 0.5 millimeters.

Code	Description
<pre>CALL "AFPDFLD" USING   BY CONTENT     AFP-API-HANDLE     AFP-DOCUMENT-HANDLE     FOCENTER     AFP-ALIGNMENT-POSITION     VERCENTER     AFP-LEFT-MARGIN     AFP-RIGHT-MARGIN     AFP-LINE-SPACING     TXTORO-0     SCREEN     AFP-SHADING-INTENSITY     AFP-TOP-THICKNESS     AFP-BOTTOM-THICKNESS     AFP-LEFT-THICKNESS     AFP-RIGHT-THICKNESS   BY REFERENCE     FIELDH2     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPDFLD (Define Field):          Defines characteristics for the field for row 1 column 2.          It uses the same characteristics as row 1 column 1 above.</p>
<pre>CALL "AFPDFLD" USING   BY CONTENT     AFP-API-HANDLE     AFP-DOCUMENT-HANDLE     FOCENTER     AFP-ALIGNMENT-POSITION     VERCENTER     AFP-LEFT-MARGIN     AFP-RIGHT-MARGIN     AFP-LINE-SPACING     TXTORO-0     SCREEN     AFP-SHADING-INTENSITY     AFP-TOP-THICKNESS     AFP-BOTTOM-THICKNESS     AFP-LEFT-THICKNESS     AFP-RIGHT-THICKNESS   BY REFERENCE     FIELDH3     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPDFLD (Define Field):          Defines characteristics for the field for row 1 column 3.          It uses the same characteristics as row 1 column 1 above.</p>

## Tables

Code	Description
<pre> MOVE 3 TO AFP-NUMBER-COLUMNS. MOVE 1 TO AFP-NUMBER-SUBROWS. MOVE AFP-DEFAULT TO AFP-SUBROW-DEPTH(1). MOVE FIELDH1 TO AFP-COLUMN-ARRANGE (1, 1). MOVE 25.0 TO AFP-COLUMN-WIDTH (1). MOVE FIELDH2 TO AFP-COLUMN-ARRANGE (1, 2). MOVE 70.0 TO AFP-COLUMN-WIDTH (2). MOVE FIELDH3 TO AFP-COLUMN-ARRANGE (1, 3). MOVE 30.0 TO AFP-COLUMN-WIDTH (3). CALL "AFPDROW" USING     BY CONTENT     AFPAPI-HANDLE     AFP-DOCUMENT-HANDLE     AFP-MIN-SUBROW-DEPTH-ARRAY     AFP-TOP-THICKNESS     AFP-BOTTOM-THICKNESS     AFP-NUMBER-COLUMNS     AFP-NUMBER-SUBROWS     AFP-ROW-ARRANGE-ARRAY     AFP-COLUMN-WIDTH-ARRAY     BY REFERENCE     ROW1     AFP-RET-CODE     AFP-SEVERITY-CODE.  /-----* * END OF THE FIELD AND ROW DEFINITIONS *-----*  /-----* *-----* *          Begin a table *          Write the header row *          End the table *-----*  *-----* * Start the table whose maximum depth is the remaining page * body space after the white space preceding the table. *-----* </pre>	<p>AFPDROW (Define Row):          Defines characteristics for the three columns in row 1.</p> <ul style="list-style-type: none"> <li>AFP-DEFAULT TO AFP-SUBROW-DEPTH(1) means use a subrow depth for each subrow determined by the font used.</li> <li>FIELDH1 TO AFP-COLUMN-ARRANGE (1, 1) means use the characteristics for row 1 column 1 that were specified in the AFPDFLD that returned an ID of FIELDH1 (above).</li> <li>25.0 TO AFP-COLUMN-WIDTH (1) means use 25.0 millimeters for the width of column 1.</li> <li>FIELDH2 TO AFP-COLUMN-ARRANGE (1, 2) means use the characteristics for row 1 column 2 that were specified in the AFPDFLD that returned an ID of FIELDH2 (above).</li> <li>70.0 TO AFP-COLUMN-WIDTH (2) means use 70.0 millimeters for the width of column 2.</li> <li>FIELDH3 TO AFP-COLUMN-ARRANGE (1, 3) means use the characteristics for row 1 column 3 that were specified in the AFPDFLD that returned an ID of FIELDH3 (above).</li> <li>30.0 TO AFP-COLUMN-WIDTH (3) means use 30.0 millimeters for the width of column 3.</li> </ul>
<pre> MOVE 45 TO AFP-X-COORDINATE. MOVE TABLE-WHITE-SPACE TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT     AFPAPI-HANDLE     AFP-PAGE-HANDLE     AFP-X-COORDINATE     XABS     AFP-Y-COORDINATE     YREL     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>AFPSPOS (Set Position):          Sets the position for the table at 45 millimeters in the X-direction and after the designated white space in the Y-direction. DATA DIVISION contains the value for TABLE-WHITE-SPACE. XABS and YREL mean the X-absolute and Y-relative reference coordinate systems, respectively. See "Setting Attributes (and Querying Them)" on page 73 for a description of absolute and relative coordinate systems.</p>

Code	Description
<pre> COMPUTE AFP-MAX-TABLE-DEPTH = PAGE-BODY -                                 TABLE-WHITE-SPACE. MOVE 125.0 TO AFP-TABLE-WIDTH. MOVE 1.0 TO AFP-TOP-THICKNESS. MOVE .5 TO AFP-BOTTOM-THICKNESS. MOVE .5 TO AFP-LEFT-THICKNESS. MOVE .5 TO AFP-RIGHT-THICKNESS. CALL "AFPBTBL" USING   BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-TABLE-WIDTH     AFP-MAX-TABLE-DEPTH     ROTATE0     AFP-TOP-THICKNESS     AFP-BOTTOM-THICKNESS     AFP-LEFT-THICKNESS     AFP-RIGHT-THICKNESS   BY REFERENCE     AFP-TABLE-HANDLE     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>AFPBTBL (Begin Table): Begins the table, sets up its size, and sets up the rule thickness.</p> <p>ROTATE0 means the table is not rotated.</p>
<pre> *-----* /-----* *-----* * WRITE-HEADER-ROW. * * * * Write the header row for the table. * *-----* WRITE-HEADER-ROWS. CALL "AFPBROW"   USING   BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     ROW1   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>AFPBROW (Begin Row): Begins the header row, using the ROW1 ID returned from AFPDROW for formatting.</p>
<pre> *-----* * Write the first field in column 1. * *-----* CALL "AFPBFLD"   USING   BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     FIELDH1   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>AFPBFLD (Begin Field): Begins the first field in the header row, using the FIELDH1 ID returned from AFPDFLD for formatting.</p>
<pre> CALL "AFPSFNT" USING   BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     TIM12MED   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>AFPSFNT (Set Font): Sets the font for the header row.</p>

## Tables

Code	Description
<pre>MOVE "Date" TO AFP-STRING-IN. CALL "TRIM" USING AFP-STRING-IN,     BY CONTENT LENGTH OF AFP-STRING-IN,     BY REFERENCE AFP-CHARACTER-STRING,     AFP-STRING-LENGTH.</pre>	<p>TRIM</p> <p>Uses "Date" as the AFP-STRING-IN identifier, strips off unnecessary leading and trailing blanks in the data for proper formatting, counts the characters (length) of the data, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>
<pre>CALL "AFPPCHS" USING     BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     AFP-STRING-LENGTH     AFP-CHARACTER-STRING     CENTER     AFP-ALIGNMENT-CHAR     FALS     FALS     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPPCHS:</p> <p>Puts a character string (Date) in the field and centers it.</p>
<pre>CALL "AFPEFLD" USING     BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPEFLD:</p> <p>Ends the first field in row 1.</p>
<pre>*-----* * Write the second field in column 2.          * *-----* CALL "AFPBFLD"     USING     BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     FIELDH2     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPBFLD (Begin Field):</p> <p>Begins the second field in the header row, using the FIELDH2 ID returned from AFPDFLD for formatting.</p>
<pre>MOVE SPACES TO AFP-STRING-IN. MOVE "Transaction Description" TO AFP-STRING-IN. CALL "TRIM" USING AFP-STRING-IN,     BY CONTENT LENGTH OF AFP-STRING-IN,     BY REFERENCE AFP-CHARACTER-STRING,     AFP-STRING-LENGTH.</pre>	<p>TRIM:</p> <p>Uses "Transaction Description" for AFP-STRING-IN, strips off unnecessary leading and trailing blanks in the data for correct formatting, counts the characters (length) of the data, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>

Code	Description
<pre>CALL "AFPPCHS" USING   BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     AFP-STRING-LENGTH     AFP-CHARACTER-STRING     CENTER     AFP-ALIGNMENT-CHAR     FALS     FALS   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPPCHS: Puts a character string (Transaction Description) on the page and centers it.</p>
<pre>CALL "AFPEFLD" USING   BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPEFLD: Ends the second field in row 1.</p>
<pre>*-----* * Write the third field in column 3.          * *-----* CALL "AFPBFLD"   USING     BY CONTENT       AFP-API-HANDLE       AFP-TABLE-HANDLE       FIELDH3     BY REFERENCE       AFP-RET-CODE       AFP-SEVERITY-CODE.</pre>	<p>AFPBFLD (Begin Field): Begins the second field in the header row, using the FIELDH3 ID returned from AFPDFLD for formatting.</p>
<pre>MOVE SPACES TO AFP-STRING-IN. MOVE "Amount" TO AFP-STRING-IN. CALL "TRIM" USING AFP-STRING-IN,   BY CONTENT LENGTH OF AFP-STRING-IN,   BY REFERENCE AFP-CHARACTER-STRING,   AFP-STRING-LENGTH.</pre>	<p>TRIM: Uses "Amount" for AFP-STRING-IN, strips off unnecessary leading and trailing blanks in the data for correct formatting, counts the characters (length) of the data, inserts the trimmed string in AFP-CHARACTER-STRING, and inserts the string length in AFP-STRING-LENGTH. TRIM is a subprogram shipped with the example code.</p>
<pre>CALL "AFPPCHS" USING   BY CONTENT     AFP-API-HANDLE     AFP-TABLE-HANDLE     AFP-STRING-LENGTH     AFP-CHARACTER-STRING     CENTER     AFP-ALIGNMENT-CHAR     FALS     FALS   BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p>AFPPCHS: Puts a character string (Amount) on the page and centers it.</p>

## Tables


Code	Description
<pre>CALL "AFPEFLD" USING   BY CONTENT   AFP-API-HANDLE   AFP-TABLE-HANDLE   BY REFERENCE   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	<p>AFPEFLD: Ends the third field in row 1.</p>
<pre>CALL "AFPEROW"   USING   BY CONTENT   AFP-API-HANDLE   AFP-TABLE-HANDLE   BY REFERENCE   AFP-CURRENT-TABLE-DEPTH   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	<p>AFPEROW: Ends row 1.</p>
<pre>*-----* * End the table. *-----*</pre>	<p>AFPETBL: Ends the table, returns the final depth of the table to the program, resets the font to what it was before the AFPBTBL call, and establishes the current position as the bottom-left corner of the table.</p>
<pre>CALL "AFPETBL" USING   BY CONTENT   AFP-API-HANDLE   BY REFERENCE   AFP-TABLE-HANDLE   AFP-TABLE-DEPTH   AFP-RET-CODE   AFP-SEVERITY-CODE.</pre>	

After this overview and studying the code for one row, examine the sample document again. The code for the entire table is lengthy and won't be repeated here. By knowing what to look for in the source code, you should be able to understand the coding for the entire table.

Figure 18 on page 57 shows the sample document, which has a table containing the following:

- A heading row with three columns and shading (code covered above)
- Several transaction rows with three columns and no shading
- A summary row with two-columns and shading






3000/000000  
 3000 0000 7777 8888  
 Date: 01/01/14 10:00  
 Amount: \$1,075.18  
 Payment No: 01075.00  
 Card Order No: 00000000000000000000  
 \* Please check the service with your agency.

Susan B. Ames  
 09960 St. Christopher Road  
 The City, Colorado 80000

**CONGRATULATIONS, Susan B. Ames!** Because of your excellent credit rating, you are now eligible for free credit insurance which protects you in case your Primo card is ever lost or stolen. Call NOW for more information!

ID	Description	Amount
0020	ONTEL SERV Lodging Winter Park CO	\$173.12
0024	The Last Dinner Winter Park CO	\$28.95
0025	Jeans Cheese House Winter Park CO	\$44.08
0025	Ticket Sales Winter Park CO	\$34.00
0025	Ticket Sales Winter Park CO	\$34.00
0025	Old School Winter Park CO	\$90.01
0025	Cha-N-Go Winter Park CO	\$25.00
0025	Winter Park Restaurant Winter Park CO	\$55.22
0901	Alex's Restaurant Boulder CO	\$22.03
0903	Pont and Ankle Clinic Boulder CO	\$90.01
0903	Covenant Brewery Boulder CO	\$59.07
0905	Subs Your Taste Boulder CO	\$69.00
0906	Play CO Boulder CO	\$167.89
0906	Kids the 'Troy Town CO	\$53.02
0909	Townsend Restaurant Boulder CO	\$20.15
0910	Boulder Elks Boulder CO	\$264.52
0912	Polman John's Boulder CO	\$50.59
0917	PAPA Auto Parts Boulder CO	\$187.71
0917	Harold's Restaurant Boulder CO	\$30.98
0918	Clubs Lim Boulder CO	\$266.53
0918	John's Boulder CO	\$40.59
0918	Walt's Jewelry Boulder CO	\$1,200.00
0918	Krugger's Cards and Books Boulder CO	\$13.92
0919	Fill-R-Up Denver CO	\$18.50
0919	Best Buy Afr Denver CO	\$209.95
0920	Phat's Diner Inland Denver CO	\$75.48
0920	Joe's (M) Denver CO	\$15.77

Page 1



ID	Description	Amount
0920	Smith Gas Denver CO	\$10.00
0920	Pete's Easy Riders Christmas CO	\$85.00
Total Amount		\$9,872.99

Page 2

Figure 18. Sample Document

You must define a separate AFPDROW call for each type of row in the table and for all the unique AFPDFLD parameters for each field in each row. To do this, determine the number of columns and subrows in each row. All rows have one subrow, and the table has three types of rows, as described above.

**Note:** You can use a field in more than one row, but you cannot use a field more than once in the same row.

Look at the source code in *AFP Application Programming Interface: COBOL Language Reference* and in *AFP Application Programming Interface: PL/1 Language Reference*. For COBOL, the AFPINIT section contains all of the row and field definitions. In the code, look for these IDs and study the code and comments:

AFPDROW (Define Row) and AFPBROW (Begin Row):

- ROW1 (ID for the header row)
- ROW2 (ID for the transaction rows)
- ROW3 (ID for the summary row)

AFPDFLD (Define Field) and AFPBFLD (Begin Field):

- FIELDH1, FIELDH2, and FIELDH3 (IDs for the three fields, one in each column of the header row)
- FIELDT1, FIELDT2, and FIELDT3 (IDs for the three fields, one in each column of the transaction row)
- FIELDS1 and FIELDS2 (IDs for the two fields, one in each column of the summary row)

## Tables

Other procedure calls that affect tables are as follows. (See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions for these procedure calls).

AFPPTXT (Put Text)

Puts text (to be formatted) in the field

AFPSCLR (Set Color)

Sets the color of the field

AFPSSICS (Set Intercharacter Spacing)

Sets the intercharacter spacing for the field

AFPSWSP (Set Word Spacing)

Sets the word spacing for the field

### More About Tables

The preceding example showed the code for a row with one subrow. Here is the code and the process for developing a table that has one row with three subrows.

First, please review some terms. When you build a table, you use three basic elements:

**Field** A field is rectangular and is usually separated from other fields by horizontal and vertical rules. A field is similar to a column but does not always go from the top of the row to the bottom of the row.

**Row** A row is a horizontal, rectangular collection of one or more fields. The fields that make up a row may have different widths and depths.

**Table** A table is a collection of one or more rows. The rows that make up a table may contain different field arrangements.

The parameters on the AFPTBL, AFPDROW, and AFPDFLD procedure calls provide flexibility in specifying different characteristics of the table. For example, you can specify shading to be used in a field and the vertical alignment of the contents of the field. Consider the following table:

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

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

**Step 1. Sketch the Row**

Sketch your row and assign an identifier to each field in the row. You don't have to number the fields sequentially, but the process is easier if you do.

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

**Step 2. Define all of the Fields in the Row**

The table has eight Fields, so you must issue the AFPDFLD call eight times and save the IDs from each call. The field IDs are FIELD1, FIELD2, FIELD3, FIELD4, FIELD5, FIELD6, FIELD7, and FIELD8.

**Step 3. Form a Grid**

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


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

**Step 4. Determine the Number of Columns and Their Widths**

Determine the horizontal width of each rectangle in the grid from Step 3. These are the values to use on the COLUMN-WIDTH parameter of the AFPDROW procedure call.

12	12	50	12	12

The value of the NUMBER-COLUMNS parameter in AFPDROW is 5.

The value of the TABLE-WIDTH parameter in AFPBTABL is the sum of the column widths, which is 98 (12 plus 12 plus 50 plus 12 plus 12).

### Step 5. Determine the Number Subrows and Their Depths

Determine the number of subrows in the grid from Step 3. This is the value to use in the NUMBER-SUBROWS parameter in the AFPDROW procedure call. The value in this case is 3.

The depth of each subrow is specified in the SUBROW-DEPTH parameter in AFPDROW. This example uses the default subrow depth determined by the font being used.

### Step 6. Determine the Arrangement of Each Field Within Each Subrow

Construct an array called COLUMN-ARRANGE with five columns and three subrows, and store the arrangement of the field IDs in the elements of the array, as shown in the following figure:

Field1	Field1	Field5	Field6	Field7
Field2	Field3	Field5	Field8	Field8
Field4	Field4	Field5	Field8	Field8

### Step 7. Define The Row

This AFPDROW includes the IDs returned from the eight AFPDFLD calls identified in Step 2. The example uses IDs from the eight AFPDFLD calls but doesn't show the code for them. Page50 described how to code AFPDFLD calls. The complete row definition AFPDROW looks like this:

Code	Description
*-----*	
* DEFINE THE FIELDS *	
*-----*	
CALL "AFPDFLD" USING	AFPDFLD (Define Field):
..	See page50 for how to code FIELDH1, FIELDH2, and FIELDH3. The
..	coding for FIELD1, FIELD2, FIELD3, FIELD4, FIELD5, FIELD6, FIELD7,
..	and FIELD8 is similar.

Code	Description
<pre> *-----* *  DEFINE THE ROW  * *-----* </pre>	<p>AFPDROW (Define Row): Defines characteristics for the five columns in the row.</p>
<pre> MOVE 5 TO AFP-NUMBER-COLUMNS. MOVE 3 TO AFP-NUMBER-SUBROWS. MOVE AFP-DEFAULT TO AFP-SUBROW-DEPTH(1). MOVE AFP-DEFAULT TO AFP-SUBROW-DEPTH(2). MOVE AFP-DEFAULT TO AFP-SUBROW-DEPTH(3). MOVE FIELD1 TO AFP-COLUMN-ARRANGE (1, 1). MOVE FIELD1 TO AFP-COLUMN-ARRANGE (1, 2). MOVE FIELD2 TO AFP-COLUMN-ARRANGE (1, 3). MOVE FIELD6 TO AFP-COLUMN-ARRANGE (1, 4). MOVE FIELD7 TO AFP-COLUMN-ARRANGE (1, 5). MOVE FIELD8 TO AFP-COLUMN-ARRANGE (2, 1). MOVE FIELD3 TO AFP-COLUMN-ARRANGE (2, 2). MOVE FIELD5 TO AFP-COLUMN-ARRANGE (2, 3). MOVE FIELD8 TO AFP-COLUMN-ARRANGE (2, 4). MOVE FIELD8 TO AFP-COLUMN-ARRANGE (2, 5). MOVE FIELD4 TO AFP-COLUMN-ARRANGE (3, 1). MOVE FIELD4 TO AFP-COLUMN-ARRANGE (3, 2). MOVE FIELD5 TO AFP-COLUMN-ARRANGE (3, 3). MOVE FIELD8 TO AFP-COLUMN-ARRANGE (3, 4). MOVE FIELD8 TO AFP-COLUMN-ARRANGE (3, 5). MOVE 12.0 TO AFP-COLUMN-WIDTH (1). MOVE 12.0 TO AFP-COLUMN-WIDTH (2). MOVE 50.0 TO AFP-COLUMN-WIDTH (3). MOVE 12.0 TO AFP-COLUMN-WIDTH (4). MOVE 12.0 TO AFP-COLUMN-WIDTH (5). CALL "AFPDROW" USING   BY CONTENT   AFP-API-HANDLE   AFP-DOCUMENT-HANDLE   AFP-MIN-SUBROW-DEPTH-ARRAY   AFP-TOP-THICKNESS   AFP-BOTTOM-THICKNESS   AFP-NUMBER-COLUMNS   AFP-NUMBER-SUBROWS   AFP-ROW-ARRANGE-ARRAY   AFP-COLUMN-WIDTH-ARRAY   BY REFERENCE   ROW1   AFP-RET-CODE   AFP-SEVERITY-CODE. </pre>	<ul style="list-style-type: none"> <li>• AFP-DEFAULT TO AFP-SUBROW-DEPTH(1) means use the default for the depth of subrow 1. The font you use determines this default.</li> <li>• AFP-DEFAULT TO AFP-SUBROW-DEPTH(2) means use the default for the depth of subrow 2. The font you use determines this default.</li> <li>• AFP-DEFAULT TO AFP-SUBROW-DEPTH(3) means use the default for the depth of subrow 3. The font you use determines this default.</li> <li>• FIELD1 TO AFP-COLUMN-ARRANGE (1, 1) means use the characteristics for subrow 1 column 1 that were specified in the AFPDFLD call that returned an ID of FIELD1.</li> <li>• FIELD1 TO AFP-COLUMN-ARRANGE (1, 2) means use the characteristics for subrow 1 column 2 that were specified in the AFPDFLD call that returned an ID of FIELD1.</li> <li>• FIELD5 TO AFP-COLUMN-ARRANGE (1, 3) means use the characteristics for subrow 1 column 3 that were specified in the AFPDFLD call that returned an ID of FIELD5.</li> <li>• FIELD6 TO AFP-COLUMN-ARRANGE (1, 4) means use the characteristics for subrow 1 column 4 that were specified in the AFPDFLD call that returned an ID of FIELD6.</li> <li>• FIELD7 TO AFP-COLUMN-ARRANGE (1, 5) means use the characteristics for subrow 1 column 5 that were specified in the AFPDFLD call that returned an ID of FIELD7.</li> <li>• FIELD2 TO AFP-COLUMN-ARRANGE (2, 1) means use the characteristics for subrow 2 column 1 that were specified in the AFPDFLD call that returned an ID of FIELD2.</li> <li>• FIELD3 TO AFP-COLUMN-ARRANGE (2, 2) means use the characteristics for subrow 2 column 2 that were specified in the AFPDFLD call that returned an ID of FIELD3.</li> <li>• FIELD5 TO AFP-COLUMN-ARRANGE (2, 3) means use the characteristics for subrow 2 column 3 that were specified in the AFPDFLD call that returned an ID of FIELD5.</li> <li>• FIELD8 TO AFP-COLUMN-ARRANGE (2, 4) means use the characteristics for subrow 2 column 4 that were specified in the AFPDFLD call that returned an ID of FIELD8.</li> <li>• FIELD8 TO AFP-COLUMN-ARRANGE (2, 5) means use the characteristics for subrow 2 column 5 that were specified in the AFPDFLD call that returned an ID of FIELD8.</li> <li>• FIELD4 TO AFP-COLUMN-ARRANGE (3, 1) means use the characteristics for subrow 3 column 1 that were specified in the AFPDFLD call that returned an ID of FIELD4.</li> <li>• FIELD4 TO AFP-COLUMN-ARRANGE (3, 2) means use the characteristics for subrow 3 column 2 that were specified in the AFPDFLD call that returned an ID of FIELD4.</li> <li>• FIELD5 TO AFP-COLUMN-ARRANGE (3, 3) means use the characteristics for subrow 3 column 3 that were specified in the AFPDFLD call that returned an ID of FIELD5.</li> <li>• FIELD8 TO AFP-COLUMN-ARRANGE (3, 4) means use the characteristics for subrow 3 column 4 that were specified in the AFPDFLD call that returned an ID of FIELD8.</li> <li>• FIELD8 TO AFP-COLUMN-ARRANGE (3, 5) means use the characteristics for subrow 3 column 5 that were specified in the AFPDFLD call that returned an ID of FIELD8.</li> <li>• 12 TO AFP-COLUMN-WIDTH (1) means that column 1 is 12 millimeters wide.</li> <li>• 12 TO AFP-COLUMN-WIDTH (2) means that column 2 is 12 millimeters wide.</li> <li>• 50 TO AFP-COLUMN-WIDTH (3) means that column 3 is 50 millimeters wide.</li> <li>• 12 TO AFP-COLUMN-WIDTH (4) means that column 4 is 12 millimeters wide.</li> <li>• 12 TO AFP-COLUMN-WIDTH (5) means that column 5 is 12 millimeters wide.</li> </ul>

## Box

You can draw a box anywhere on a page and shade it if you want. You can even shade an “invisible” box (that is, a box with shading but no rules enclosing it) by setting the rule thickness to zero before drawing the box. The example does not contain a AFPPBOX procedure call, but here is a typical box. (See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions.)

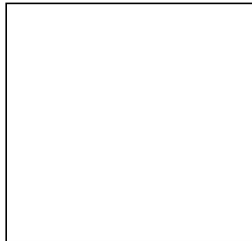


Figure 19. Box

Here are the steps for placing this box on the page:

1. Set the rule thickness.
2. Set the position for where the box begins.
3. Put the box at the specified position.

Here is the code:

Code	Description
<pre> *-----* * Draw a box *-----* MOVE 0.3 TO AFP-RULE-THICKNESS. CALL "AFPSRTH" USING     BY CONTENT         AFPAPI-HANDLE         AFP-PAGE-HANDLE         AFP-RULE-THICKNESS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. MOVE 29 TO AFP-X-COORDINATE. MOVE 73 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT         AFPAPI-HANDLE         AFP-PAGE-HANDLE         AFP-X-COORDINATE         XABS         AFP-Y-COORDINATE         YABS     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. MOVE 33 TO AFP-BOX-WIDTH. MOVE 31 TO AFP-BOX-DEPTH. MOVE 0 TO AFP-SHADING-INTENSITY. CALL "AFPPBOX" USING     BY CONTENT         AFPAPI-HANDLE         AFP-PAGE-HANDLE         AFP-BOX-WIDTH         AFP-BOX-DEPTH         NOSHADE         AFP-SHADING-INTENSITY     BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	<p>AFPSRTH (Set Rule Thickness): Specifies the thickness for the sides of a box in the unit of measure currently in effect, in this case, 0.3 millimeters for the box.</p> <p>AFPSPOS (Set Position): Sets the position for placing the rule as 29 millimeters in the X-direction and 73 millimeters in the Y-direction. XABS and YABS mean the X-absolute and Y-absolute reference coordinate systems, respectively. See “Setting Attributes (and Querying Them)” on page 73 for a description of absolute and relative coordinate systems.</p> <p>AFPPBOX (Put Box): Draws a box with the top-left corner beginning at the current position. The example specifies a width of 33 millimeters, a depth of 31 millimeters, and no shading for the box. The current values for rule thickness and color apply. See “AFPSCLR (Set Color)” on page 181 for how to specify color. If you specify shading and a value of 0 for rule thickness, you get shading with no box. The current position is unchanged after the box is drawn.</p>

## Include Object

You can include Image Object Content Architecture (IOCA) and Graphics Object Content Architecture (GOCA) objects in a document. The object can be either in a document or a page segment, such as those created by Graphical Data Display Manager (GDDM), or it can be an individual object. When you include an individual object, you can instruct the printer to change the size and rotation of the object when it is printed.

When AFP API is running in a CICS/ESA environment, you cannot include individual image and graphic objects. The objects must be in page segments; use the AFPIPSG (Include Page Segment) procedure call to include a page segment.

When you include an individual object, AFP API retrieves the member containing the object either from the object library specified in the AFPSLIB procedure call or from the default object library. If the included member contains more than one object, only the first object inside the member is included in the document. AFP API places the object at the current position. “Data Objects and AFP Resource Objects” on page 11 describes data objects.

The example does not contain an AFPIOBJ procedure call, but here is an example of using it to include an IOCA object named IOCAMMR that is shipped with PSF/MVS, PSF/VM, and PSF/VSE. The PSF installation process stores the object in the PSF page segment library. To use the IOCAMMR object with AFP API, copy it into the AFP API object library. See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions. Here are the steps for including a data object in a document:

1. Set the position for where to place the upper-left corner of the object.
2. Include the object at the specified position.

The IOCAMMR object looks like this when it is included as shown in the following sample code:

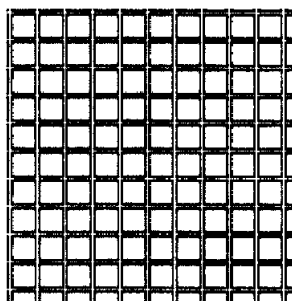


Figure 20. Example of the IOCAMMR Data Object Shipped with PSF

## Include Object

Here is the code:

Code	Description
<pre>MOVE 29 TO AFP-X-COORDINATE. MOVE 23 TO AFP-Y-COORDINATE. CALL "AFPSPOS" USING     BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-X-COORDINATE     XABS     AFP-Y-COORDINATE     YREL     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p><b>AFPSPOS (Set Position):</b> Sets the position for the object at 29 millimeters in the X-direction and 23 millimeters in the Y-direction.</p>
<pre>*-----* * Include Object *-----*</pre>	
<pre>MOVE "IOCAMMR" TO AFP-OBJECT-NAME. MOVE 38.1 TO AFP-OBJECT-WIDTH. MOVE 38.1 TO AFP-OBJECT-DEPTH. MOVE ROTATEO TO AFP-OBJECT-ROTATION. MOVE SCALE-TO-FIT TO AFP-OBJECT-MAPPING-OPTION. MOVE AFP-DEFAULT TO AFP-OBJECT-X-OFFSET. MOVE AFP-DEFAULT TO AFP-OBJECT-Y-OFFSET. CALL "AFPIOBJ" USING     BY CONTENT     AFP-API-HANDLE     AFP-PAGE-HANDLE     AFP-OBJECT-NAME     AFP-OBJECT-WIDTH     AFP-OBJECT-DEPTH     AFP-OBJECT-ROTATION     AFP-OBJECT-MAPPING-OPTION     AFP-OBJECT-X-OFFSET     AFP-OBJECT-Y-OFFSET     BY REFERENCE     AFP-RET-CODE     AFP-SEVERITY-CODE.</pre>	<p><b>AFPIOBJ (Include Object):</b> Includes the SPECIFIED object at the current position. Specify the width and the depth of the space on the page in which you are placing the object (38.1 millimeters) and specify how the printer is to map the object into the space. The example specifies SCALE-TO-FIT, which means that the object should be shrunk or expanded to fit into the specified space. The current position is unchanged after including the object.</p>



## Introducing Return Codes and Severity Codes

AFP API sends return codes and severity codes back to the program after processing every procedure call. The application program should monitor these codes after every call and respond accordingly. The example doesn't check return codes on every call; however, checking them aids in debugging your program.

See Appendix B, "Return Codes and Severity Codes" for detailed descriptions for these codes.

Figure 21 shows the usual flow of information between AFP API and the application program and from AFP API to an output file or to a CICS/ESA temporary storage queue.

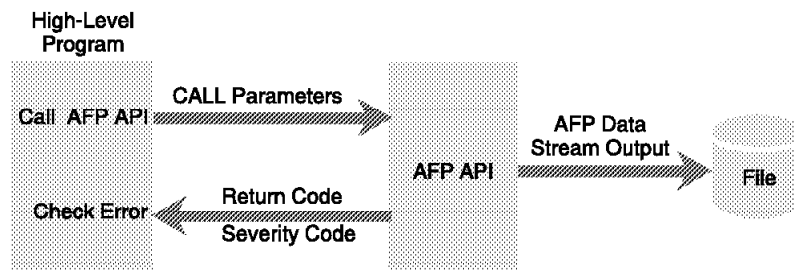


Figure 21. Information Flow. AFP API sends the AFP data stream to an output file or to a CICS/ESA temporary storage queue for printing.

**Note:** AFP API can write the AFP data stream to an output buffer in your program instead of to an output file. See "Buffering AFP API Output" on page 66 for more information.

## Setting Up and Defining the Environment for an AFP API Session

When the example in "Program Template" on page 24 was initialized, it didn't include the code that sets output characteristics, defines resource libraries, defines fonts, and sets (and queries) attributes. After the output characteristics are set and the resource libraries, fonts, and attributes are defined, you can use these values throughout AFP API.

AFP API also has default values for most parameters, which is described in "Setting Attributes (and Querying Them)" on page 73. With *set* procedure calls, you can override both values specified at initialization and any AFP API default values.

### Setting Output Characteristics and Resource Libraries

During initialization, you must set some output characteristics and identify the libraries where resources are stored, unless you want to use the defaults. The procedure calls described here do that. See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions for these procedure calls.

The procedure calls for setting up the characteristics of an AFP API session are:

**AFPSOUT** (Set Output Characteristics):

Establishes the following:

- The maximum size of a variable-length data-stream record
- The name of the AFP API output file (or the DD card name on MVS or the file name of a DLBL statement on VSE) if AFP API is to write the output records to a file
- Whether or not to create a new output file or replace an existing one
- The name of the queue if AFP API is to write output records to a CICS/ESA transient data queue
- Whether AFP API is to return output records to a buffer in your program or discard the output, instead of writing the output to an output file.

**AFPSLIB** (Set Resource Library Names):

Identifies the libraries where fonts, page segments, and objects reside. AFP API must have access to these libraries. See “More About Resources” on page 34 for more information about resources and how to access them.<sup>3</sup>

In a CICS/ESA environment, you do not identify font and page segment resources with the AFPSLIB (Set Resource Library Names) procedure call. Font and page segment resources must be in VSAM data sets defined to CICS/ESA with the names FONTLIB and SEGLIB.

### Buffering AFP API Output

During initialization, you can request that AFP API return the output data stream to a buffer in your program, instead of writing it to an output file or to a CICS/ESA temporary storage queue. Returning the output data stream to a buffer allows your program to modify the data stream.

Figure Figure 22 on page 67 shows the flow of information when you request that AFP API buffer the output data stream.

---

<sup>3</sup> On the VM operating system, AFP API searches in alphabetical order for fonts, page segments, and included objects on the first minidisk on which the resource is stored. For example, AFP API searches for a page segment named RUFUS as RUFUS PSEG3820 \*.

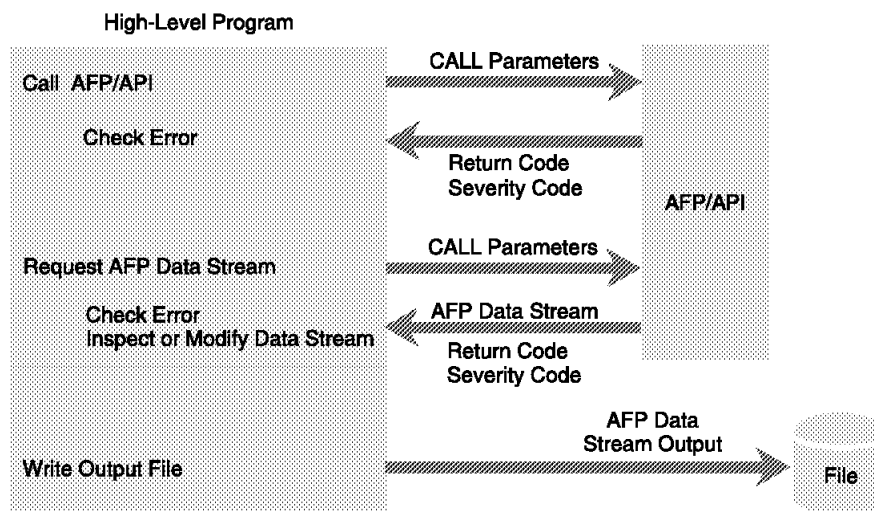


Figure 22. Information Flow Using Buffered Output. AFP API sends the AFP data stream to your program.

Follow these steps to use output buffering:

1. Issue the AFPSOUT procedure call to request that AFP API return output records to a buffer in your program instead of writing them to a file. To do this, specify the constant BUFFERED as the output file name on the AFPSOUT procedure call.
2. After each page is ended, that is, after the AFPEPAG procedure call, issue the AFPGBUF procedure call repeatedly to retrieve each record for the page just completed. AFP API returns each record and its length in a buffer you provide in your program. Each record contains one structured field.
3. Issue the AFPEDOC procedure call to end the document and obtain the last output record in the same buffer provided in the last AFPGBUF procedure call.
4. Write the output records to an output file or to a CICS temporary storage queue.

Sample COBOL code showing how to use output buffering is printed in *AFP Application Programming Interface: COBOL Language Reference*.

## Defining Fonts and Using Them with AFP API

You must tell AFP API which fonts to use and where they are stored. This section provides an overview of IBM fonts and describes some of the terms used to identify them. AFP API requires that the fonts you tell AFP API to use have been installed using the Font Indexing Library Program (FLIP). Also, the installation-verification program (IVP) for the AFP API uses certain fonts, which you must install using FLIP; refer to the *Program Directory* for information about the fonts required by the IVP.

The AFP Font Collection product provides fonts. For samples of these fonts, refer to *IBM AFP Fonts: Font Samples*.

You can identify fonts by appearance, height, width, and thickness of the characters in the font, provided the fonts are defined in your font library. The steps below show how to identify, select, and use the font for Susan's name in

the example, shown again in Figure 23 on page 68. Notice that Susan's name is printed with a different font than that used for the address. The following is the process for selecting only the font for her name.

**Susan B. Ames**  
98765 N. Frontage Road  
Tin Cup, Colorado 80000

Figure 23. Character String

### Selecting the Font You Want

The code for placing Susan's name on the page is described in "Character String" on page 28, but the sample doesn't show how to define the font. The following are the steps to select and use the font for Susan's name:

1. Select a typeface from *IBM AFP Fonts: Font Samples*. The example uses Latin1: Times New Roman Roman Bold and notes the name, point size, weight, and width.
2. Look for that typeface in *IBM AFP Fonts: Technical Reference for IBM Expanded Core Fonts* to select the font's code page<sup>4</sup> from a list of valid code pages. The example uses code page T1V10500.

**Note:** ASCII code pages are *not* supported.

3. Verify that the font is on your system by doing the following:
  - Locate the font's characteristics (name, point size, weight, and width) in the Font Library Indexing Program output listing. Figure 24 on page 71 is an example of such a listing.

Fonts have a *Descriptive Name*, which in this case is "TIMES NEW ROMAN LATIN1." From the listing, you can get the information about the font that you need to complete the AFPDFNT procedure call.
  - Look in the font library on your system for the code page that you selected, in the sample, code page T1V10500. The font library is the one you identified in the AFPSLIB call. If you don't know the font library name, see your system programmer.
4. Invoke the AFPDFNT call with the following information for the example:
  - Code page: T1V10500
  - Descriptive name length: 22 (TIMES NEW ROMAN LATIN1 has a length of 22, including characters and blanks.)
  - Descriptive name: TIMES NEW ROMAN LATIN1.
  - Point size: 12 points (vertical font size)
  - Weight: Bold
  - Width: Normal (the horizontal size of a font)
  - Rotation: 0 (the clockwise rotation of a character in a font). Rotation values are not in the listing, but the choices are: 0°, 90°, 180°, or 270°.

---

<sup>4</sup> A code page is a particular assignment of hexadecimal identifiers to graphic characters. For example, X'4F' is an exclamation point (!) in code page T1V10500, and X'5A' is an exclamation point (!) in code page T1V10037. If the code page used by your terminal or system for input is different from the code page used to present the data, you may not get the output you expect.

- Style: Roman (the posture of a font, roman or italic)
5. Invoke AFPSFNT with the ID returned from AFPDFNT.
  6. Invoke AFPPCHS with Susan’s name. The complete code for doing this is shown in “Character String” on page 28.

Here is the code for AFPDFNT:

Code	Description
<pre> MOVE "TIV10500" TO AFP-CODE-PAGE. MOVE 22 TO AFP-DESC-NAME-LENGTH. MOVE "TIMES NEW ROMAN LATINI" TO AFP-DESCRIPTIVE-NAME. MOVE 12 TO AFP-POINT-SIZE. CALL "AFPDFNT" USING   BY CONTENT     AFP-API-HANDLE     AFP-DOCUMENT-HANDLE     AFP-CODE-PAGE     AFP-DESC-NAME-LENGTH     AFP-DESCRIPTIVE-NAME     AFP-POINT-SIZE     BOLD     NORMAL     ORIENTO     ROMAN   BY REFERENCE     TIM12BOLD     AFP-RET-CODE     AFP-SEVERITY-CODE. </pre>	<p>Defines all the attributes for the font specified with the AFPSFNT for Susan’s name. TIM12BOLD is the ID to use in the AFPSFNT call to identify the font defined in this AFPDFNT call. “Character String” on page 28 contains the code for placing Susan’s name on the page.</p>

See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions for the following procedure calls.

**AFPDFNT (Define Font by Attributes):**

Creates a font ID on your system that matches a specified font’s attributes, such as, code page, point size, weight, width, rotation, and style. AFPSFNT uses this font ID when specifying a font. The font must exist in a font library available to AFP API.

**AFPSFNT (Set Font):**

Specifies one of the fonts defined with the AFPDFNT Procedure Call for use in subsequent text. You must issue a new AFPSFNT call whenever you want to change fonts.

**Font Library Indexing Program**

This section provides a brief overview of the Font Library Indexing Program (FLIP) and shows how to use it. A system programmer runs this program whenever a font is modified, added, or deleted.

**Font Library Indexing Program (FLIP):** FLIP,<sup>5</sup> which is used to document the contents of your AFP font library, produces a listing of the available fonts; this listing includes several attributes of the fonts that you need when you code the AFPDFNT.

---

<sup>5</sup> The FLIP program that comes with AFP API is similar to the one available with Document Composition Facility (DCF). The AFPINDEX file is identical in content to the DCFINDEX file, so if you have already installed DCF and have run FLIP, you may not need to re-run it for AFP API. However, the listing files produced from AFPINDEX are quite different, so you may want to keep both. AFP API searches first for AFPINDEX and then for DCFINDEX. DCF does not recognize or process the AFPINDEX file.

You must use FLIP to re-create the AFPINDEX whenever a font object in the font library is modified, added, or deleted. The program directory shipped with the product contains instructions for running FLIP. Appendix A, “Font Library Indexing Program (FLIP)” contains an overview of running FLIP, in case the program directory is not available.

**AFP Font Listing:** The FLIP listing, which is illustrated in Figure 24 on page 71, specifies the contents of the library in the same terms that are used when identifying the font with the Define Font by Attributes call.

- On VM, the listing is a file named FONTxxxx LISTING, where FONTxxxx matches the file type of the font objects. The listing is stored on the same disk as the font library index.
- On MVS, the listing is a member named LISTING, which is stored in the font library.
- On VSE, the listing is named AFPINDEX and is held on the POWER LST QUEUE. The listing can be stored in the font sublibrary.

See your system programmer if you cannot locate the FLIP listing. The system programmer may need to run FLIP again, which is described in Appendix A, “Font Library Indexing Program (FLIP).”

APQFPRPT: AFP FONT LIBRARY INDEX PROGRAM REPORT. FONT LIBRARY: F O NT3820 PAGE: 47  
 For a description of this report, see the AFP API Programmers Guide section on Defining Fonts  
 FORMAT: BOUNDED BOX

DESCRIPTIVE NAME: TIMES NEW ROMAN LATIN1									
CHARACTER SET	POINT SIZE	WEIGHT	WIDTH	STYLE	DEVICE	LINE SPACE	FIGURE SPACE	WORD SPACE	
CON40000	10	Bold	Normal	Roman	3820	35	17	8	
CON50000	10	Bold	Normal	Italic	3820	35	17	8	
CON200A0	11	Medium	Normal	Roman	3820	40	18	9	
CON300A0	11	Medium	Normal	Italic	3820	40	18	9	
CON400A0	11	Bold	Normal	Roman	3820	40	18	9	
CON500A0	11	Bold	Normal	Italic	3820	40	18	9	
CON200B0	12	Medium	Normal	Roman	3820	43	20	10	
CON300B0	12	Medium	Normal	Italic	3820	43	20	10	
CON400B0	12	Bold	Normal	Roman	3820	43	20	10	
CON500B0	12	Bold	Normal	Italic	3820	43	20	10	
CON200D0	14	Medium	Normal	Roman	3820	50	23	12	
CON300D0	14	Medium	Normal	Italic	3820	50	23	12	
CON400D0	14	Bold	Normal	Roman	3820	50	23	12	
CON500D0	14	Bold	Normal	Italic	3820	50	23	12	
CON200F0	16	Medium	Normal	Roman	3820	57	27	13	
CON300F0	16	Medium	Normal	Italic	3820	57	27	13	
CON400F0	16	Bold	Normal	Roman	3820	57	27	13	
CON500F0	16	Bold	Normal	Italic	3820	57	27	13	
CON200H0	18	Medium	Normal	Roman	3820	64	30	15	
CON300H0	18	Medium	Normal	Italic	3820	64	30	15	
CON400H0	18	Bold	Normal	Roman	3820	64	30	15	
CON500H0	18	Bold	Normal	Italic	3820	64	30	15	
CON200J0	20	Medium	Normal	Roman	3820	72	33	17	
CON300J0	20	Medium	Normal	Italic	3820	72	33	17	
CON400J0	20	Bold	Normal	Roman	3820	72	33	17	
CON500J0	20	Bold	Normal	Italic	3820	72	33	17	
CON200N0	24	Medium	Normal	Roman	3820	86	40	20	
CON300N0	24	Medium	Normal	Italic	3820	86	40	20	
CON400N0	24	Bold	Normal	Roman	3820	86	40	20	
CON500N0	24	Bold	Normal	Italic	3820	86	40	20	
CON200T0	30	Medium	Normal	Roman	3820	107	50	25	
CON300T0	30	Medium	Normal	Italic	3820	107	50	25	
CON400T0	30	Bold	Normal	Roman	3820	107	50	25	
CON500T0	30	Bold	Normal	Italic	3820	107	50	25	
CON200Z0	36	Medium	Normal	Roman	3820	128	60	30	
CON300Z0	36	Medium	Normal	Italic	3820	128	60	30	
CON400Z0	36	Bold	Normal	Roman	3820	128	60	30	
CON500Z0	36	Bold	Normal	Italic	3820	128	60	30	

Figure 24. Portion of a Sample of Font Library Index Program Listing

## Fonts

The character set is not used as a parameter on Define Font, but it tells the formatter which character set in the font library to read when its attribute values are specified with the other parameters on Define Font, such as descriptive name, width, weight, and so on. Within each descriptive name group, fonts are ordered by point size, weight, width, and style. Point sizes can range from 6 through 72.

Programming constants for specifying weight, width, and style are identified in the description of the AFPDFNT call in *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference*.

The first font found (in the order shown in the font report listing) whose description matches the attributes on AFPDFNT is used. For example, if you code the following, character set CON400B0 of the font library is used.

---

```
MOVE "T1V10500" TO AFP-CODE-PAGE.
MOVE 22 TO AFP-DESC-NAME-LENGTH.
MOVE "TIMES NEW ROMAN LATIN1" to AFP-DESCRIPTIVE NAME.
MOVE 12 to AFP-POINT-SIZE.
CALL "AFPDFNT" USING
    BY CONTENT
        AFP-API-HANDLE
        AFP-CURRENT-HANDLE
        AFP-CODE-PAGE
        AFP-DESC-NAME-LENGTH
        AFP-DESCRIPTIVE-NAME
        AFP-POINT-SIZE
        BOLD
        NORMAL
        ORIENTO
        ROMAN
    BY REFERENCE
        TIM12BOLD
        AFP-RET-CODE
        AFP-SEVERITY-CODE.
```

---



## Setting Attributes (and Querying Them)

To override defaults, you must tell AFP API which units of measure to use (inches, millimeters, centimeters, 240th of an inch, or 1440ths of an inch) when placing information on a page, where to place the information, how thick to make rules, how much space to put between words and characters, and which color (if any) to use. See “Understanding States” on page 75 for more information about the attribute hierarchy. If a value is not specifically established for an attribute, the following default values apply:

<b>Attribute</b>	<b>Default</b>
Unit of measure	Millimeters.
Initial position	X position is 0; Y position is 0 (that is, the logical page origin).
Color	Black.
Rule Thickness	0.4 millimeters.
Font	Coded font X0N2100C (Times New Roman 10 point with code page T1V10500).
Intercharacter Spacing	The additional intercharacter spacing increment is 0.
Word Spacing	The word space associated with the current font.

The following procedure calls set the attributes indicated:

### AFPSUNI (Set Units):

Sets the units of measure in a document, page, or area. For example, you can specify inches in the AFPBDOC procedure call for the page width and depth for the entire document, then specify millimeters in a page for spacing and placing characters on the page. The default is millimeters.

**Note:** The output file generated by AFP API is in logical units of 1440 per inch, even if you specify a different unit of measure in the Set Units procedure call.

### AFPSPOS (Set Position):

Specifies where to place data (text, rules, and so on) with X and Y coordinates. Placement can be either relative to the current position or absolute to the data or area coordinate system origin. “AFPBDOC (Begin Document)” on page 100 describes the data coordinate system and logical-page orientation, and “More About Areas” on page 47 describes the area coordinate system.

**Relative** Place relative to the current position either in the unit of measure currently in effect or in lines of text.

**Absolute** Place at the specified position in the unit of measure currently in effect. You don’t care where you were, because you specify values for X and Y for where you want to be. When placing data or placing areas in a specified position, you can specify X and Y values that are greater than, less than, or equal to the current X and Y values; however, when placing data *within* an area, the Y value must be greater than or equal to the current Y value.

## Setting Attributes

### AFPSWSP (Set Word Spacing):

Specifies the width of spaces (X' 40') between words in the unit of measure currently in effect. The default is the word spacing associated with the current font. Whatever value you specify is used when the text contains a X' 40' (space).

### AFPSICS (Set Intercharacter Spacing):

Specifies the amount of space to insert between characters in a word in the unit of measure currently in effect. This space is in addition to the normal space for a character in a particular font. The default is to add no additional space between characters.

### AFPSCLR (Set Color):

Specifies the color for subsequent text and rules. Valid colors are black, blue, red, magenta, green, cyan, yellow, brown, and the color of the media. The default is black.

You can ask AFP API what attributes and position parameter values are in effect for the document. You can also ask AFP API what size area is required to print a specified character string using the current attributes. The procedure calls described here do that. For AFPQATT and AFPQPOS, the numeric values returned may be slightly different than the ones specified, because of rounding.

See "Format of the AFP API Procedure Call Descriptions" on page 98 for detailed parameter descriptions for these procedure calls.

### AFPQATT (Query Current Attributes):

Returns the current values for units of measure, position, color, rule thickness, font, intercharacter spacing, and word spacing.

### AFPQPOS (Query Current Position):

Returns the current X-coordinate and Y-coordinate values for position in the units of measure currently in effect. Use this procedure call instead of AFPQATT if you need only the position; response may be a little faster.

### AFPQSTR (Query Character String Size):

Returns the width and depth of the specified character string printed in the current font. The width and depth is returned in the current unit of measure, using the font, intercharacter spacing, and word spacing attributes currently in effect.

## Understanding States and Handles

Now that you've read about most of the AFP API parts, you need to put them together in a hierarchy (states) and keep track of where you are in the hierarchy (handles).

## Understanding States

AFP API is *state* driven. This means that AFP API needs to know at all times where in the document it is operating, so that it can determine which attributes are in effect. AFP API uses states to do this. Figure 25 shows all of the AFP API states and their relationship to each other.

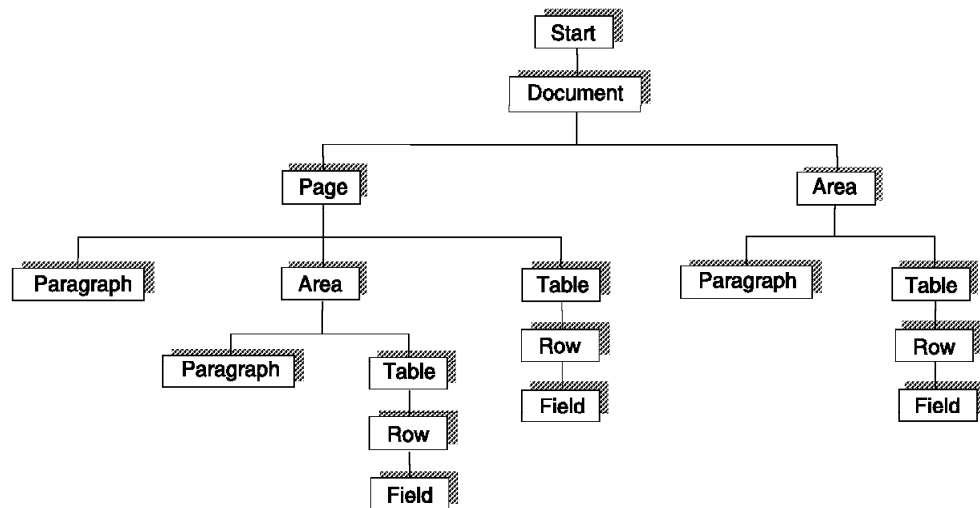


Figure 25. Hierarchy of AFP API States

When you issue a AFP API procedure call that begins a state (AFPINIT, AFPBDOC, AFPBPAG, AFPCARE, AFPBPAR, AFPBTBL, AFPBROW, or AFPBFLD), AFP API enters a new state. When you issue a AFP API procedure call that ends the state (AFPEND, AFPTERM, AFPEDOC, AFPEPAG, AFPEARE, AFPEPAR, AFPETBL, AFPEROW, or AFPEFLD), AFP API returns to the previous state. When backing out, you must keep track of where AFP API is. For example, when coming out of an area, AFP API can be in either document state or page state. The state in which AFP API is determines your next action. When you end a document (AFPEDOC), AFP API enters start state; when you terminate a document (AFPTERM) or end AFP API (AFPEND), the AFP API session ends.

Other characteristics about states are:

- Calls made in one state set the defaults for lower states. For example, units set in document state apply to pages and areas in that document, and fonts set in page state apply to an area, table, or paragraph in that page.
- Calls made in lower states override defaults. For example, AFPSUNI in page state overrides the units set in document state.
- Calls affect current and lower states only. For example, AFPSUNI in paragraph state does not affect units in page state.

When AFP API is initialized and a document is created (see “ AFPINIT (Initialize AFP API)” on page 151 and “ AFPBDOC (Begin Document)” on page 100), the program changes to document state. In document state, a default environment is established for each page within the document, and you can create one or more pages in that document. If a default is not specifically established for an attribute, the defaults listed in “Setting Attributes (and Querying Them)” on page 73 apply.

When a page is created (see “ AFPBPAG (Begin Page)” on page 108), the program changes to page state. Each page inherits its initial environment from the parent document. You can override the default environment within a page, but when the page ends, the program returns to document state, and the document environment is reestablished for subsequent pages within the document.

You can create an area in either document or page state, and several areas can be active at the same time. An area provides a way of defining an autonomous object that is in AFP API storage. You can place the area on one or more pages by referencing the area within page state. Creating an area is independent of the state in which it was created, except that its initial default environment is inherited from the current environment. You can override the default environment within the area, but when the area ends, the current environment is reestablished.

## Understanding Handles

Handles are IDs that identify the session and the portion (state) of the document in which AFP API is operating. AFP API uses these handles to keep track of which state in the document it is processing at any given time.

The AFP API handle is required on all calls to identify the program to AFP API. You learned in “Understanding States” on page 75 that when you issue a AFP API procedure call that begins a state (AFPINIT, AFPBDOC, AFPBPAG, AFPCARE, AFPBPAR, AFPBTBL, AFPBROW, or AFPBFLD), you enter a new state. When AFP API enters a state, except for row state and field state, it assigns an identifier or *handle* to that state and sends that handle back to the program; in row state and in field state, the table handle is used. Your program must place the handle of the current state in the Current-Handle field on AFP API procedure calls.

The handles for a simple document look like Figure 26 on page 78. When you issue the AFPINIT call, AFP API returns the AFP API handle back to the program, and this becomes the handle for the session and is required in all calls for that session. When you issue the AFPBDOC call, AFP API returns document handle back to the program. The next call in the document state, in this case the AFPBPAG, includes the AFP API handle and the document handle, and AFP API returns the page handle back to the program. The next call in the page state, in this case the AFPBPAR, includes the AFP API handle and the page handle, and AFP API returns the paragraph handle to the program. The next call in the paragraph state, in this case the AFPPTXT, includes the AFP API handle and the paragraph handle.

When you use a procedure call that ends a state (AFPEND, AFPTERM, AFPEDOC, AFPEPAG, AFPEARE, AFPEPAR, AFPETBL, AFPEROW, or AFPEFLD), you use all of the handles that got you where you are, and the program goes back to the previous state. For example, when you end the paragraph (AFPEPAR) in Figure 26 on page 78, you include the AFP API handle and the paragraph handle; when you end the page, you include the AFP API handle and the page handle.

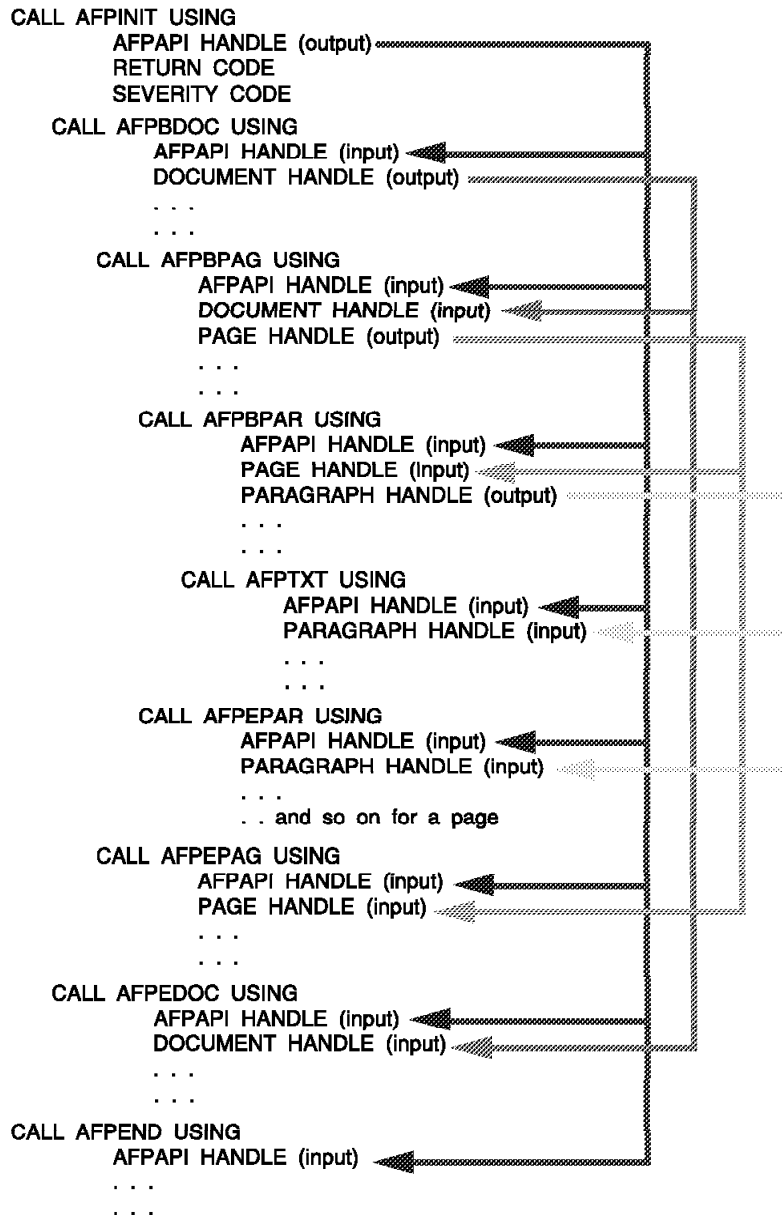


Figure 26. An Example Showing the Use of Handles

Chapter 3, “Procedure Call Reference” also uses the term *current handle*. The current handle identifies the state from which a call is made. The handle is returned from the call that initiated that state. For example, in the AFPBPAR call in Figure 26, the current handle is the page handle received from the AFPBPAG call. If this AFPBPAR had been in an area, the current handle would have been the area handle. The application program must set the appropriate current handle prior to invoking any AFP API “Begin” procedure calls.

## Indexing Data for Viewing and Archiving

As described in “Indexing AFP Data for Viewing and Archiving” on page 14, you can use AFP API to place indexing information in your document. This section describes the AFP API procedure calls for indexing your document.

### What’s Involved?

Three IBM programs are involved in generating and using indexing information: AFP API, AFP Conversion and Indexing Facility (ACIF), and AFP Workbench for Windows, the Viewer Application.

- AFP API: Can insert three kinds of indexing information in a document:
  - Group boundaries
  - Group-level indexing tags
  - Page-level indexing tags
- ACIF: Can insert two kinds of indexing information in a document (if the input document does not already contain any indexing information):
  - Group boundaries
  - Group-level indexing tags

Also, ACIF can create an *index object file* that identifies the locations of all of the groups and indexing tags in a document.

- Viewer Application: Uses the index object file and indexing information to facilitate navigation through an online document.

This section describes by example how to insert tags into the document using AFP API. For creating the index object file, refer to *AFP Conversion and Indexing Facility: Application Programming Guide*. For information about how the indexing information is used when navigating through a document, refer to either *AFP Workbench for Windows: Using the Viewer Application* or the help screens for Viewer.

### What Are the Indexing Procedure Calls?

The AFP API procedure calls that insert indexing information into the document are AFPBGRP (Begin Group), AFPEGRP (End Group), and AFPPTAG (Put Tag). The AFPBGRP and AFPEGRP procedure calls identify the boundaries of a *group*, which is a named collection of sequential pages. A group name should be unique within a document.

In addition to defining groups, you can use the AFPPTAG procedure call to add a group-level indexing tag inside a defined group. The Attribute Name and Attribute Value parameters of the AFPPTAG procedure call are used to define the content of the indexing tag. You can use multiple tags to define a single group, such as both the Customer Name and the Account Number, to provide more than one way to identify the group. With AFP Workbench for Windows, the Viewer Application, you can locate a group of pages by selecting the attribute name and value of the indexing tag associated with the group.

You can use the AFPPTAG procedure call to produce indexing tags that identify a single page in the document. For example, to locate the summary page of a multi-page statement, you can define each statement in the document as a group, tag the group with the customer’s name and account number, and tag the Summary page within the group. With AFP Workbench for Windows, the Viewer

## Indexing

Application, you can use this page-level tag to locate the summary page in a statement after locating the customer's statement as a group in the document.

When the AFPPTAG procedure call is issued in page state, the tag is valid without an enclosing group. When the AFPPUT procedure call is issued in document state, group-level indexing is required, and the tag is not valid without an enclosing group. Nested groups are not supported (for example, an AFPBGRP followed by another AFPBGRP). Basically, the code is as follows for generating group-level indexing information for the sample application:

Code	Description
*-----*	
* PROCESS INDEXING *	
*-----*	
<pre> PROCESS-INDEXING   MOVE ACCOUNT-NUM-OUT TO AFP-GROUP-NAME   CALL  "AFPBGRP" USING         BY CONTENT         AFP-API-HANDLE         AFP-DOCUMENT-HANDLE         AFP-GROUP-NAME         BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE.    MOVE "LAST-NAME" TO AFP-ATTRIBUTE-NAME   MOVE CUST-NAME TO AFP-ATTRIBUTE-VALUE   CALL  "AFPPTAG" USING         BY CONTENT         AFP-API-HANDLE         AFP-CURRENT-HANDLE         AFP-TAG-NAME         AFP-TAG-VALUE         BY REFERENCE         AFP-RET-CODE         AFP-SEVERITY-CODE.    CALL  "AFBPBAG" USING         BY CONTENT         AFP-API-HANDLE         AFP-DOCUMENT-HANDLE         ...         BY REFERENCE         AFP-PAGE-HANDLE         AFP-RET-CODE         AFP-SEVERITY-CODE.    CALL  "AFPEPAG" USING         BY CONTENT         AFP-API-HANDLE         AFP-CURRENT-HANDLE         ...         BY REFERENCE         AFP-PAGE-HANDLE         AFP-RET-CODE         AFP-SEVERITY-CODE.    CALL  "AFBPBAG" USING         BY CONTENT         AFP-API-HANDLE         AFP-DOCUMENT-HANDLE         ...         BY REFERENCE         AFP-PAGE-HANDLE         AFP-RET-CODE         AFP-SEVERITY-CODE. </pre>	<p><b>AFPBGRP (Begin Group):</b> Identifies the beginning of a named group of pages in the document. The name of the group can be any unique identifier. In this case, it is Susan's Primo charge account number. AFPBGRP is valid only between pages.</p> <p><b>AFPPTAG (Put Tag):</b> Inserts an indexing tag for the group of pages. In this case, the attribute name is LAST NAME, and the attribute value is the customer's name (AMES). AFPPTAG is valid either in a group or in a page.</p> <p><b>AFBPBAG (Begin Page):</b> Represents page 1 of the sample for Susan's statement.</p> <p><b>AFPEPAG (End Page)</b> Ends page 1 of the sample for Susan's statement.</p> <p><b>AFBPBAG (Begin Page):</b> Represents page 2 of the sample for Susan's statement.</p>



Code	Description
<pre>CALL "AFPEPAG" USING       BY CONTENT       AFP-API-HANDLE       AFP-CURRENT-HANDLE       ...       BY REFERENCE       AFP-PAGE-HANDLE       AFP-RET-CODE       AFP-SEVERITY-CODE.</pre>	<p>AFPEPAG (End Page): Ends the collection of pages for Susan's statement.</p>
<pre>MOVE ACCOUNT-NUM-OUT TO AFP-GROUP-NAME CALL "AFPEGRP" USING       BY CONTENT       AFP-API-HANDLE       AFP-DOCUMENT-HANDLE       AFP-GROUP-NAME       BY REFERENCE       AFP-RET-CODE       AFP-SEVERITY-CODE.</pre>	<p>AFPEGRP (End Group): Ends the group having the group name ACCOUNT-NUM-OUT. AFPEGRP is valid only between pages.</p>

## Determining Page Breaks and Changing Page Layout

As you've seen, an important function of AFP API is to place variable information (names, account information, and so on) on a page. Figure 3 on page 8 describes this.

AFP API also allows you to create different page layouts in the same document. That is, page 2 can have a different header or bottom margin or different artwork than page 1, as shown in Figure 27.

The code won't be shown here; this section will describe the approach used to create the page break and the different page layouts for pages 1 and 2 of the example. *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference* show the complete, actual code. Remember that the actual COBOL and PL/1 code to create it is on the tape shipped with the product.

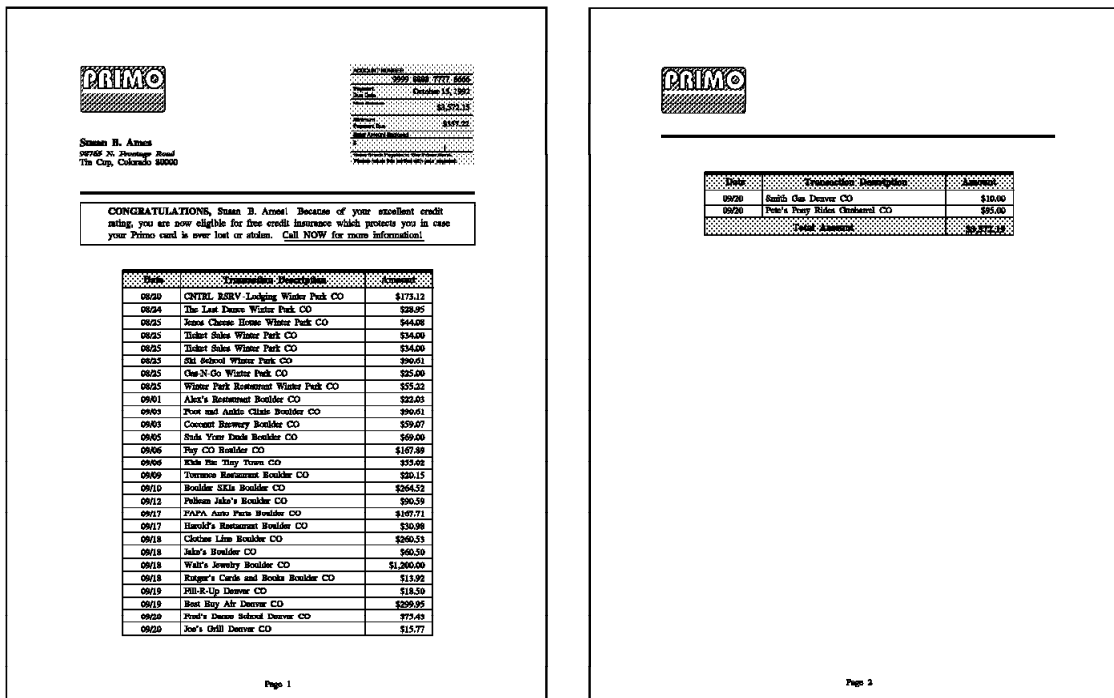


Figure 27. Sample Document with Different Page Layouts

The goals for this part of the example are:

- Place as much data as possible on page 1.
- Place the remaining data on page 2.
- Modify the layout of page 2.

Here is a brief overview of the steps to accomplish this, followed by a description of the details.

1. Conceptually break the pages into three parts: header, body, and footer, as shown in Figure 28 on page 83.
2. Determine the size (depth in millimeters) of the page body as the page depth less the header and footer depth.

3. Subtract the depth for each piece of data in the body from the current body depth in order to determine the remaining space.
4. Determine when the data in the body approaches the footer and prepare to put the remaining data on a new page.
5. Calculate the size of the body on page 2 and begin printing the remainder of the table in the body of page 2.

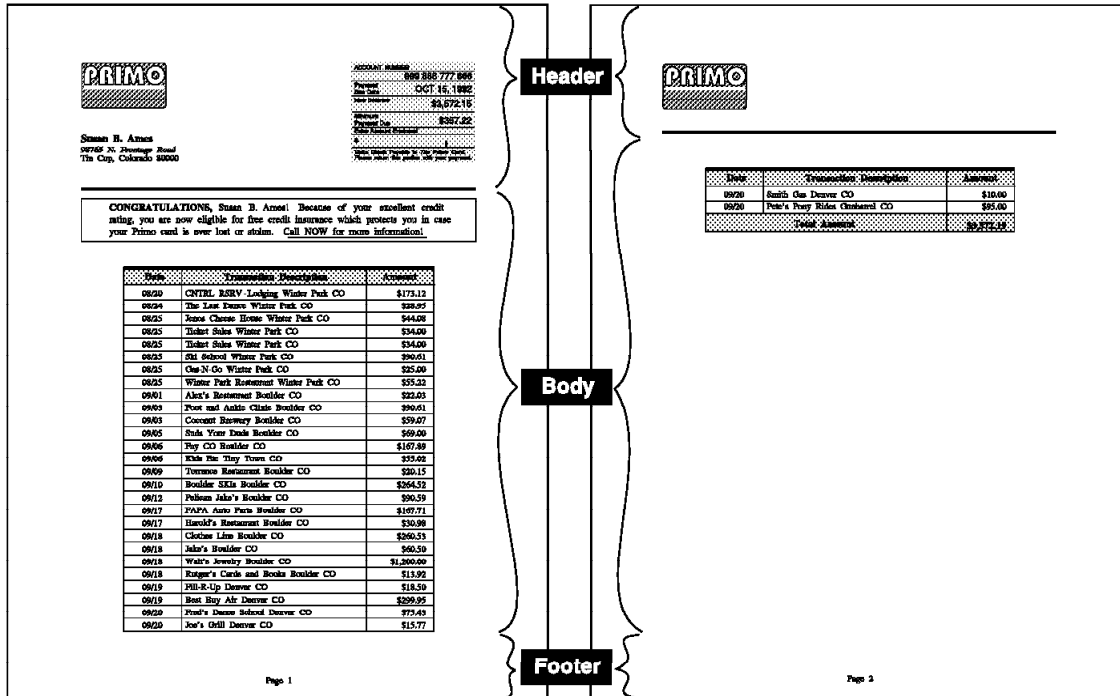


Figure 28. Coding the Sample Document for Page Breaks. The source code in AFP Application Programming Interface: COBOL Language Reference and AFP Application Programming Interface: PL/1 Language Reference shows the code for calculating the space on a page and then ejecting to a new page.

Here are the details of steps 1–5 above:

**Header, Body, and Footer:** The headers for both pages are different, but after being established, their size remains constant when data is put into them. The footers for both pages are the same, and after being established, their size remains constant as data is put into them. The bodies for both pages vary as data is written into them.

As you write data into the body, subtract the depth of the data from the depth of the body. Here are the steps:

1. For Page 1, make the footer 20 millimeters deep.
2. Write the header, including the horizontal rule and white space, after the horizontal rule.
3. Write the paragraph, which is small and will not print off the page.
4. After the paragraph, calculate the remaining body space, subtracting the paragraph size and white space preceding the table from the page body. Specify this as the maximum depth of the table on the AFPBTBL call. As you write data into the table, one of two things happens:

## Invoke Medium Map

- The data does not exceed the maximum depth specified. In this case, write another row of data.

**Note:** Another way to do this is to look at the parameter AFP API returns that contains the actual depth of the table after each AFPEROW. With this depth, calculate the space left in the body, begin another data element, and repeat the process until the page is full.

- The data exceeds the maximum depth specified. In this case, AFP API returns a WARNING severity code on AFPEROW. At this point, end the table and write the footer for the first page, start a new page, write the header for the new page, and continue writing data on page 2.
5. For page 2, monitor the severity code of the AFPEROW call in the same way as for page 1. In this example, the data ends before reaching the footer.

To summarize, you can use AFP API to format pages in the following ways:

- Areas and tables: You can specify a maximum depth on the AFPCARE call and on the AFPBTBL call. As data approaches the maximum depth, AFP API issues a warning in a severity code, and you know to stop writing data and begin a new page. The table is described in the sample document.
- Character strings: Issue AFPQPOS after each AFPPCHS to determine where on the page you are, then calculate the remaining space.
- Paragraphs: AFP API returns the actual depth of the paragraph on the AFPEPAR call. You can also put paragraphs in an area, so that you can specify a maximum depth of the area. In this case, AFP API issues a WARNING severity code when you are at the bottom of the area.

---

## Specifying Presentation Options

You can specify presentation options to use for the pages in your document. To do this, reference a medium map (also called a copy group) that is defined in a resource called a form definition. Although the example does not contain an AFPINVM procedure call, the example shows using the call to reference a typical medium map. See “Form Definitions” on page 13 for more information about medium maps. See “Format of the AFP API Procedure Call Descriptions” on page 98 for detailed parameter descriptions.

To reference a form definition in a document, specify the name of the medium map, as shown in the sample code:

Code	Description
*-----*	
* Invoke Medium Map	*
*-----*	
MOVE f1XXXXXX TO AFP-MEDIUM-MAP. CALL "AFPINVM" USING BY CONTENT AFPAPI-HANDLE AFP-DOCUMENT-HANDLE AFP-MEDIUM-MAP BY REFERENCE AFP-RET-CODE AFP-SEVERITY-CODE.	AFPINVM (Invoke Medium Map): Invokes the medium map (copy group) specified. Medium maps are in form definitions. AFPINVM is valid only between pages, because it selects presentation options for subsequent pages.

## Using AFP API in a CICS/ESA Environment

You write a program that calls AFP API in a CICS/ESA environment in much the same way you write any other program that calls AFP API. You use the COBOL programming language, running the program under the MVS operating system.

Be aware, however, of differences in the following areas when using AFP API in a CICS/ESA environment:

- Defining the temporary storage queue for AFP output
- Using IOCA and GOCA objects
- Creating font and page segment data sets
- Using the error-checking routine in APQPERF
- Link-editing your CICS/ESA program with AFP API

## Defining the Temporary Storage Queue for AFP Output

Instead of writing AFP output to an output file, in a CICS/ESA environment, AFP API writes output to a temporary storage queue. You can then use the CEBR PUT command to copy the contents of the temporary storage queue to a transient data queue, from which the data can be printed using TSO or a batch job.

Your program must manage the temporary storage queue, as follows:

- Name the temporary storage queue in the Set Output Characteristics (AFPSOUT) procedure call.
- Purge the temporary storage queue as necessary.

Instead of writing the AFP output to a CICS/ESA temporary storage queue, AFP API can return the AFP output to a buffer in your program. Output buffering allows you to inspect and modify the AFP output. When you use output buffering, your program is responsible for writing the output to a temporary storage queue or another output data set. See “Buffering AFP API Output” on page 66 for a description of output buffering.

AFP API provides two sample COBOL programs for a CICS/ESA environment. The transaction names for these sample programs are:

- APQS, which writes AFP output to a CICS/ESA temporary storage queue. The source code for the program is provided in APQCISMP.
- APQB, which writes AFP output to a buffer in the program. The program then writes the AFP output to a CICS/ESA temporary storage queue. The source code for the program is provided in APQCISMB.

These sample COBOL programs are printed in *AFP Application Programming Interface: COBOL Language Reference*.

## Using IOCA and GOCA Objects

In a CICS/ESA environment, AFP API does not support including IOCA and GOCA objects as individual objects; therefore, you cannot use the Include Object (AFPIOBJ) procedure call. You can, however, use the Include Page Segment (AFPIPSG) procedure call to include a page segment that contains an IOCA or GOCA object. Page segments can be created using programs such as the Graphical Data Display Manager (GDDM).

## Creating VSAM Data Sets for Fonts and Page Segments

Fonts and page segments must be located in key-sequenced VSAM data sets that are defined to CICS/ESA with file names FONTLIB and SEGLIB.

AFP API provides program APQCIVSM and associated JCL to copy an existing font or page segment partitioned data set (PDS) into a VSAM data set. Your system programmer must modify and run the JCL in APQCIFON and APQCISEG at installation and whenever the font and page segment PDSs are modified.

Because AFP API reads fonts and page segments from VSAM data sets with standard names, you do not need to define resource library names with the Set Resource Library Names (AFPSLIB) procedure call. AFP API ignores the AFPSLIB procedure call if it occurs in your program.

## Using the Error-Checking Routine in APQPERF

The error-checking routine, CHKSUCC, in APQPERF contains modifications for a CICS/ESA environment; however these modifications are commented out in the version of APQPERF distributed with AFP API. Therefore, before using the COBOL paragraphs in APQPERF, follow the instructions in the comments of the CHKSUCC routine to make the necessary modifications.

## Link-Editing Your Program with AFP API

When you link-edit your program with AFP API code, you must specify INCLUDE statements to include the AFP API code for a CICS/ESA environment. See “MVS JCL for Compiling and Link-Editing a COBOL Application in a CICS/ESA Environment” on page 273 for a description of the required INCLUDE statements.

---

## Improving Performance

You may be able to improve the performance of AFP API by following these suggestions:

- Ensure that AFP API is at the current service level. Several problems have been corrected involving storage and processing of areas and tables. You can request a service update from IBM Service (1-800-237-5511) if you are not sure what level you have installed.
- Use AFP API only if your application cannot be performed with a page definition.
- Use multiple Begin Document (AFPBD) and End Document (AFPED) procedure calls or multiple Initialize AFP API (AFPINIT) and End AFP API (AFPEND) procedure calls only when necessary. Instead, you can define multiple groups of pages in a single document using the Begin Group (AFPBGRP) and End Group (AFPEG) procedure calls.
- Create areas only when necessary and delete them when they are no longer needed. Because you can now place character data on a page in any order, you no longer have to create areas for this purpose.
- Create tables only when necessary. Tables provide powerful formatting functions for vertically aligning related text, but they require a large amount of overhead during formatting time. You can often use the Put Character String (AFPPCHS) procedure call with alignment options to simulate tables of data.
- Define rows and fields only when necessary. Reuse common row and field definitions whenever possible.
- Define fonts and areas in the document state rather than in the page state.
- Define only those fonts, areas, fields, and rows that your program actually uses, because AFP API performs a linear search of these definitions whenever it processes the object.
- Use the character-alignment option on the Put Character String (AFPPCHS) procedure call only when necessary. Often, the right-alignment option produces the same result more efficiently.

For example, to align columns of data at a decimal point when the data contains exactly the same number of decimal places, you can use the right-alignment option. If you use the right-alignment option instead of the character-alignment options, AFP API does not need to search for the decimal point in each string.

### Coding Tips

This section contains tips for coding your program, based on problems customers have reported:

- Issue the Define Row (AFPDROW) and Define Field (AFPDFLD) procedure calls only in the document state. You can no longer issue the AFPDROW and AFPDFLD calls in other states.
- Issue the Put Area (AFPPARE) procedure call only in page state. This means that you *cannot* create an area that contains another area. Instead of creating and putting a nested area, you can put several areas on the same page with separate Put Area procedure calls.
- Fractional (non-integer) point sizes are not supported.
- Do not request an ASCII code page on the Define Font by Attributes (AFPDFNT) procedure call. ASCII fonts are not supported.
- Specify a descriptive name on the Define Font by Attributes (AFPDFNT) procedure call that matches *exactly* the descriptive name in the FLIP listing. Notice that the descriptive names are case sensitive.
- Ensure that the default coded font (X0N2100C) exists as member X0N2100C in your font library even if you are not using the default font. This coded font can point to any valid character set and code page combination that you want, but the member name must be X0N2100C.  
  
If you are formatting for a 3800 printer, copy a valid coded font member with zero rotation and rename it to X0N2100C. A coded font member with zero rotation begins with X1.
- Ensure that all character sets have unique typeface and attribute combinations in the FLIP listing. AFP API uses the first character set in the FLIP listing that has the descriptive name and attributes specified on the Define Font by Attributes (AFPDFNT) procedure call.
- Null characters (X'00') are not permitted in character strings in AFP API procedure calls.



## Troubleshooting Your Program

This section contains information to help you debug your program. It addresses the following areas:

- Debugging errors in your application program
- Modifying the error-checking routine provided with AFP API

## Debugging Errors in Your Application Program

AFP API sends return codes and severity codes back to the program after every procedure call. Your program should check these codes after every call and respond as suggested in Appendix B, “Return Codes and Severity Codes” Checking the return codes aids in debugging your program.

Some errors in your application program may cause your program to terminate abnormally or result in incorrect formatting of the AFP output. Table 1 describes some of the more common errors of this type.

Error	Solutions
Abend S0C1 occurs with message IEC135 DD STATEMENT MISSING when you use the buffered-output function or discard the output.	Use the latest APQCONST or APQPCON copy books to select the BUFFERED or DISCBUFF constant on the Set Output Characteristics (AFPSOUT) procedure call. Notice that this constant value is case sensitive. Also, be sure to specify the <i>constant</i> BUFFERED or DISCBUFF, not the <i>string</i> BUFFERED or DISCBUFF.
You receive message IEW2456E when using the JCL in APQCOCOB or APQCOPLI to compile your application program.	Ensure that you are using the latest level of APQCOCOB or APQCOPLI. The old level contains INCLUDE statements for modules in the “C” runtime library: EDCXGET, EDCXFREE, EDCXLOAD, EDCXSRVI, and EDCXSRVN. The current level contains INCLUDE statements for AFP API modules instead: APQXGET, APQXFREE, APQXLOAD, APQXSRVI, APQXSRVN, and APQXUNLD.
An unexpected line break occurs when you change the font in a paragraph or table.	Set the Concatenate parameter to TRU the first time you invoke the Put Text (AFPPTXT) procedure call in each field and paragraph.
An expected page break does not occur when you use the Invoke Medium Map (AFPINVM) procedure call.	Ensure that the medium map named on the Invoke Medium Map (AFPINVM) procedure call is correct. If the medium map named is the <i>same</i> as the medium map named on the previous AFPINVM procedure call, AFP API ignores the AFPINVM call.

### Modifying the Error-Checking Routine Supplied with AFP API

The AFP API Installation Verification Programs (IVPs) and the AFP API sample programs perform error checking after most of the AFP API procedure calls. These programs use the CHKSUCC error-checking routine provided in APQPERF (for COBOL programs) and APQPPRF (for PL/1 programs).

CHKSUCC displays an error message and terminates when the severity code returned by AFP API is greater than 4, because a severity code greater than 4 when running the IVPs indicates that the installation of AFP API did not complete successfully. If you use the routines provided in either APQPERF or APQPPRF, and if you want your application to handle severity codes of 8 or less, modify CHKSUCC to allow severity codes less than or equal to 8.

**Note:** Modify the CHKSUCC routine in APQPERF before using it in a CICS/ESA environment. Follow the instructions in the comments of the CHKSUCC routine.

---

## Chapter 3. Procedure Call Reference

<b>Chapter 3. Procedure Call Reference</b> .....	93
The Application Programming Interface Program .....	93
Format of the AFP API Procedure Call Descriptions .....	98
AFPBDOC (Begin Document) .....	100
AFPBFLD (Begin Field) .....	104
AFPBGRP (Begin Group) .....	106
AFPBPAG (Begin Page) .....	108
AFPBPARG (Begin Paragraph) .....	112
AFPBROW (Begin Row) .....	116
AFPBTBL (Begin Table) .....	118
AFPCARE (Create Area) .....	121
AFPDFLD (Define Field) .....	124
AFPDFNT (Define Font by Attributes) .....	128
AFPDROW (Define Row) .....	131
AFPEARE (End Area) .....	134
AFPEDOC (End Document) .....	136
AFPEFLD (End Field) .....	137
AFPEGRP (End Group) .....	138
AFPEND (End AFP API) .....	140
AFPEPAG (End Page) .....	141
AFPEPAR (End Paragraph) .....	143
AFPEROW (End Row) .....	145
AFPETBL (End Table) .....	147
AFPGBUF (Get Output Buffer) .....	149
AFPINIT (Initialize AFP API) .....	151
AFPINVM (Invoke Medium Map) .....	152
AFPIOBJ (Include Object) .....	154
AFPIOVL (Include Page Overlay) .....	158
AFPIPSG (Include Page Segment) .....	160
AFPPARE (Put Area) .....	162
AFPPBOX (Put Box) .....	164
AFPPCHS (Put Character String) .....	166
AFPPRUL (Put Rule) .....	169
AFPPTAG (Put Tag) .....	171
AFPPTXT (Put Text) .....	173
AFPQATT (Query Current Attributes) .....	175
AFPQPOS (Query Current Position) .....	177
AFPQSTR (Query Character String Size) .....	179
AFPSCLR (Set Color) .....	181
AFPSFNT (Set Font) .....	183
AFPSICS (Set Intercharacter Spacing) .....	185
AFPSLIB (Set Resource Library Names) .....	187
AFPSOUT (Set Output Characteristics) .....	190
AFPSPOS (Set Position) .....	193
AFPSRTH (Set Rule Thickness) .....	195
AFPSUNI (Set Units) .....	197
AFPSWSP (Set Word Spacing) .....	199
AFPTERM (Terminate AFP API) .....	201
AFPXARE (Destroy Area) .....	202



---

## Chapter 3. Procedure Call Reference

This chapter contains reference information for the AFP API procedure calls. All of the parameters for the procedure calls are required and must be in the order shown. If all parameters are not present, an addressing exception may occur. The files that contain the COBOL and PL/1<sup>6</sup> source code for the examples shown in Chapter 2, "Using AFP API" and the copy files that contain the constants and variables in the code are shipped with the AFP API product. The listings for the code are printed in *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference*.

---

### The Application Programming Interface Program

The AFP API program is state driven. Figure 29 shows the hierarchy of valid states in AFP API (except for final state). Table 2 on page 94 shows the procedure calls that are valid in each state.

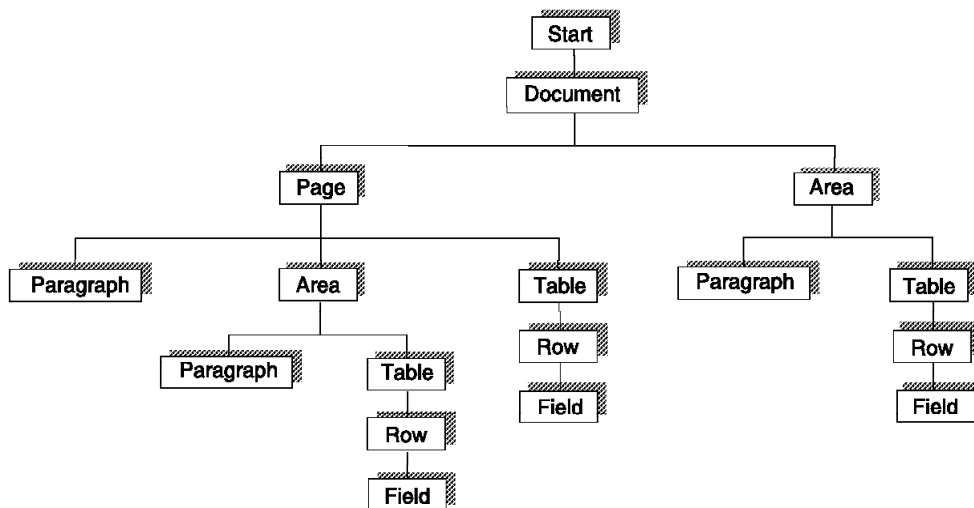


Figure 29. The Hierarchy of States in AFP API

After successfully executing AFP API initialization, the program is in start state, and the AFP API session is assigned a *handle* identifier. From start state, you can create a document.

---

<sup>6</sup>

You can use the PL/1 programming language only on the VM and MVS operating systems, in a non-CICS environment.

Table 2 shows the AFP API states and valid procedure calls in each state.

<i>Table 2 (Page 1 of 4). AFP API State Diagram</i>		
<b>Initial State</b>	<b>Function</b>	<b>Resultant State</b>
<i>Start State</i>	Begin Document	Document State
	End AFP API	Final State
	Set Output Characteristics	Start State
	Set Resource Library Names	Start State
	Terminate AFP API	Final State
<i>Final State</i>	Initialize AFP API	Start State
<i>Document State</i>	Begin Group	Document State
	Begin Page	Page State
	Create Area	Area State
	Define Field	Document State
	Define Font by Attributes	Document State
	Define Row	Document State
	Destroy Area	Document State
	End Document	Start State
	End Group	Document State
	Get Output Buffer	Document State
	Invoke Medium Map	Document State
	Put Tag	Document State
	Query Current Attributes	Document State
	Set Color	Document State
	Set Font	Document State
	Set Intercharacter Spacing	Document State
	Set Rule Thickness	Document State
	Set Units	Document State
	Set Word Spacing	Document State
	Terminate AFP API	Final State

Table 2 (Page 2 of 4). AFP API State Diagram

Initial State	Function	Resultant State
<i>Page State</i>	Begin Paragraph	Paragraph State
	Begin Table	Table State
	Create Area	Area State
	Define Font by Attributes	Page State
	Destroy Area	Page State
	End Page	Document State
	Include Object	Page State
	Include Page Overlay	Page State
	Include Page Segment	Page State
	Put Area	Page State
	Put Box	Page State
	Put Character String	Page State
	Put Rule	Page State
	Put Tag	Page State
	Query Current Attributes	Page State
	Query Current Position	Page State
	Query Character String Size	Page State
	Set Color	Page State
	Set Font	Page State
	Set Intercharacter Spacing	Page State
	Set Position	Page State
	Set Rule Thickness	Page State
	Set Units	Page State
	Set Word Spacing	Page State
Terminate AFP API	Final State	

Table 2 (Page 3 of 4). AFP API State Diagram

Initial State	Function	Resultant State
<i>Area State</i>	Begin Paragraph	Paragraph State
	Begin Table	Table State
	Define Font by Attributes	Area State
	End Area	Return to Invoking State
	Include Object	Area State
	Include Page Overlay	Area State
	Include Page Segment	Area State
	Put Box	Area State
	Put Character String	Area State
	Put Rule	Area State
	Query Current Attributes	Area State
	Query Current Position	Area State
	Query Character String Size	Area State
	Set Color	Area State
	Set Font	Area State
	Set Intercharacter Spacing	Area State
	Set Position	Area State
	Set Rule Thickness	Area State
	Set Units	Area State
	Set Word Spacing	Area State
Terminate AFP API	Final State	
<i>Paragraph State</i>	End Paragraph	Return to Invoking State
	Put Text	Paragraph State
	Query Character String Size	Paragraph State
	Set Color	Paragraph State
	Set Font	Paragraph State
	Set Intercharacter Spacing	Paragraph State
	Set Word Spacing	Paragraph State
	Terminate AFP API	Final State
<i>Table State</i>	Begin Row	Row State
	End Table	Return to Invoking State
	Terminate AFP API	Final State
<i>Row State</i>	Begin Field	Field State
	End Row	Table State
	Terminate AFP API	Final State



<i>Table 2 (Page 4 of 4). AFP API State Diagram</i>		
<b>Initial State</b>	<b>Function</b>	<b>Resultant State</b>
<i>Field State</i>	End Field	Row State
	Put Character String	Field State
	Put Text	Field State
	Query Character String Size	Field State
	Set Color	Field State
	Set Font	Field State
	Set Intercharacter Spacing	Field State
	Set Word Spacing	Field State
	Terminate AFP API	Final State

**Note:** AFP API does not assign a unique handle to the row and field states; therefore, you must use the table handle as the current handle for the row and field states on AFP API procedure calls.

---

## Format of the AFP API Procedure Call Descriptions

Procedure Call descriptions are listed in alphabetical order. “Getting Started” on page 23 gives the sequence for using procedure calls. Each procedure call description contains these parts:

<b>Procedure Call Name</b>	Specifies the AFP API procedure call name and the natural language command name (in parentheses).
<b>Function</b>	Describes the procedure call’s function.
<b>Syntax</b>	Specifies the syntax for the procedure call. Procedure Call parameters must be in the order given, and all are required. If all are not present, an addressing exception may occur. Refer to <i>AFP Application Programming Interface: COBOL Language Reference</i> and <i>AFP Application Programming Interface: PL/1 Language Reference</i> for a description of the syntax for your programming language.
<b>Input Parameters</b>	Briefly describes each parameter and its function. The parameter description also tells how to use the parameter.
<b>Output Parameters</b>	Handles (when appropriate) return codes and severity codes for the procedure call, which indicate whether the call was successful and whether the calling program should examine them. See Appendix B, “Return Codes and Severity Codes” for descriptions of these codes.

This section defines the parameter data types for each procedure call without regard to a specific high-level language. The data types are:

<b>HANDLE</b>	An AFP API-assigned, 4-byte, unsigned integer for an AFP API session and for the document, table, area, page, or paragraph in a session
<b>TOKEN</b>	An 8-byte character string
<b>FILENAME-TOKEN</b>	A string of characters containing: <ul style="list-style-type: none"><li>• For MVS, the name of a DD statement or a CICS temporary storage queue</li><li>• For VM, the file name</li><li>• For VSE, the name specified in the DLBL JCL statement that identifies a file</li></ul>
<b>FILETYPE-TOKEN</b>	For VM, a string of characters containing the file type
<b>FILEMODE-TOKEN</b>	For VM, a string of characters containing the file mode
<b>BOOLEAN</b>	A 4-byte, unsigned integer with a value of 0 or 1
<b>STRING</b>	A string of single-byte, unsigned characters

<b>CHARACTER</b>	A single-byte, unsigned character
<b>ADDRESS</b>	A 4-byte, unsigned integer containing an address
<b>REAL</b>	A floating-point number greater than or equal to 0
<b>SREAL</b>	A positive or negative floating-point number
<b>INTEGER4</b>	A 4-byte, unsigned integer
<b>SGLARRAY</b>	A single-dimension array of REALs
<b>MULTARRAY</b>	An n X m array of REALs, where n and m are specified in INTEGER parameters

### AFPBDOC (Begin Document)

#### Function

Begins a document and specifies the default unit of measure, page dimensions, and orientation for a logical page. These attributes apply to every page in the document unless they are overridden on individual Begin Page procedure calls.

#### Syntax

```
AFPBDOC(  
    HANDLE    AFPAPI-handle,  
    INTEGER4  unit-of-measure,  
    REAL      doc-page-width,  
    REAL      doc-page-depth,  
    INTEGER4  page-orientation,  
    HANDLE    document-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 30. Format of the AFPBDOC Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Unit of Measure

The unit of measure for the document; that is, inches, millimeters, centimeters, 240ths of an inch, or 1440ths of an inch.

**Note:** The output file generated by AFP API is in logical units of 1440 per inch, even if you specify a different unit of measure in this parameter.

##### Document Logical Page Width

The page width in the specified unit of measure.

##### Document Logical Page Depth

The page depth in the specified unit of measure.

##### Logical Page Orientation

The orientation (rotation) of the top of the data relative to the top of the media. The top of the logical page is the side associated with the page width. The top of the media depends on the printer that is used. Refer to *Advanced Function Presentation: Printer Information* for a description of the default media origin for AFP printers.

For example:

- If a page orientation of  $0^\circ$  is specified, the top of the data coincides with the top of the media, as shown in Figure 31.
- If a page orientation of  $90^\circ$  is specified, the top of the data is rotated  $90^\circ$  in the clockwise direction with respect to the top of the media, but the top of the logical page remains the side associated with the width, as shown in Figure 32 on page 102.

AFPSPOS always sets the position with respect to the top of the data.

Figure 31. Page Orientation of  $0^\circ$ . The figure shows the relationship between the logical page and the data coordinate system.

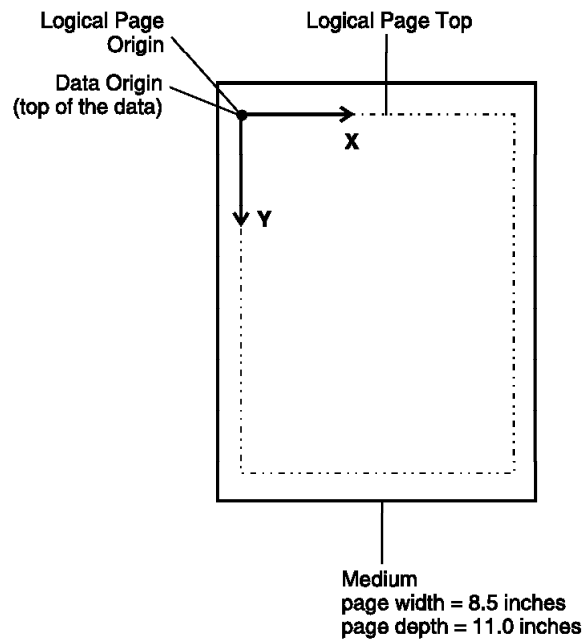
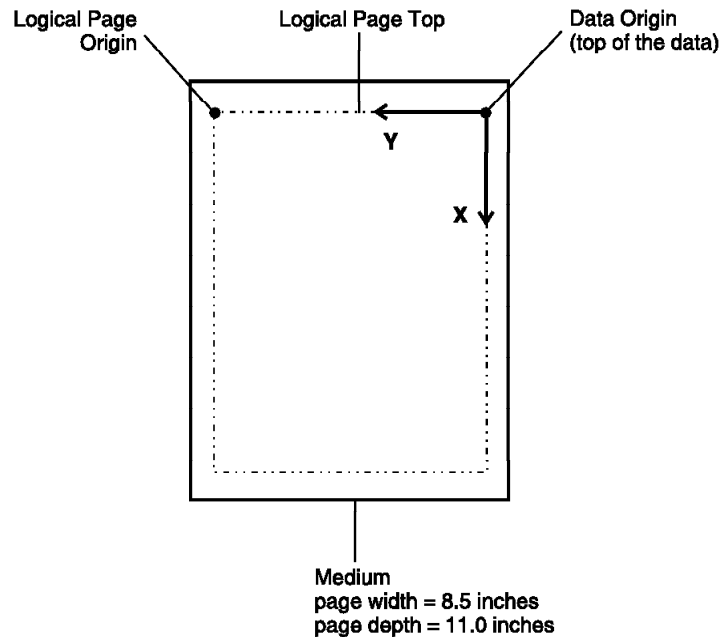


Figure 32. Page Orientation of 90°. The figure shows the relationship between the logical page and the data coordinate system.



The process of rotating pages does not rotate any page segments or overlays in the logical page; the process does, however, rotate any included data objects.

## Output Parameters

### Document Handle

The handle for the document.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0008** The state is invalid; a document is already started. Severity 8 (ERROR).
- 0017** The unit of measure is invalid. Severity 8 (ERROR).
- 0018** The page orientation is invalid. Severity 8 (ERROR).
- 0021** The state is invalid; the state must be start. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0125** The page width is invalid. Severity 8 (ERROR).

- 0126** The page depth is invalid. Severity 8 (ERROR).
- 0186** Null handle. Severity 8 (ERROR).
- 0217** The font specified cannot be used by AFP API. Severity 8 (ERROR).
- 0224** The output file exists, but Replace was not specified on the Set Output Characteristics procedure call. Severity 12 (SEVERE). (VM only)

### AFPBFLD (Begin Field)

#### Function

Begins a field of data in a row of a table.

#### Syntax

```
AFPBFLD(  
    HANDLE    AFPAPI-handle,  
    HANDLE    table-handle,  
    INTEGER4  field-id,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 33. Format of the AFPBFLD Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Table Handle

The handle of the table in which this row is being placed, returned from the AFPBTBL call.

##### Field ID

The ID of the field definition returned from the associated AFPDFLD (Define Field) call.

#### Output Parameters

##### Return Code

The return code for this call.

##### Severity Code

The severity of the return code.

#### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0111** The field has not been previously defined. Severity 8 (ERROR).
- 0156** The handle is invalid. Severity 8 (ERROR).



- 0157** The state is invalid; the state must be row. Severity 8 (ERROR).
- 0225** The field is not part of the Begin Row procedure call currently in effect. Severity 8 (ERROR).
- 0229** The field is rotated, and no subrow depth was specified in the Define Row procedure call. Severity 8 (ERROR).

### AFPBGRP (Begin Group)

#### Function

Begins a logical grouping of pages for viewing purposes using AFP Workbench for Windows and for archiving. Related pages can be indexed with the Attribute Name and Attribute Value parameters of the AFPPTAG procedure call. The group name aids in identifying a particular group if an Attribute Value is not unique within a document. A group name should be unique within a document. Nesting groups is not supported (that is, an AFPBGRP followed by another AFPBGRP). See “Indexing Data for Viewing and Archiving” on page 79 for more information about using this procedure call.

#### Syntax

```
AFPBGRP(  
    HANDLE    AFPAPI-handle,  
    HANDLE    document-handle,  
    CHAR(64)  group-name,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 34. Format of the AFPBGRP Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Document Handle

The handle for the document returned from the AFPBDOC call.

##### Group Name

The name for the group to be started, encoded using code page T1V10500; it should be unique within a document. The maximum number of characters in the group name is 64. Code page T1V10500 is IBM’s universal graphic character map. For more information about code page T1V10500 and other IBM code pages, refer to *IBM AFP Fonts: Technical Reference for Code Pages*.

#### Output Parameters

##### Return Code

The return code for this call.

##### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0226** A group is already active. Severity 8 (ERROR).
- 0269** The state is invalid; the state must be document. Severity 8 (ERROR).
- 0270** The group name is invalid. Severity 8 (ERROR).

### AFPBPAG (Begin Page)

#### Function

Begins a logical page and optionally overrides the page dimensions and orientation specified for the document. The initial current position is at the page origin that is at the top-left corner of the page.

#### Syntax

```
AFPBPAG(  
    HANDLE    AFPAPI-handle,  
    HANDLE    document-handle,  
    SREAL     page-width,  
    SREAL     page-depth,  
    INTEGER4  page-orientation,  
    HANDLE    page-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 35. Format of the AFPBPAG Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Document Handle

The handle for the document, returned from the AFPBDOC call.

##### Logical Page Width

The page width in the current unit of measure. If the default is specified, the page width from the AFPBDOC (Begin Document) procedure call is used. If data placed in the page exceeds the page width, the call that placed the data generates a non-zero return code with an ERROR severity code.

##### Logical Page Depth

The page depth in the current unit of measure. If the default is specified, the page depth from AFPBDOC (Begin Document) is used. If data placed in the page exceeds the page depth, the call that placed the data generates a non-zero return code with a WARNING severity code.

**Logical Page Orientation**

The orientation of the top of the logical page relative to the top of the media. If the default is specified, the page orientation from AFPBDOC (Begin Document) is used. The top of the logical page is the side associated with the page width. The top of the media depends on the printer that is used. Refer to *Advanced Function Presentation: Printer Information* for a description of the default media origin for AFP printers. For example:

- If a page orientation of 0° is specified, the top of the data coincides with the top of the media, as shown in Figure 36. You do not need to switch the width and depth measurements if the page is rotated 90° or 270°. The page rotation is absolute from the media origin and is not relative to the document orientation; that is, a 90°-rotated page is always relative to 0°, not 90° from the document orientation.
- If a page orientation of 90° is specified, the top of the data is rotated 90° in the clockwise direction with respect to the top of the media, but the top of the logical page remains the side associated with the width, as shown in Figure 37 on page 110.

AFPSPOS always sets the position with respect to the top of the data.

Figure 36. Page Orientation of 0°. The figure shows the relationship between the logical page and the data coordinate system.

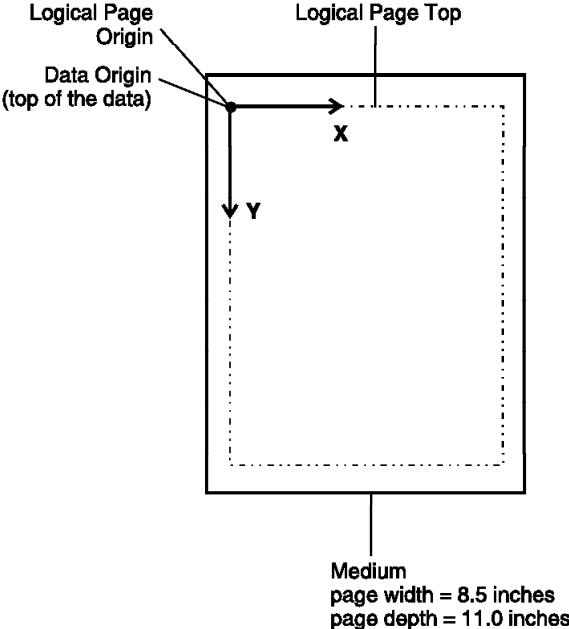
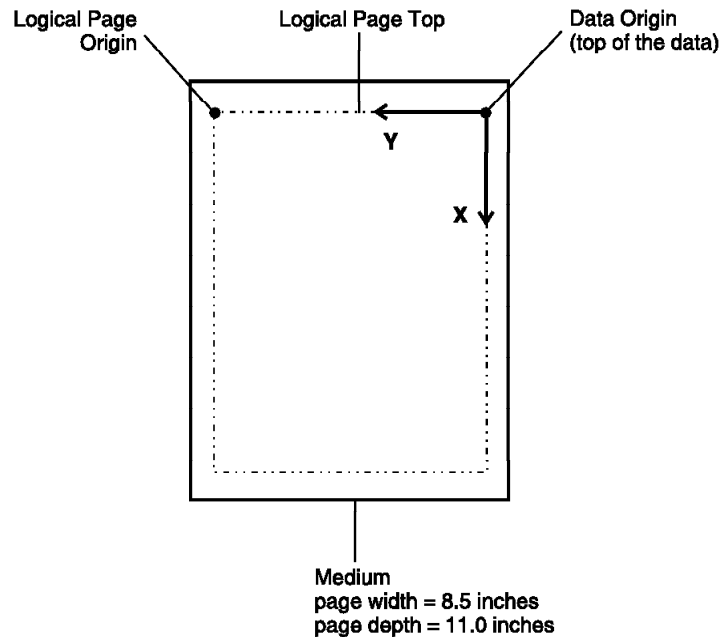


Figure 37. Page Orientation of 90°. The figure shows the relationship between the logical page and the data coordinate system.



The process of rotating pages does not rotate any page segments or overlays in the logical page; the process does, however, rotate any included data objects.

## Output Parameters

### Page Handle

The handle for the page.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

**0005** The handle is invalid. Severity 8 (ERROR).

**0009** The state is invalid; a page is already started. Severity 8 (ERROR).

**0018** The page orientation is invalid. Severity 8 (ERROR).

- 0022** The state is invalid; the state must be document. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0125** The page width is invalid. Severity 8 (ERROR).
- 0126** The page depth is invalid. Severity 8 (ERROR).
- 0186** Null handle. Severity 8 (ERROR).

### AFPBPARG (Begin Paragraph)

#### Function

Begins a paragraph at the current position using the current values for intercharacter spacing, word spacing, color, and font. You can format paragraphs four ways: ragged right, ragged left, centered, and justified. Hyphenation is not supported. See “AFPPTXT (Put Text)” on page 173 for information about putting data in a paragraph. The current position at the end of a paragraph is at the bottom-left corner of the paragraph.

#### Syntax

```
AFPBPARG(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    SREAL     first-line-indent,  
    INTEGER4  format-option,  
    SREAL     first-line-offset,  
    REAL      left-margin,  
    REAL      line-length,  
    SREAL     line-spacing,  
    BOOLEAN   paragraph-frame,  
    REAL      rt-rule-offset,  
    SREAL     bot-rule-offset,  
    INTEGER4  shading-pattern  
    INTEGER4  shading-intensity  
    HANDLE    paragraph-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 38. Format of the AFPBPARG Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Current Handle

The handle for the current state (page or area) that will contain this paragraph, returned from the AFPBPARG or AFPCARE calls.

##### First-Line Indent

The amount to indent text in the first line of the paragraph: block, indented, or hanging indent. A hanging indent is when the first line in the paragraph begins at the left margin and all subsequent lines are indented from the left margin by the specified amount. A value of 0 creates a block paragraph where the first line in the paragraph begins at the left margin. A positive value creates an indented paragraph where the first line in the paragraph is indented from the left margin by the specified amount and all subsequent lines are positioned at the left margin. A negative value creates a hanging indent paragraph.



### Format Option

The type of formatting to be performed for the paragraph:

- |                     |   |
|---------------------|---|
| <b>Ragged Right</b> | Text is aligned at the left margin of the paragraph; the right margin is not aligned.   |
| <b>Center</b>       | Text is centered between the left and right margins of the paragraph.   |
| <b>Ragged Left</b>  | Text is aligned at the right margin of the paragraph; the left margin is not aligned.   |
| <b>Justify</b>      | Text is aligned at both the left margin and right margin of the paragraph. The formatter varies the space between words to stretch or shrink a line to fit. |

### First Line Offset

The location of the character baseline for the first line of text in the paragraph as an offset from the paragraph origin in the current unit of measure. The value is in addition to the normal line spacing. If the default is specified, the offset is calculated based on the character height of the font being used.

For unframed paragraphs, the paragraph origin is at the current position. For framed paragraphs, the Y coordinate of the paragraph origin is at the bottom of the top rule of the paragraph frame (that is, adjusted by the rule thickness of the top rule). The X coordinate of the paragraph origin is unchanged.

### Left Margin

The left margin to be used for the lines in the paragraph as an offset from the paragraph origin.

### Line Length

The length of a line of formatted text in the current unit of measure.

### Line Spacing

The spacing between subsequent lines of text in the paragraph. If the default is specified, the line spacing associated with the largest font in each line of text is used.

### Paragraph Frame

The box or frame enclosing a paragraph. Specifies whether or not to enclose the paragraph in a box. Rules of the frame use the current rule thickness. See “AFPSRTH (Set Rule Thickness)” on page 195 for more information about rule thickness.

## Begin Paragraph

### Paragraph Frame Offset

If the paragraph is framed, the location of the right vertical rule and the bottom rule of the frame. The top horizontal rule begins at the paragraph origin and extends to the right vertical rule offset. The left vertical rule begins at the paragraph origin and extends to the bottom horizontal offset.

If the paragraph is not framed but is shaded, the area to be shaded.

### Right Vertical Rule Offset

The location of the right vertical rule as an offset from the paragraph origin if the paragraph is framed.

### Bottom Horizontal Rule Offset

The location of the bottom horizontal rule as an offset from the baseline of the last line in the paragraph if the paragraph is framed. The value is in addition to the normal line spacing. You can specify a specific value or use the default value. The default is determined by the line spacing of the largest font in the last line of text in the paragraph.

### Shading Pattern

Specifies whether to shade a paragraph and the shading pattern to use, either Standard or Screen. See Appendix C, “Shade Patterns and Types” for examples of shading patterns.

### Shading Intensity

A value between 0–100 that specifies the shading intensity to be used, with 1 being the least intense; 0 indicates no shading. See Appendix C, “Shade Patterns and Types” for examples of shading intensities.

## Output Parameters

### Paragraph Handle

The handle for the paragraph.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

**0005** The handle is invalid. Severity 8 (ERROR).

**0028** The Y position is invalid. Severity 8 (ERROR).

**0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).

**0072** A numeric overflow occurred; the specified left margin, when added to the current inline position, exceeded the maximum valid value. Severity 8 (ERROR).

**0129** The state is invalid; the state must be page or area. Severity 8 (ERROR).

- 0130** The shading pattern is invalid. Severity 8 (ERROR).
- 0131** The shading intensity parameter is invalid. Severity 8 (ERROR).
- 0133** The format option is invalid. Severity 8 (ERROR).
- 0144** The first line offset is invalid. Severity 8 (ERROR).
- 0145** The left margin parameter is invalid. Severity 8 (ERROR).
- 0146** The line length is invalid. Severity 8 (ERROR).
- 0147** The line spacing is invalid. Severity 8 (ERROR).
- 0148** The right vertical rule offset is invalid. Severity 8 (ERROR).
- 0149** The bottom rule offset is invalid. Severity 8 (ERROR).
- 0150** The state is invalid; a paragraph already exists. Severity 8 (ERROR).
- 0186** Null handle. Severity 8 (ERROR).
- 0191** The first line indent is invalid. Severity 8 (ERROR).
- 0202** The line length or right rule offset exceeds the width of the page or area. Severity 8 (ERROR).
- 0213** The maximum depth specified in the area, page, or table was exceeded; the object will not fit. Severity 4 (WARNING).
- 0279** A numeric overflow occurred; the specified line length, when added to the current inline position and the left margin, exceeded the maximum valid value. Severity 8 (ERROR).

### AFPBROW (Begin Row)

#### Function

Begins a new row in a table.

#### Syntax

```
AFPBROW(  
    HANDLE    AFPAPI-handle,  
    HANDLE    table-handle,  
    STRING    row-id,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 39. Format of the AFPBROW Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Table Handle

The handle for the associated table, returned from the AFPBTBL call.

##### Row ID

The ID of the row definition returned from the associated AFPDROW call.

#### Output Parameters

##### Return Code

The return code for this call.

##### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0110** The row has not been previously defined. Severity 8 (ERROR).
- 0154** The state is invalid; the state must be table. Severity 8 (ERROR).
- 0159** The state is invalid; the state must be table. Severity 8 (ERROR).

### AFPBTBL (Begin Table)

#### Function

Begins a table at the current position. Then you can format data into rows consisting of fields that can be framed and shaded. See “AFPBFLD (Begin Field)” on page 104, “AFPDFLD (Define Field)” on page 124, and “AFPDROW (Define Row)” on page 131 for more information. Also see “Tables” on page 48 for more information about creating tables. The current position at the end of a table is at the bottom-left corner of the table.

#### Syntax

```
AFPBTBL(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    REAL      table-width,  
    REAL      max-table-depth,  
    INTEGER4  table-rotation,  
    REAL      top-thickness,  
    REAL      bottom-thickness,  
    REAL      left-thickness,  
    REAL      right-thickness,  
    HANDLE    table-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 40. Format of the AFPBTBL Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Current Handle

The handle for the current page or area, returned from the AFPBPAG or AFPCARE calls.

##### Table Width

The width of the table in the current unit of measure. The width of the table should be the sum of the widths of the columns. If the sum of the column widths does not equal the table width, the column widths override the table width. If the table is framed, the left and right vertical rule thickness used for the table frame are included in the width of the table.

##### Maximum Table Depth

The maximum depth of the table in the current unit of measure. If the table is framed, the thickness of the top and bottom horizontal rule of the table frame is included in the depth of the table. If data placed in the table exceeds the table depth, the AFPEROW call that placed the data generates a non-zero return code with a WARNING severity code.

**Table Rotation**

The rotation of the table in a clockwise direction around the table origin. For 0° rotation, the top of the table is parallel to the top of the page or area containing the table.

**Top Horizontal Rule Thickness**

The thickness of the top horizontal rule of the table frame in the current unit of measure. A value of 0 specifies no rule.

**Bottom Horizontal Rule Thickness**

The thickness of the bottom horizontal rule of the table frame in the current unit of measure. A value of 0 specifies no rule.

**Left Vertical Rule Thickness**

The thickness of the left vertical rule of the table frame in the current unit of measure. A value of 0 specifies no rule; however, a value of 0 does not override a field rule.

**Right Vertical Rule Thickness**

The thickness of the right vertical rule of the table frame in the current unit of measure. A value of 0 specifies no rule; however, a value of 0 does not override a field rule.

**Output Parameters****Table Handle**

The handle for the table.

**Return Code**

The return code for this call.

**Severity Code**

The severity of the return code.

**Return Codes**

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0028** The Y position is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0106** The state is invalid; the state must be page or area. Severity 8 (ERROR).
- 0107** The table rotation is invalid. Severity 8 (ERROR).
- 0108** The right vertical thickness is invalid. Severity 8 (ERROR).
- 0136** The width is invalid. Severity 8 (ERROR).
- 0137** The depth is invalid. Severity 8 (ERROR).
- 0153** The state is invalid. Severity 8 (ERROR).
- 0170** The top horizontal rule thickness is invalid. Severity 8 (ERROR).

## Begin Table

- 0171** The bottom horizontal rule thickness is invalid. Severity 8 (ERROR).
- 0172** The left vertical rule thickness is invalid. Severity 8 (ERROR).
- 0186** Null handle. Severity 8 (ERROR).
- 0202** The table width is greater than the page or area width. Severity 8 (ERROR).



## AFPCARE (Create Area)

### Function

Creates an area in AFP API storage that you can fill with formatted elements for use on one or more pages in the document. To fill the area with elements, include the Area Handle returned from this call on subsequent calls. By doing this, you can set attributes or build and put elements such as character strings, boxes, rules, paragraphs, tables, resources, and objects. AFP API places elements relative to the area origin.

You can create and maintain multiple areas concurrently, using the individual area handles to indicate the area in which elements are placed and which area to place on a page.

You must issue AFPCARE and AFPEARE procedure calls either within the same page or in document state. You must end an area before placing it on a page. To place the area on a page, use the AFPPARE (Put Area) procedure call. The area and its contents remain in AFP API storage, until you delete them using the AFPXARE (Destroy Area) procedure call. As long as an area is in storage, you can place it multiple times. AFPEARE (End Area) ends the formatted area.

For unframed areas, the area origin is at the top-left corner of the area (similar to pages). For framed areas, the Y coordinate of the area origin is at the bottom of the top rule of the area frame (that is, adjusted by the rule thickness of the top rule). The X coordinate of the area origin is unchanged.

An area improves performance because it provides a mechanism of repeating boilerplate data without recreating it each time.

### Syntax

```
AFPCARE(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    REAL      area-width,
    REAL      maximum-area-depth,
    BOOLEAN   area-frame,
    INTEGER4  shading-pattern,
    INTEGER4  shading-intensity,
    HANDLE    area-handle,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 41. Format of the AFPCARE Procedure Call

### Input Parameters

**AFPAPI Handle**

The session handle returned from the AFPINIT call.

**Current Handle**

The handle for the current document or page returned from the AFPBDOC or AFPBPAG calls.

**Area Width**

The width of the area in the current unit of measure. If the area is framed, the thickness of the left and right vertical rule of the area frame is included in the width of the area. If data placed in the area exceeds the area width, the call that placed the data generates a non-zero return code with an ERROR severity code.

**Maximum Area Depth**

The maximum depth of the area in the current unit of measure. If the area is framed, the thickness of the top and bottom horizontal rule of the area frame is included in the depth of the area. If data placed in the area exceeds the area depth, the call that placed the data generates a non-zero return code with a WARNING severity code. AFP API returns the depth of the area when the AFPEARE (End Area) call is issued after all elements are placed in the area.

**Area Frame**

The box or frame enclosing the area. If the area is framed, the current rule thickness is used for the rules in the frame. The bottom horizontal rule thickness is included in the area depth that is returned on the AFPEARE (End Area) call.

**Note:** Vertical rules in a framed area are not included in the area depth, because the current position is unchanged when a vertical rule is drawn. This means that the bottom rule of the area may not enclose the vertical rule.

**Shading Pattern**

Specifies whether to shade an area and the shading pattern to use, either Standard or Screen. See Appendix C, "Shade Patterns and Types" for examples of shading patterns.

**Shading Intensity**

A value between 0–100 that specifies the shading intensity to be used, with 1 being the least intense; 0 indicates no shading. See Appendix C, "Shade Patterns and Types" for examples of shading intensities.

### Output Parameters

**Area Handle**

The handle to be used on subsequent calls that place data in the area or place the area on the page.

**Return Code**

The return code for this call.

**Severity Code**

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0023** The state is invalid; the state must be document or page. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0100** The shading pattern is invalid. Severity 8 (ERROR).
- 0101** The shading intensity is invalid. Severity 8 (ERROR).
- 0136** The width is invalid. Severity 8 (ERROR).
- 0137** The depth is invalid. Severity 8 (ERROR).

### AFPDFLD (Define Field)

#### Function

Creates a field definition for a table. Subsequent AFPBFLD (Begin Field) procedure calls use the field definition. You can use either AFPPCHS (Put Character String) or AFPPTXT (Put Text) procedure calls to put data into the field.

#### Syntax

```
AFPDFLD(  
    HANDLE    AFPAPI-handle,  
    HANDLE    document-handle,  
    INTEGER4  format-option,  
    REAL      alignment-position,  
    INTEGER4  vertical-format,  
    REAL      left-margin,  
    REAL      right-margin,  
    SREAL     line-spacing,  
    INTEGER4  text-orientation,  
    INTEGER4  shading-pattern,  
    INTEGER4  shading-intensity,  
    REAL      top-thickness,  
    REAL      bottom-thickness,  
    REAL      left-thickness,  
    REAL      right-thickness,  
    INTEGER4  field-id,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 42. Format of the AFPDFLD Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Document Handle

The handle for the current document returned from the AFPBDOC call.

**Format Option for AFPPTXT (Put Text)**

The type of formatting to be performed on formatted text placed with the AFPPTXT procedure call in the field as follows:

**Left**

Text is aligned at the left margin of the field, with a ragged right margin.

**Center**

Text is centered between the left and right margins of the field.

**Right**

Text is aligned at the right margin of the field.

**Justify**

Text is aligned at both the left margin and right margin of the field. The formatter varies the space between words to stretch or shrink a line to fit.

**Alignment Position for AFPPCHS (Put Character String)**

The position within the field for character alignment. The position is specified as an offset from the left margin of the field. This parameter has no effect on formatted data; that is, data included on the AFPPTXT (Put Text) call and is used only when placing data with the AFPPCHS (Put Character String) call with Character Alignment Option.

**Vertical Field Format Option**

The vertical alignment option for lines of text within the field. You can align text at the top, center, or bottom.

**Left Margin**

The left margin for lines of text in the field as an offset from the leftmost edge of the left vertical rule in current units.

**Right Margin**

The right margin for lines of text in the field as an offset from the leftmost edge of the right vertical rule in current units.

**Line Spacing**

The spacing between subsequent lines of text in the current unit of measure. You can specify a specific value or use the default baseline increment of the font. If the default is specified, the line spacing associated with the largest font in each line of text is used.

This parameter has no effect on unformatted data; that is, on data included with the AFPPCHS (Put Character String call).

**Field Text Orientation**

The degree of orientation for lines of text in the field.

**Shading Pattern**

Specifies whether to shade a paragraph and the shading pattern to use, either Standard or Screen. See Appendix C, "Shade Patterns and Types" for examples of shading patterns.

**Shading Intensity**

A value between 0–100 that specifies the shading intensity to be used, with 1 being the least intense; 0 indicates no shading. See Appendix C, "Shade Patterns and Types" for examples of shading intensities.

## Define Field

### Field Frame

The vertical rules to be used for the field. The field width and row depth specifications are used to determine the size of the area to be framed. If a field shares a common boundary with another field, the thicker of the two rules is used. If a field shares a common boundary with a table or row, the table or row rule is used.

### Top Horizontal Rule Thickness

The thickness of the top horizontal rule. A value of 0 specifies no rule.

### Bottom Horizontal Rule Thickness

The thickness of the bottom horizontal rule. A value of 0 specifies no rule.

### Left Vertical Rule Thickness

The thickness of the left vertical rule. A value of 0 specifies no rule.

### Right Vertical Rule Thickness

The thickness of the right vertical rule. A value of 0 specifies no rule.

## Output Parameters

### Field ID

The field ID for use on associated Begin Field calls.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0087** The state is invalid; the state must be document. Severity 8 (ERROR).
- 0094** The format option is invalid. Severity 8 (ERROR).
- 0095** The vertical field format is invalid. Severity 8 (ERROR).
- 0096** The shading pattern is invalid. Severity 8 (ERROR).
- 0097** The shading intensity is invalid. Severity 8 (ERROR).
- 0098** The field text orientation is invalid. Severity 8 (ERROR).
- 0099** The top horizontal rule thickness is invalid. Severity 8 (ERROR).
- 0112** The alignment position is invalid. Severity 8 (ERROR).
- 0118** The left margin is invalid. Severity 8 (ERROR).
- 0119** The line spacing is invalid. Severity 8 (ERROR).

- 0120** The right margin is invalid. Severity 8 (ERROR).
- 0173** The bottom horizontal rule thickness is invalid. Severity 8 (ERROR).
- 0174** The left vertical rule thickness is invalid. Severity 8 (ERROR).
- 0175** The right vertical rule thickness is invalid. Severity 8 (ERROR).
- 0176** The column width is invalid. Severity 8 (ERROR).

## AFPDFNT (Define Font by Attributes)

### Function

Creates a font ID on your system that matches a specified attribute. The font ID is used on subsequent AFPSFNT (Set Font) calls. The font must exist in a font library available to AFP API. See “AFPSLIB (Set Resource Library Names)” on page 187 for more information about libraries.

### Syntax

```
AFPDFNT (
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    TOKEN     code-page,
    INTEGER4  desc-name-length,
    STRING    descriptive-name,
    INTEGER4  point-size,
    INTEGER4  weight,
    INTEGER4  width,
    INTEGER4  rotation,
    INTEGER4  style,
    STRING    font-id,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 43. Format of the AFPDFNT Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the document, page, or area that the font is defined for, returned from the AFPBDOC, AFPBPAG, or AFPCARE calls.

#### Code Page

The code page name.

- For MVS, this is the member name.
- For VM, this is the file name.
- For VSE, this is the member name in the font phase.

A code page is a particular assignment of hexadecimal identifiers to graphic characters. For example, X'4F' is an exclamation point (!) in code page T1V10500, and X'5A' is an exclamation point (!) in code page T1V10037. If the code page used by your terminal or system for input is different from the code page used to present the data, you may not get the output you expect.

**Note:** ASCII code pages are *not* supported.



**Font Attributes:**

The parameters that follow describe the font attributes. See “Selecting the Font You Want” on page 68 for information about determining the attributes for fonts on your system.

**Descriptive Name Length**

Length of the descriptive name, including spaces. For example, “TIMES NEW ROMAN LATIN1” has a length of 22. The descriptive name can have a maximum length of 32.

**Descriptive Name**

Name of the font, for example, “TIMES NEW ROMAN LATIN1.” The descriptive name must match the name in the FLIP listing *exactly*. The descriptive name is case sensitive; all descriptive names are currently defined in uppercase. See “Font Library Indexing Program” on page 69 for information about the FLIP listing.

**Point Size**

Size of the font, for example, 12 points.

**Note:** Fractional (that is, non-integer) point sizes are *not* supported.

**Weight**

The thickness of the font:

UltraLight  
ExtraLight  
Light  
SemiLight  
Medium  
SemiBold  
Bold  
ExtraBold  
UltraBold

**Width**

The width of the font:

UltraCondensed  
ExtraCondensed  
Condensed  
SemiCondensed  
Normal  
SemiExpanded  
Expanded  
ExtraExpanded  
UltraExpanded

**Rotation**

Character rotation clockwise from the baseline as follows:

0°  
90°  
180°  
270°

This means that the font is built with this rotation, not that AFP API rotates it for you.

**Note:** Character rotation is not the same as text rotation. In most cases, you use a character rotation of 0° for all text rotations. A character rotation of 270° can produce columnar text; character rotations of 90° or 180° usually produce unreadable character strings.

## Define Font

### Style

Roman or italic.

## Output Parameters

### Font ID

The font ID for use on the AFPSFNT (Set Font) procedure call.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

Some font errors are detected by the Set Font (AFPSFNT) procedure. See page 184 for a list of these errors.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0074** The code page is invalid. Severity 8 (ERROR).
- 0075** The descriptive name is invalid. Severity 8 (ERROR).
- 0076** The point size is invalid. Severity 8 (ERROR).
- 0077** The weight is invalid. Severity 8 (ERROR).
- 0078** The width is invalid. Severity 8 (ERROR).
- 0079** The rotation is invalid. Severity 8 (ERROR).
- 0080** The style is invalid. Severity 8 (ERROR).
- 0088** The state is invalid; the state must be document, page, or area. Severity 8 (ERROR).
- 0143** The descriptive name length is invalid. Severity 8 (ERROR).
- 0151** A null font ID parameter was specified. Severity 8 (ERROR).
- 0177** The maximum number of font definitions has been exceeded. Severity 12 (SEVERE).

## AFPDROW (Define Row)

### Function

Creates a row definition for use in a table. Associated AFPBROW (Begin Row) calls use this row definition. The AFPBFLD (Begin Field) procedure call places fields in a row.

### Syntax

```

AFPDROW(
    HANDLE      AFPAPI-handle,
    HANDLE      document-handle,
    SGLARRAY    min-subrow-depth-array,
    REAL        top-thickness,
    REAL        bottom-thickness,
    INTEGER4    number-columns,
    INTEGER4    number-subrows,
    MULTARRAY   row-arrange-array(number-subrows,number-columns),
    SGLARRAY    column-width-array(number-columns),
    STRING      row-id,
    INTEGER4    ret-code,
    INTEGER4    severity-code
)

```

Figure 44. Format of the AFPDROW Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Document Handle

The handle for the current document returned from the AFPBDOC call.

#### Minimum Subrow Depth Array

The minimum depth of the subrows in a row in the current unit of measure. You can specify a value or use a default that is determined by the field that uses the largest font (that is, the font with the largest default line space).

**Note:** When using a text orientation of 90° or 270° for a field in a subrow, the minimum subrow depth must be other than 0.

The depth of a subrow includes the top horizontal rule thickness used for the subrow. The depth of the last subrow in a table also includes the bottom horizontal rule thickness. If a subrow shares a common boundary with a table, the table rule is used.

#### Top Thickness, Horizontal Rule

The thickness of the top horizontal rule. If you don't want a rule, specify a value of 0. If a row shares a common boundary with a table, the table rule is used. The rule used for the boundary between two rows is the thicker of the two rules.

## Define Row

### Bottom Thickness, Horizontal Rule

The thickness of the row's horizontal rule. If you don't want a rule, specify a value of 0. If a row shares a common boundary with a table, the table rule is used. The rule used for the boundary between two rows is the thicker of the two rules.

### Number of Columns and Subrows

The subrows and columns form an array that is used to describe the contents of the row. The number of elements in the array, computed by multiplying the number of subrows by the number of column, cannot exceed 64.

### Number of Columns

The number of vertical divisions within a row.

### Number of Subrows

The number of horizontal divisions within a row.

### Row Arrange Array

The field IDs to be used in the row and the arrangement of fields within the row. The field IDs are from the AFPDFLD (Define Field) call.

### Column Widths

The widths of the columns in the row in the current unit of measure. The width of a column includes the left vertical rule thickness used for the field. The right-most column in a table also includes the right vertical rule thickness in the column width. If a field shares a common boundary with a table, the rule thickness of the table frame is used. The sum of individual column widths should equal the table width specified on the AFPBTBL (Begin Table) call. If data placed in a column with AFPPCHS exceeds the column width, the call that placed the data generates a non-zero return code with an ERROR severity code.

## Output Parameters

### Row ID

The row ID for use on associated Begin Row calls.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

**0005** The handle is invalid. Severity 8 (ERROR).

**0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).

**0084** The bottom horizontal rule thickness is invalid. Severity 8 (ERROR).

**0089** The state is invalid; the state must be document. Severity 8 (ERROR).

- 0102** The depth is invalid. Severity 12 (SEVERE).
- 0103** The top horizontal rule thickness is invalid. Severity 8 (ERROR).
- 0111** The field has not been previously defined. Severity 8 (ERROR).
- 0113** The number of columns is invalid. Severity 8 (ERROR).
- 0152** A null row ID parameter was specified. Severity 8 (ERROR).

## AFPEARE (End Area)

### Function

Ends an area that has been created using the AFPCARE (Create Area) procedure call. You cannot place any additional elements in the area after issuing this call. All elements in the area are formatted and placed in AFP API storage for placement with the AFPPARE (Put Area) call. The area remains in storage until you delete it with the AFPXARE (Destroy Area) call.

### Syntax

```
AFPEARE(  
    HANDLE    AFPAPI-handle,  
    HANDLE    area-handle,  
    REAL      area-depth,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 45. Format of the AFPEARE Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Area Handle

The handle of the area ended, returned from the AFPCARE call.

### Output Parameters

#### Area Depth

The depth of the area, after it is formatted with all elements that were placed in it, in the current unit of measure.

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0012** A null area depth parameter is specified. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).

## AFPEDOC (End Document)

### Function

Ends the document. If the AFPGBUF procedure call has been issued, also returns the last output record and record length in the parameters specified on the last AFPGBUF call.

### Syntax

```
AFPEDOC(  
    HANDLE    AFPAPI-handle,  
    HANDLE    document-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 46. Format of the AFPEDOC Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Document Handle

The handle of the document ended, returned from the AFPBDOC call. When returned, this parameter is set to 0.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0179** The state is invalid. Severity 8 (ERROR).
- 0210** An error occurred writing to the output file specified in the Set Output Characteristics procedure call. Severity 12 (SEVERE).



---

## AFPEFLD (End Field)

### Function

Ends the field in a row of a table.

### Syntax

```
AFPEFLD(
    HANDLE    AFPAPI-handle,
    HANDLE    table-handle,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 47. Format of the AFPEFLD Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Table Handle

The handle of the table in which this field is used, returned from the AFPBTBL call.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0158** The state is invalid; the state must be field. Severity 8 (ERROR).
- 0163** The handle is invalid. Severity 8 (ERROR).

### AFPEGRP (End Group)

#### Function

Ends a logical grouping of pages for archiving and viewing purposes. See “Indexing Data for Viewing and Archiving” on page 79 for more information about using this procedure call.

#### Syntax

```
AFPEGRP(  
    HANDLE    AFPAPI-handle,  
    HANDLE    document-handle,  
    CHAR(64)  group-name,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 48. Format of the AFPEGRP Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Document Handle

The handle for the document returned from the AFPBDOC call.

##### Group Name

The name of the group to be ended, encoded using code page T1V10500. A previous AFPBGRP must have been issued with the name. The maximum number of characters in the group name is 64. Code page T1V10500 is IBM's universal graphic character map. For more information about code page T1V10500 and other IBM code pages, refer to *IBM AFP Fonts: Technical Reference for Code Pages*.

#### Output Parameters

##### Return Code

The return code for this call.

##### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0227** The specified group is not active. Severity 8 (ERROR).
- 0270** The group name is invalid. Severity 8 (ERROR).
- 0271** The state is invalid; the state must be document. Severity 8 (ERROR).

## AFPEND (End AFP API)

### Function

Ends the AFP API session and frees all AFP API storage. For the AFP API Handle parameter, use the AFP API handle returned by the Initialize AFP API procedure call.

The only AFP API procedure call you can issue after AFPEND is AFPINIT.

### Syntax

```
AFPEND(  
    HANDLE    AFPAPI-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 49. Format of the AFPEND Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

**0004** The state is invalid; the state must be start. Severity 8 (ERROR).

**0005** The handle is invalid. Severity 8 (ERROR).

**0011** The state is invalid; the state must be start. Severity 8 (ERROR).

**0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).

---

## AFPEPAG (End Page)

### Function

Ends the page and causes AFP API to write the page to the designated output file. See “AFPSOUT (Set Output Characteristics)” on page 190 for a description of the output file that’s created.

### Syntax

```
AFPEPAG(  
    HANDLE    AFPAPI-handle,  
    HANDLE    page-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 50. Format of the AFPEPAG Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Page Handle

The handle of the page terminated returned from the AFPBPAG call. When returned, this parameter is set to 0.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0178** The state is invalid. Severity 8 (ERROR).
- 0210** An error occurred writing to the output file specified in the Set Output Characteristics procedure call. Severity 12 (SEVERE).
- 0219** A page segment or included object on the page extends beyond the dimensions of the logical page. Severity 8 (ERROR).
- 0220** The page may not be printable because of the number or size of the fonts selected. Severity 8 (ERROR).

---

## AFPEPAR (End Paragraph)

### Function

Ends the paragraph and returns the depth of the paragraph, including the depth of the bottom rule if the paragraph is framed, in the current unit of measure. The current position after the call to AFPEPAR is at the bottom-left corner of the paragraph.

### Syntax

```
AFPEPAR(  
    HANDLE    AFPAPI-handle,  
    HANDLE    paragraph-handle,  
    REAL      paragraph-depth,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 51. Format of the AFPEPAR Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Paragraph Handle

The handle of the paragraph ended, returned from the AFPBPAR call. When returned, this parameter is set to 0.

### Output Parameters

#### Paragraph Depth

The depth of the paragraph in the current unit of measure.

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

**End Paragraph**

## **Return Codes**

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0165** The handle is invalid. Severity 8 (ERROR).
- 0166** A null paragraph depth parameter is specified. Severity 8 (ERROR).



---

## AFPEROW (End Row)

### Function

Ends a row in a table. The current position is at the bottom-left corner of the row.

### Syntax

```
AFPEROW (  
    HANDLE    AFPAPI-handle,  
    HANDLE    table-handle,  
    REAL      current-table-depth,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 52. Format of the AFPEROW Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Table Handle

The handle of the table in which this row is used, returned from the AFPBTBL call.

### Output Parameters

#### Current Table Depth

The depth of the table in the current unit of measure.

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

**End Row**

## **Return Codes**

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0160** The state is invalid; the state must be row. Severity 8 (ERROR).
- 0164** The handle is invalid. Severity 8 (ERROR).
- 0194** A null current table depth parameter is specified. Severity 8 (ERROR).
- 0209** A table row contains more data than will fit in the remaining or maximum table depth or on the page. Severity 4 (WARNING).

---

## AFPETBL (End Table)

### Function

Ends a table and returns the depth of the table, including the depth of the horizontal rule, in the current units of measure. The current position at the end of a table is at the bottom-left corner of the table.

### Syntax

```
AFPETBL(  
    HANDLE    AFPAPI-handle,  
    HANDLE    table-handle,  
    REAL      table-depth,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 53. Format of the AFPETBL Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Table Handle

The handle of the table terminated, returned from the AFPBTBL call. When returned, this parameter is set to 0.

### Output Parameters

#### Table Depth

This parameter contains the depth of the table in the current unit of measure.

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0116** A null depth parameter is specified. Severity 8 (ERROR).
- 0155** The state is invalid; the state must be page or area. Severity 8 (ERROR).

## AFPGBUF (Get Output Buffer)

### Function

Returns an AFP API output record and its length to a buffer in your program. Each record contains one structured field.

Before calling AFPGBUF, end the page with an AFPEPAG call. Then issue the AFPGBUF call repeatedly, until the More Records parameter indicates that no more records exist for the page. Repeat the AFPGBUF calls for each page of the document. When you issue the AFPEDOC call to end the document, AFP API returns the final record and record length into the same areas used in the last AFPGBUF call.

**Note:** Before using AFPGBUF, request that AFP API write output records to an output buffer instead of to an output file on the AFPSOUT call.

### Syntax

```
AFPGBUF(
    HANDLE    AFPAPI-handle,
    HANDLE    document-handle,
    STRING    buffer,
    INTEGER4  buffer-length
    BOOLEAN   more-records
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 54. Format of the AFPGBUF Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Document Handle

The handle for the current document returned from the AFPBDOC call.

### Output Parameters

#### Buffer

The area where AFPGBUF returns the next output record (structured field) in the page. The buffer should be large enough to hold the largest output record possible, as specified in the output record size parameter on the AFPSOUT procedure call.

### Buffer Length

The length of the output record returned in the buffer parameter. If the length of the output record is longer than the output record size specified on the AFPSOUT call, only the number of bytes specified in the output record size parameter is returned. In this case, the return code indicates that the entire record was not returned, and the buffer length contains the actual length of the record so that you know how large the output record size parameter needs to be.

### More Records

Indicates whether or not another record exists for the page. If this parameter is set to TRU, call AFPGBUF again to obtain the next record.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0280** The state is invalid; the state must be document. Severity 8 (ERROR).
- 0281** Buffered output was not requested on the AFPSOUT procedure call. Severity 12 (SEVERE).
- 0282** The record (structured field) is larger than the size of the buffer provided for the record. AFP API truncated the record to fit into the buffer. Severity 8 (ERROR).

## AFPINIT (Initialize AFP API)

### Function

Initializes AFP API. You cannot issue any other AFP API procedure call until you have successfully initialized AFP API; otherwise, the system ends abnormally.

### Syntax

```
AFPINIT(
    HANDLE    AFPAPI-handle,
    BOOLEAN   trace,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 55. Format of the AFPINIT Procedure Call

### Input Parameters

#### Trace

The trace facility is no longer supported; however, you must still specify this parameter. AFP API ignores this parameter.

### Output Parameters

#### AFPAPI Handle

The identifier for this AFP API session. This handle is an input on all subsequent related AFP API calls.

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0212** The font index could not found or could not be read. Severity 12 (SEVERE).
- 0223** During initialization, the AFP API module could not be loaded into the system. Severity 12 (SEVERE).

### AFPINVM (Invoke Medium Map)

#### Function

Selects a medium map from the form definition resource used for printing the AFP API output. A medium map, also called a copy group, is a set of print options that includes names of medium overlays to be printed, which input bin to use, the number of copies, and whether to duplex the output.

This call forces printing to begin on a new sheet of paper. All pages following this call print with the medium map named, until another AFPINVM call is issued.

Use a form definition resource that contains the medium map named in this call for printing or viewing your AFP API output.

#### Syntax

```
AFPINVM(  
    HANDLE    AFPAPI-handle,  
    HANDLE    document-handle,  
    TOKEN     medium-map name,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 56. Format of the AFPINVM Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Document Handle

The handle for the document, returned from the AFPBDOC call.

##### Medium Map Name

The name of the medium map to be used for printing subsequent pages in the document.

##### Notes:

1. AFP API does not verify the existence of the medium map in the form definition and returns no errors if the medium map doesn't exist. However, PSF errors may result.
2. If the medium-map you name is the same as the medium-map named on the previous Invoke Medium Map call, AFP API ignores the call and returns no errors.



## Output Parameters

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0192** The state is invalid. Severity 8 (ERROR).

### AFPIOBJ (Include Object)

#### Function

Includes an image or graphics object inline at the current position and specifies the size, rotation, mapping option, and offset of the printed object.

The object must be formatted as Image Object Content Architecture (IOCA) or Graphics Object Content Architecture (GOCA) in S/370 print record format,<sup>7</sup> such as that produced by Graphical Data Display Manager (GDDM).

The object can be part of a page segment or document or can be included as an individual object. The object must reside in the object library identified by the AFPSLIB procedure call or in the AFP API default object library. Use AFPIPSG to include page segments that contain an IM1 image. If the included member contains more than one object, only the first object inside the member is included in the document. The object is placed at the current position. The following apply:

- The current environment, for example, current font, color, rule thickness, and so on, has no affect on the included object.
- The included object has no affect on the current environment, for example, current position, font, color, rule thickness, and so on.
- At the end of this function, the current position is unchanged.
- If included in an area, the object rotates with the area.

#### Notes:

1. The AFPIOBJ call is *not* supported in a CICS/ESA environment; if you issue the AFPIOBJ call, your program receives a return code with a SEVERE severity code. You can, however, include an object that is in a page segment using the AFPIPSG call.
2. Refer to *Advanced Function Presentation: Printer Information* to determine whether your printer can process IOCA image data and GOCA graphics data.

---

<sup>7</sup> Each structured field must be in a separate print record with the value X'5A' in the first byte of each record.

## Syntax

```

AFPIOBJ(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    TOKEN     object-name,
    SREAL     object-width,
    SREAL     object-depth,
    INTEGER4  object-rotation,
    INTEGER4  object-mapping-option,
    SREAL     object-x-offset,
    SREAL     object-y-offset,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)

```

Figure 57. Format of the AFPIOJB Procedure Call

## Input Parameters

### AFPAPI Handle

The session handle returned from the AFPINIT call.

### Current Handle

The handle for the current page or area, returned from the AFPBPAG or AFPCARE calls.

### Object Name

The name of the object in the object library. See “AFPSLIB (Set Resource Library Names)” on page 187 for information about the resource library.

- For MVS, this is the member name.
- For VM, this is the file name. The default file type is OBJT3820.
- For VSE, this is the member name in the object phase.

### Object Area Width

The width of the object area in which to map the object data in the current unit of measure. You can specify a value or use the width in the object.

### Object Area Depth

The depth of the object area in which to map the object data in the current unit of measure. You can specify a specific value or use the default depth in the object.

### Object Area Rotation

The rotation of the object area in a clockwise direction around the object’s origin. Valid values are 0°, 90°, 180°, 270°, or you can specify an indicator to use the default rotation in the object.

### Object Mapping Options

The mapping of the object data within its area. The possible values depend upon the type of object (or you can use the object's default).

- For IOCA objects, the valid values are:
  - Scale to fit
  - Center and Trim
  - Position and Trim
  - Point to pel
  - Point to pel with double dot
- For GOCA objects, the valid values are:
  - Scale to fit
  - Center and Trim
  - Position and Trim

**Note:** If the POINT-TO-PEL or the DOUBLE-DOT option is specified for a GOCA object, the mapping option specified in the object is used. If the object does not contain a mapping option, the default value of SCALE-TO-FIT is used. In this case, AFP API does not return an error.

These terms mean the following:

#### **SCALE-TO-FIT**

Center the object within the area dimensions specified in Object Area Width and in Object Area Depth, and scale the object to fit within the area.

#### **CENTER-AND-TRIM**

Center the object within the area dimensions specified in Object Area Width and in Object Area Depth, and trim what falls outside the area.

#### **POSITION-AND-TRIM**

Position the object at the location specified in Object Position within the dimensions specified in Object Area Width and in Object Area Depth, and trim what falls outside the area.

#### **POINT-TO-PEL**

Position the object at the object area origin within the dimensions specified in Object Area Width and in Object Area Depth, and trim what falls outside the area. No resolution correction is done; that is, each image point is mapped to a pel.

#### **DOUBLE-DOT**

Position the object at the object area origin within the dimensions specified in Object Area Width and in Object Area Depth, and trim what falls outside the area. Each image point is doubled. No resolution correction is done; that is, each of the new image points is mapped to a pel.

**Object Position**

The position of the object data relative to the object area origin. It is valid only for an object mapping option of position and trim. For all other mapping options, this parameter is ignored. You can specify a specific value or use the default position in the object.

**Object X Offset**

The X offset from the object area origin.

**Object Y Offset**

The Y offset from the object area origin.

**Output Parameters****Return Code**

The return code for this call.

**Severity Code**

The severity of the return code.

**Return Codes**

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0201** The formatter cannot read the object library. Severity 12 (SEVERE).
- 0213** The maximum depth specified in the area, page, or table was exceeded; the object will not fit. Severity 4 (WARNING).
- 0216** The requested resource cannot be found. Severity 8 (ERROR).
- 0222** The object contains invalid structured fields. Severity 8 (ERROR).
- 0260** The state is invalid. Severity 8 (ERROR).
- 0262** The object area width is invalid.
- 0263** The object area depth is invalid. Severity 8 (ERROR).
- 0264** The object area rotation is invalid. Severity 8 (ERROR).
- 0265** The object area mapping option is invalid. Severity 8 (ERROR).
- 0266** The object X offset is invalid. Severity 8 (ERROR).
- 0267** The object Y offset is invalid. Severity 8 (ERROR).
- 0269** The state is invalid; the state must be document. Severity 8 (ERROR).

### AFPIOVL (Include Page Overlay)

#### Function

Creates a reference to an overlay at the current position. The following apply:

- The current environment, for example, current font and color, has no affect on the included overlay.
- The included overlay has no affect on the current environment.
- The current position is unchanged after the AFPIOVL procedure call.
- The physical top of the overlay is parallel to the top of the medium, even if the top of the page is rotated. See “ AFPBDOC (Begin Document)” on page 100 for a description of orientation.

#### Syntax

```
AFPIOVL(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    TOKEN     ovly-name,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 58. Format of the AFPIOVL Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Current Handle

The handle for the current page or area, returned from the AFPBPAG (Begin Page) or AFPCARE (Create Area) calls.

##### Overlay Name

The name of the page overlay in the overlay library used by PSF.

- For MVS, this is the member name.
- For VM, this is the file name.
- For VSE, this is the member name in the overlay phase.

The overlay must be available to the printer at the time of printing. You can include up to 127 unique page overlays on a page.

**Note:** AFP API does not verify the existence of the overlay in the library and returns no errors if the overlay doesn't exist.

## Output Parameters

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0070** The state is invalid; the state must be page or area. Severity 8 (ERROR).

### AFPIPSG (Include Page Segment)

#### Function

Creates a reference to a page segment or brings the page segment inline at the current position. The current position is unchanged after the AFPIPSG procedure call.

**Note:** If the page segment contains text, the current environment may affect the page segment, and the page segment may affect the current environment.

#### Syntax

```
AFPIPSG(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    TOKEN     pseg-name,  
    BOOLEAN   inline-option,  
    BOOLEAN   reuse-option,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 59. Format of the AFPIPSG Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Current Handle

The handle for the current page or area, returned from the AFPBPAG or AFPCARE calls.

##### Page Segment Name

The name of the page segment in the page segment library.

- For MVS, this is the member name.
- For VM, this is the file name.
- For VSE, this is the member name in the page segment phase.

The page segment must be available to AFP API in the page segment library specified in the AFPSLIB (Set Resource Library Names) procedure call. If the page segment is not brought inline, it must be available to the printer at print time.

##### Inline Option

Indicates whether the page segment should be brought inline as part of the page or simply referenced.

##### Reuse Option

Indicates whether a page segment that is to be referenced will be reused on multiple pages within the document. This parameter is ignored if the page segment is brought inline.



## Output Parameters

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0028** The Y position is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0180** The state is invalid. Severity 8 (ERROR).
- 0185** Invalid values are specified for the inline or reuse parameters. Severity 8 (ERROR).
- 0201** The formatter cannot read the page segment library. Severity 12 (SEVERE).
- 0213** The maximum depth specified in the area, page, or table was exceeded; the object will not fit. Severity 4 (WARNING).
- 0216** The requested resource cannot be found. Severity 8 (ERROR).
- 0222** The page segment contains invalid structured fields. Severity 8 (ERROR).

### AFPPARE (Put Area)

#### Function

Places an area at the current position. You cannot place an area until after you create it with the AFPCARE (Create Area) call and end it with AFPEARE (End Area) call. The current position is unchanged by an AFPPARE procedure call. See “AFPSPOS (Set Position)” on page 193 for ways to place an area.

#### Syntax

```
AFPPARE(  
    HANDLE    AFPAPI-handle,  
    HANDLE    page-handle,  
    HANDLE    area-handle,  
    INTEGER4  area-rotation,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 60. Format of the AFPPARE Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Page Handle

The handle of the current page, returned from the AFPBPAG call.

##### Area Handle

The handle of the area to be placed on the page, returned from the AFPCARE call.

##### Area Rotation

The degree of rotation of the area in the clockwise direction around the area origin. Rotating an area rotates data objects but does not rotate any page segments or overlays included in the area.

#### Output Parameters

##### Return Code

The return code for this call.

##### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0007** The area has not been ended. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0053** The state is invalid; the state must be page. Severity 8 (ERROR).
- 0054** The rotation is invalid. Severity 8 (ERROR).
- 0071** The area handle does not exist. Severity 8 (ERROR).
- 0221** The contents of the area extend beyond the dimensions of the logical page. Severity 8 (ERROR).

## AFPPBOX (Put Box)

### Function

Draws a box with the top-left corner beginning at the current position for the specified width and depth, using the current color value and rule thickness for the rules of the box. The top of the box is parallel to the top of the area or page containing the box. The current position is unchanged at the end of this call.

### Syntax

```
AFPPBOX(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    REAL      box-width,  
    REAL      box-depth,  
    INTEGER4  shading-pattern,  
    INTEGER4  shading-intensity,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 61. Format of the AFPPBOX Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current page or area returned from the AFPBPAG or AFPCARE calls.

#### Box Width

The width of a box in the current unit of measure. The box width is measured from the left side of the left rule to the left side of the right vertical rule.

#### Box Depth

The depth of the box in the current unit of measure. The box depth is measured from the top side of the top horizontal rule to the bottom side of the bottom horizontal rule.

**Shading Pattern and Intensity**

The shading pattern and intensity for the box.

**Shading Pattern**

Specifies whether to shade a paragraph and the shading pattern to use, either Standard or Screen. See Appendix C, “Shade Patterns and Types” for examples of shading patterns.

**Shading Intensity**

A value between 0–100 identifying different intensities, with 1 being the least intense; 0 indicates no shading. See Appendix C, “Shade Patterns and Types” for examples of shading intensities.

**Output Parameters****Return Code**

The return code for this call.

**Severity Code**

The severity of the return code.

**Return Codes**

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0028** The Y position is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0122** The shading pattern is invalid. Severity 8 (ERROR).
- 0123** The shading intensity is invalid. Severity 8 (ERROR).
- 0124** The state is invalid; the state must be page or area. Severity 8 (ERROR).
- 0167** The box width is invalid. Severity 8 (ERROR).
- 0168** The box depth is invalid. Severity 8 (ERROR).
- 0213** The maximum depth specified in the area, page, or table was exceeded; the object will not fit. Severity 4 (WARNING).

### AFPPCHS (Put Character String)

#### Function

Places a character string on the page using the current values for intercharacter spacing, word spacing, color, and font. Characters are placed in the direction of the X axis to form a line of text.

At the end of this call, the current position is either unchanged or is adjusted in the X (inline) direction as described in the Position Option parameter.

**Note:** If you place the character string at a valid position with the AFPSPPOS call, but the top of the character string extends beyond the boundaries of a page or area, the results may not be what you expect, and the text may not print. AFP API does not issue an error return code for this situation.

#### Syntax

```
AFPPCHS(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    INTEGER4  string-length,  
    STRING    character-string,  
    INTEGER4  alignment-option,  
    CHARACTER alignment-char,  
    BOOLEAN   position-option,  
    BOOLEAN   underline,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 62. Format of the AFPPCHS Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Current Handle

The handle for the current page, area, or table, returned from the AFPBPAG, AFPCARE, or AFPBTBL calls.

##### String Length

The number of characters in the character string; the maximum number is determined by your compiler.

##### Character String

The character string, including any leading and trailing blanks in the strings, as indicated by the string length. The string cannot contain any null characters (X'00').

**Alignment Option**

The horizontal alignment option for the character data:

**Right**

Places the last character in the string at the current position.

**Left**

Places the first character in the string at the current position.

**Center**

Centers the character string around the current position.

**Character**

Places the character, specified in the alignment character parameter, at the current position. If the designated character is not found in the string, the string is right-aligned.

**Alignment Character**

The character for character alignment; it cannot be a space (X'40'). This parameter is ignored for all other alignment options.

**Position Option**

Indicates whether the current position is updated at the conclusion of this function. If this parameter is set to TRU, the current position remains at the origin of the string. Otherwise, the current position is moved to the position at which the next character would be placed.

**Note:** This parameter causes a line break when you place a character string in a table field. If this parameter is set to TRU, the character string begins at the start of a new line.

**Underline**

Indicates whether the character string (including blanks) will be underlined. If underline is specified, the width and position of the underline are taken from the current font.

**Output Parameters****Return Code**

The return code for this call.

**Severity Code**

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0020** The alignment option is invalid. Severity 8 (ERROR).
- 0028** The Y position is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0051** The state is invalid; the state must be page or area. Severity 8 (ERROR).
- 0127** The string length is invalid. Severity 8 (ERROR).
- 0202** The space in the area, page, or paragraph is not wide enough for the text. Severity 8 (ERROR).



## AFPPRUL (Put Rule)

### Function

Draws a rule from the current position extending in the specified X or Y direction using the current color value and rule thickness for the rule. The rule thickness extends below horizontal rules and to the right of vertical rules. At the end of this call, the current position is unchanged.

**Note:** Vertical rules in a framed area are not included in the area depth, because the current position is unchanged when a vertical rule is drawn. This means that the bottom rule of the area may not enclose the vertical rule.

### Syntax

```
AFPPRUL(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    INTEGER4  direction,
    REAL      rule-length,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 63. Format of the AFPPRUL Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current page or area, returned from the AFPBPAG or AFPCARE calls.

#### Direction

The direction of the rule is either parallel to the X axis or parallel to the Y axis.

#### Rule Length

The rule length in the current unit of measure.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0028** The Y position is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0066** The state is invalid; the state must be page or area. Severity 8 (ERROR).
- 0067** The direction is invalid. Severity 8 (ERROR).
- 0169** The rule length is invalid. Severity 8 (ERROR).
- 0213** The maximum depth specified in the area, page, or table was exceeded; the rule will not fit. Severity 4 (WARNING).

## AFPPTAG (Put Tag)

### Function

Creates an indexing tag in the document for archiving or viewing purposes. See “Indexing Data for Viewing and Archiving” on page 79 for more information about using this procedure call.

### Syntax

```
AFPPTAG(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    STRING    tag-name,
    STRING    tag-value,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 64. Format of the AFPPTAG Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document or page returned from the AFPBDOC or the AFPBPAG call.

#### Tag Name

The name of the indexing attribute, encoded using code page T1V10500. The maximum number of characters in the attribute name is 64, including blanks, and the name cannot contain any single quotes (') or null characters (X'00').

#### Tag Value

The value of the indexing attribute, encoded using code page T1V10500. The maximum number of characters in the attribute value is 64, including blanks, and the name cannot contain any single quotes (') or null characters (X'00').

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0228** A Put Tag procedure call attempted to put a tag in document state (between pages), but no group is active. Severity 8 (ERROR).
- 0272** The state is invalid; the state must be document or page. Severity 8 (ERROR).
- 0273** The attribute name is invalid. Severity 8 (ERROR).
- 0274** The attribute value is invalid. Severity 8 (ERROR).

---

## AFPPTXT (Put Text)

### Function

Places text in the paragraph or field. Text flows to fit the characteristics of the paragraph or field.

**Note:** If you place text at a valid position with the AFPSPPOS call, but the top of the text extends beyond the boundaries of a page or area, the results may not be what you expect, and the text may not print. AFP API does not issue an error return code for this situation.

### Syntax

```
AFPPTXT(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    INTEGER4  string-length,  
    STRING    character-string,  
    BOOLEAN   concatenate,  
    BOOLEAN   underline,  
    INTEGER4  remaining-length,  
    STRING    remaining-string,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 65. Format of the AFPPTXT Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current paragraph or table, returned from the AFPBPAR or AFPBTBL calls.

#### String Length

The number of characters in the string; the maximum number is determined by your compiler.

#### Character String

The string of characters in the AFPPTXT call. The string cannot contain any null characters (X'00').

## Put Text

### Concatenate

Character string concatenation. If the Concatenate parameter is set to TRU, the character string in this AFPPTXT (Put Text) call is concatenated with a previous AFPPTXT (Put Text). If Concatenate is set to FALS, the character string in this AFPPTXT (Put Text) call begins on a new line.

On the first AFPPTXT (Put Text) call in a paragraph or field, set the Concatenate parameter to TRU. On subsequent calls, set the Concatenate parameter as desired.

### Underline

Character string underline. If the underline parameter is specified, the width and position of the underline are taken from the current font.

## Output Parameters

### Remaining Length

The number of characters that could not be placed in the character string if the depth of the area or page is exceeded.

### Remaining String

The character string that could not be placed in the paragraph because the depth of the page or the maximum depth of the area was exceeded.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0190** The state is invalid. Severity 8 (ERROR).
- 0202** The space in the area, page, paragraph, or table field is not wide enough for the text. Severity 8 (ERROR).
- 0213** The maximum depth specified in the area, page, or table was exceeded; the text will not fit. Severity 4 (WARNING).

## AFPQATT (Query Current Attributes)

### Function

Returns the current values for units, position, color, rule thickness, font, intercharacter spacing, and word spacing. Because of rounding, the numeric attribute values returned may be slightly different than the ones specified.

### Syntax

```
AFPQATT(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    INTEGER4  units-of-measure,
    REAL      x-coordinate,
    REAL      y-coordinate,
    INTEGER4  color,
    REAL      rule-thickness,
    STRING    font-id,
    REAL      character-spacing,
    SREAL     word-spacing,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 66. Format of the AFPQATT Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page, or area, returned from the AFPBDOC, AFPBPAG, or AFPCARE calls.

### Output Parameters

#### Units of Measure

The current unit of measure.

#### X Coordinate

The current X position returned in the current unit of measure.

#### Y Coordinate

The current Y position returned in the current unit of measure.

#### Color

The current color.

#### Rule Thickness

The current rule thickness in the current unit of measure.

#### Font ID

The ID of the current font.

## Query Current Attributes

### Character Spacing

The current intercharacter spacing returned in the current unit of measure.

### Word Spacing

The current word spacing returned in the current unit of measure.

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0002** The state is invalid; the state must be document, page, or area. Severity 8 (ERROR).
- 0005** The handle is invalid. Severity 8 (ERROR).
- 0013** A null rule thickness parameter is specified. Severity 8 (ERROR).
- 0025** A null font ID parameter is specified. Severity 8 (ERROR).
- 0030** A null intercharacter space parameter is specified. Severity 8 (ERROR).
- 0034** A null word space parameter is specified. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0181** A null units parameter is specified. Severity 8 (ERROR).
- 0182** A null X coordinate parameter is specified. Severity 8 (ERROR).
- 0183** A null Y coordinate parameter is specified. Severity 8 (ERROR).
- 0184** A null color parameter is specified. Severity 8 (ERROR).



---

## AFPQPOS (Query Current Position)

### Function

Returns the current position in the current unit of measure. Because of rounding, the numeric values returned may be slightly different from the ones specified on the AFPSPPOS call.

### Syntax

```
AFPQPOS(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    REAL      x-coordinate,  
    REAL      y-coordinate,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 67. Format of the AFPQPOS Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current page or area, returned from the AFPBPAG or AFPCARE calls.

### Output Parameters

#### X coordinate

The X coordinate of the current position returned in the current unit of measure.

#### Y coordinate

The Y coordinate of the current position returned in the current unit of measure.

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0002** The state is invalid; the state must be page or area. Severity 8 (ERROR).
- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).

## AFPQSTR (Query Character String Size)

### Function

Returns the width and depth of a character string, using the current values for intercharacter spacing, word spacing, and font. AFPQSTR uses the font set in the last call to AFPSFNT as the current font. If you have not called AFPSFNT to set a font, AFPQSTR uses the default font.

### Syntax

```
AFPQSTR(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    STRING    character-string,
    REAL      string-length,
    REAL      measured-width,
    REAL      line-spacing,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 68. Format of the AFPQSTR Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current page, area, paragraph, or table, returned from the AFPBPAG, AFPCARE, AFPBPAR, or AFPBTBL calls, respectively.

#### Character String

The character string whose size is to be returned. The maximum size of the character string is determined by your compiler.

#### String Length

The number of characters in the character string, including blanks. Ensure that you specify the correct length of the character string; AFPQSTR determines the width of the string using the number of characters you specify as the string length.

### Output Parameters

#### Measured Width

The width of the character string, in the current unit of measure. AFPQSTR ignores any justification value specified for the paragraph in an AFPBPAR call or for the table field in an AFPDFLD call when determining the width of the character string.

#### Line Spacing

The depth of the character string, in the current unit of measure. The depth is the default line spacing associated with the current font. AFPQSTR

## Query Character String Size

ignores any line-spacing value specified for the paragraph in an AFPBPAR call or for the table field in an AFPDFLD call when determining the depth of the character string.

### **Return Code**

The return code for this call.

### **Severity Code**

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0284** The state is invalid; the state must be page, area, paragraph, or field. Severity 8 (ERROR).
- 0285** The character-string length is invalid. Severity 8 (ERROR).

---

## AFPSCLR (Set Color)

### Function

Specifies the color for subsequent data (text and rules).

### Syntax

```
AFPSCLR(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    INTEGER4  color,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 69. Format of the AFPSCLR Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page, area, paragraph, or table, returned from the AFPBDOC, AFPBPAG, AFPCARE, AFPBPAR, or AFPBTBL calls.

#### Color

The color to be printed:

- Black
- Blue
- Red
- Magenta
- Green
- Cyan
- Yellow
- Brown
- Color of medium

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0016** The color parmater is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0083** The state is invalid; the state must be document, page, area, paragraph, or field.

## AFPSFNT (Set Font)

### Function

Specifies the font for subsequent text data.

**Note:** A code page is a particular assignment of hexadecimal identifiers to graphic characters. For example, X'4F' is an exclamation point (!) in code page T1V10500, and X'5A' is an exclamation point (!) in code page T1V10037. If the code page used by your terminal or system for input is different from the code page used to present the data, you may not get the output you expect.

### Syntax

```
AFPSFNT(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    STRING    font-id,
    INTEGER4  ret-code,
    INTEGER4  severity-code
```

Figure 70. Format of the AFPSFNT Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page, area, paragraph, or table, returned from the AFPBDOC, AFPBPAG, AFPCARE, AFPBPAR, or AFPBTBL calls.

#### Font ID

The font ID returned from the AFPDFNT (Define Font by Attributes) call.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0056** The state is invalid; the state must be document, page, area, paragraph, or field. Severity 8 (ERROR).
- 0063** The font was not defined in a Define Font procedure call. Severity 8 (ERROR).
- 0201** An error occurred reading the font library. Severity 12 (SEVERE).
- 0212** The font index cannot be not found or cannot be read. Severity 12 (SEVERE).
- 0214** The formatter cannot start the requested font. Severity 8 (ERROR).
- 0217** The font specified cannot be used. Severity 8 (ERROR).
- 0218** The requested code page contains characters that are not in the current character set. Severity 8 (ERROR).



---

## AFPSICS (Set Intercharacter Spacing)

### Function

Specifies spacing (in addition to the character increment associated with the character) between the individual characters in a word in the current unit of measure. The space is in the positive inline direction. AFPSWSP (Set Word Spacing) controls the spacing between words.

### Syntax

```
AFPSICS(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    REAL      character-spacing,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 71. Format of the AFPSICS Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page, area, paragraph, or table returned from the AFPBDOC, AFPBPAG, AFPCARE, AFPBPAR, or AFPBTBL calls.

#### Character Spacing

The amount of space to be inserted between the characters in a word as a positive value.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0057** The state is invalid; the state must be document, page, area, paragraph, or field. Severity 8 (ERROR).
- 0187** The intercharacter spacing is invalid. Severity 8 (ERROR).

## AFPSLIB (Set Resource Library Names)

### Function

Establishes the names of the print resource libraries used by AFP API.

This procedure call is ignored in VSE and in a CICS/ESA environment:

- In VSE, the names of the resource libraries are specified in the // LIBDEF PHASE,SEARCH=(...) JCL statement.
- In a CICS/ESA environment, page segments and fonts must be located in VSAM data sets defined to CICS/ESA with file names of SEGLIB and FONTLIB.

### Syntax

```
AFPSLIB(
    HANDLE    AFPAPI-handle,
    TOKEN     pseg-library,
    TOKEN     object-library,
    TOKEN     font-library,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 72. Format of the AFPSLIB Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Page Segment Library

The name of the library that contains the page segment:

- For MVS, the name of the DD card that specifies the page segment library.  
Default if this call is not issued: PSEGDD.
- For VM, the file type of the page segment library.  
Default if this call is not issued: PSEG3820.  
AFP API searches the disks alphabetically, based on the file mode value, for the first minidisk where the object is stored; that is, AFP API searches for "filename PSEG3820 \*."
- For VSE, this procedure call is ignored; the name of the page segment library is specified in the // LIBDEF PHASE,SEARCH=(...) JCL statement.
- For CICS/ESA, this procedure call is ignored; the page segment data set is defined to CICS/ESA with a file name of SEGLIB.

## Set Resource Library Names

### Object Library

The name of the library that contains the objects:

- For MVS, the name of the DD card that specifies the object library.  
Default if this call is not issued: OBJTDD.
- For VM, the file type of the object file library.  
Default if this call is not issued: OBJT3820.  
AFP API searches the disks alphabetically, based on the file mode value, for the first minidisk where the object is stored; that is, AFP API searches for "filename OBJT3820 \*."
- For VSE, this procedure call is ignored; the name of the object library is specified in the // LIBDEF PHASE,SEARCH=(...) JCL statement.
- For CICS/ESA, this procedure call is ignored; objects must be included within page segments.

### Font Library

The name of the library that contains the fonts:

- For MVS, the name of the DD card that specifies the font library.  
Default if this call is not issued: FONTDD.
- For VM, the file type of the font library.  
Default if this call is not issued: FONT3820.  
AFP API searches the disks alphabetically, based on the file mode value, for the first minidisk where the object is stored; that is, AFP API searches for "filename FONT3820 \*."
- For VSE, this procedure call is ignored; the name of the font library is specified in the // LIBDEF PHASE,SEARCH=(...) JCL statement.
- For CICS/ESA, this procedure call is ignored; the font data set is defined to CICS/ESA with a file name of FONTLIB.

## Output Parameters

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0196** The state is invalid. Severity 8 (ERROR).
- 0197** A null page segment library parameter is specified. Severity 8 (ERROR).
- 0198** A null object library parameter is specified. Severity 8 (ERROR).
- 0199** A null font library parameter is specified. Severity 8 (ERROR).

### AFPSOUT (Set Output Characteristics)

#### Function

Defines where API is to write the AFP output records. Output can be written to an output file, to a CICS/ESA temporary storage queue, or to a buffer in your program.

If output is written to an output file, specifies:

- The maximum size of an output record
- The name of the output file
- Whether or not to replace an existing file

If output is written to a CICS/ESA temporary storage queue, specifies:

- The name of the queue
- The maximum size of an output record

If output is written to an output buffer, specifies:

- The maximum size of an output record

**Note:** Issue an AFPGBUF call, described in “AFPGBUF (Get Output Buffer)” on page 149, to obtain records written to an output buffer.

#### Syntax

```
AFPSOUT(  
    HANDLE           AFPAPI-handle,  
    INTEGER4        output-record-size,  
    FILENAME-TOKEN  output-filename,  
    FILETYPE-TOKEN  output-filetype,  
    FILEMODE-TOKEN  output-filemode,  
    BOOLEAN         replace,  
    INTEGER4        ret-code,  
    INTEGER4        severity-code  
)
```

Figure 73. Format of the AFPSOUT Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Output Record Size

The maximum size of a data stream record that is written as a variable length record to an output file, a CICS/ESA temporary storage queue, or an output buffer. A data stream record consists of a single structured field; therefore, the value should be size of the largest possible structured field. Default if this call is not issued: 8205 bytes.

If the output is written to an output file or to a CICS/ESA temporary storage queue, the smallest value that you can specify is 512 bytes, and the largest

value is 8205 bytes. For MVS, in a non-CICS/ESA environment, the value specified must not exceed the length in the DCB.

If the output is written to an output buffer, the smallest value that you can specify is 512 bytes, and the largest value is 32 767 bytes.

### Output File Name

Specifies the name of the output file or CICS/ESA temporary storage queue, as follows:

- For MVS, either the name of the DD statement that identifies the output file, or the name of the CICS/ESA temporary storage queue.  
Default if this call is not issued: OUTFDD.
- For VM, the file name of the output file.  
Default if this call is not issued: OUTFILE.
- For VSE, the name specified in the DLBL JCL statement that identifies the output file.  
Default if this call is not issued: OUTFDD.

Or, specifies one of the following constants, which are defined in the APQCONST and APQPCON copy books:

- BUFFERED, which requests that AFP API write output to an output buffer.
- DISCBUFF, which requests that AFP API discard the output from each page rather than converting it to AFPDS output. You can use this function to count pages without creating any output.

### Output File Type

For VM, specifies the filetype of the output file. For MVS and VSE, this parameter is ignored.

Default if this call is not issued: LISTAFP.

### Output File Mode

For VM, specifies the mode of the output file. For MVS and VSE, this parameter is ignored.

Default if this call is not issued: \*.

### Replace

For VM, specifies whether to replace an existing output file with the output of AFP API. The default if this call is not issued is not to replace the existing file.

A severe return code is issued if the file exists and if you do not specify that the file is to be replaced.

For MVS and VSE, this parameter is ignored. If an output file exists, it is replaced; if a CICS/ESA temporary queue exists, the output is written at the end of the existing data. For VM, if output is written to an output buffer or discarded, this parameter is ignored.

## Output Parameters

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0006** The state is invalid; the state must be start. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0073** The mode is invalid. Severity 8 (ERROR).
- 0115** The output file ID is invalid. Severity 8 (ERROR).



## AFPSPOS (Set Position)

### Function

Sets the position in the current unit of measure.

### Syntax

```
AFPSPOS(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    REAL      x-coordinate,
    INTEGER4  x-ref-coord-sys,
    REAL      y-coordinate,
    INTEGER4  y-ref-coord-sys,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 74. Format of the AFPSPOS Procedure Call

## Input Parameters

### AFPAPI Handle

The session handle returned from the AFPINIT call.

### Current Handle

The handle for the current page or area, returned from the AFPBPAG or AFPCARE calls.

### X Coordinate

Contains the X coordinate of the position in the current unit of measure.

You can specify an X coordinate that is less than, greater than, or equal to the current X coordinate. See “AFP Documents and Pages” on page 8 for an explanation of the coordinate system.

**Note:** To specify an X coordinate that is less than the current X coordinate, specify the coordinate as an offset from the current page or area origin.

### X Reference Coordinate System

Indicates whether the X coordinate is an offset from the current page or area origin (absolute) or whether it is an offset from the current X coordinate (relative).

### Y Coordinate

Contains the Y coordinate of the position either in the current unit of measure or in lines of text, depending on the value of the Y reference coordinate system parameter.

If you are placing an area or placing data, you can specify a Y coordinate that is less than, greater than, or equal to the current Y coordinate. However, if you are placing data *within* an area, you must specify a Y coordinate that is greater than or equal to the current Y coordinate; that is, you can *not* specify a Y coordinate less than the current Y coordinate. See

## Set Position

“AFP Documents and Pages” on page 8 for an explanation of the coordinate system.

**Note:** To specify a Y coordinate less than the current Y coordinate, specify the coordinate as an offset from the current page origin.

### Y Reference Coordinate System

Indicates if the Y coordinate is an offset from the current page or area origin (absolute), if it is an offset from the current Y coordinate (relative), or if it is the number of lines to move from the current Y-coordinate position (lines).

The line space is derived from the current font. If lines is specified, the current unit of measure has no effect on the Y coordinate value.

## Output Parameters

### Return Code

The return code for this call.

### Severity Code

The severity of the return code.

## Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0024** The state is invalid; the state must be page or area. Severity 8 (ERROR).
- 0029** The coordinate is invalid; the coordinate must be a positive number. Severity 8 (ERROR).
- 0035** A numeric overflow occurred; the specified relative value added to the current position exceeded the maximum valid value. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0059** The Y reference coordinate system is invalid. Severity 8 (ERROR).
- 0138** The X position is invalid. Severity 8 (ERROR).
- 0139** The Y position is invalid. Severity 8 (ERROR).
- 0142** The X reference coordinate system is invalid. Severity 8 (ERROR).

## AFPSRTH (Set Rule Thickness)

### Function

Specifies the rule thickness for subsequent rules in the current unit of measure. For vertical rules, the rule thickness extends in the positive (inline) direction. For horizontal rules, the rule thickness extends in the positive Y direction.

### Syntax

```
AFPSRTH(
    HANDLE    AFPAPI-handle,
    HANDLE    current-handle,
    REAL      rule-thickness,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 75. Format of the AFPSRTH Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page, or area, returned from the AFPBDOC, AFPBPAG, or AFPCARE calls.

#### Rule Thickness

The new rule thickness to be used for subsequent rules.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0058** The state is invalid; the state must be document, page, or area. Severity 8 (ERROR).
- 0140** The rule thickness is invalid. Severity 8 (ERROR).

---

## AFPSUNI (Set Units)

### Function

Sets the current units of measure.

**Note:** The output file generated by AFP API is in logical units of 1440 per inch, even if you specify a different unit of measure in this parameter.

### Syntax

```
AFPSUNI(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    INTEGER4  unit-of-measure,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 76. Format of the AFPSUNI Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page or area, returned from the AFPBDOC, AFPBPAG, or AFPCARE calls.

#### Units of Measure

The unit of measure: inches, millimeters, centimeters, 240 units, or 1440 units.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, "Return Codes and Severity Codes" for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0017** The unit specified is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0085** The state is invalid; the state must be document, page, or area. Severity 8 (ERROR).

---

## AFPSWSP (Set Word Spacing)

### Function

Specifies the width of spaces between words in the current unit of measure.

### Syntax

```
AFPSWSP(  
    HANDLE    AFPAPI-handle,  
    HANDLE    current-handle,  
    SREAL     word-spacing,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 77. Format of the AFPSWSP Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

#### Current Handle

The handle for the current document, page, area, paragraph, or table, returned from the AFPBDOC, AFPBPAG, AFPCARE, AFPBPAR, or AFPBTBL calls.

#### Word Spacing

The width of spaces between words in the current unit of measure. You can specify a value or use the default word spacing associated with the current font. If the default is specified, the word spacing associated with the current font is used.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0086** The state is invalid; the state must be document, page, area, paragraph, or field. Severity 8 (ERROR).
- 0141** The word spacing is invalid. Severity 8 (ERROR).



## AFPTERM (Terminate AFP API)

### Function

Abnormally terminates AFP API and frees all storage. If a partial page was created, the partial page is written to the output file. This call is useful when debugging your program.

The only AFP API procedure call you can issue after AFPTERM is AFPINIT. See “AFPEND (End AFP API)” on page 140 for normal termination.

### Syntax

```
AFPTERM(
    HANDLE    AFPAPI-handle,
    INTEGER4  ret-code,
    INTEGER4  severity-code
)
```

Figure 78. Format of the AFPTERM Procedure Call

### Input Parameters

#### AFPAPI Handle

The session handle returned from the AFPINIT call.

### Output Parameters

#### Return Code

The return code for this call.

#### Severity Code

The severity of the return code.

### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

- 0005** The handle is invalid. Severity 8 (ERROR).
- 0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).
- 0210** The formatter cannot write to the output file specified with the Set Output Characteristics procedure call. Severity 12 (SEVERE).
- 0278** The state is invalid. Severity 8 (ERROR).

### AFPXARE (Destroy Area)

#### Function

Deletes an area and all its contents from AFP API storage. Delete the area when it is no longer needed. For more information, see “ AFPCARE (Create Area)” on page 121.

#### Syntax

```
AFPXARE(  
    HANDLE    AFPAPI-handle,  
    HANDLE    area-handle,  
    INTEGER4  ret-code,  
    INTEGER4  severity-code  
)
```

Figure 79. Format of the AFPXARE Procedure Call

#### Input Parameters

##### AFPAPI Handle

The session handle returned from the AFPINIT call.

##### Area Handle

The handle of the area to be destroyed in AFP API storage, returned from the AFPCARE call.

#### Output Parameters

##### Return Code

The return code for this call.

##### Severity Code

The severity of the return code.

#### Return Codes

The following return codes indicate possible errors in your application program. In addition to the return codes listed here, you may receive codes that indicate an internal logic error (severity code 16) or that your program is out of memory (severity code 12). See Appendix B, “Return Codes and Severity Codes” for more information about each return code and the meaning of the severity codes.

**0005** The handle is invalid. Severity 8 (ERROR).

**0036** Invalid null handle was passed to an AFP API procedure. Severity 8 (ERROR).

---

## Appendix A. Font Library Indexing Program (FLIP)

This appendix describes invoking the Font Library Index Program (FLIP) on the system, in case the program directory is not available. Usually, a system programmer will install FLIP for use by the entire installation.

To use AFP API, your font library and the font library index created by FLIP, must contain the default font and any other fonts used by your applications. AFP API always requires that the default font be available, even though an application does not use the default font. The name of the default font is X0N2100C. To obtain fonts from IBM, order the AFP Font Collection product.

---

### Invoking the Font Library Index Program in VM

In VM, a font library is a collection of CMS files, with a common (single) default or user-specified file type, residing on a VM minidisk or its extensions. Font objects are individual CMS files, and the font library index created by FLIP will be a CMS file with the file name AFPINDEX<sup>8</sup> and the same file type as the font members.

In CMS, FLIP takes two parameters: the file type of the font objects and the file mode where the font search should begin. If FLIP is invoked without any parameters, a file type of FONT3820 and a file mode of "a" is used.

To create the index, FLIP examines each file of the library on the specified minidisk and its extensions and, if the file is a font object, adds an index entry.

For example, consider a font library consisting of the following files:

```
COB400N0 FONT3820 g1
COB400H0 FONT3820 g1
COB400F0 FONT3820 g1
COB400D0 FONT3820 g1
COB400B0 FONT3820 g1
COB40090 FONT3820 g1
COB30090 FONT3820 g1
COB500D0 FONT3820 g1
```

To create the font library index, the "G" minidisk must be accessed in READ/WRITE mode, and the following command must be issued:

```
flipvm font3820 g
```

The FLIPVM program will create a CMS file named AFPINDEX FONT3820 and a report named FONT3820 LISTING. AFPINDEX FONT3820 will have file mode "G1," because only "G" was specified. You can also request that the file be assigned a specific file mode number by specifying it explicitly. For example, specifying:

```
flipvm font3820 g5
```

---

<sup>8</sup> The FLIP program that comes with AFP API is similar to the one available with Document Composition Facility (DCF). The AFPINDEX file is identical in content to the DCFINDEX file, so if you have already installed DCF and have run FLIP, you may not need to re-run it for AFP API. However, the listing files produced from AFPINDEX are quite different, so you may want to keep both. AFP API searches first for AFPINDEX and then for DCFINDEX. DCF does not recognize or process the AFPINDEX file.

will assign the file with a file mode of "G5."

The report will be created with a file type of LISTING and stored on the same disk as the index. In the previous example, the report will be written to a file named FONT3820 LISTING G5.

To improve performance in the CMS environment, you should not have the disk that contains the existing AFPINDEX as your primary READ/WRITE disk. Instead, set up a temporary disk for the purpose of holding the new AFPINDEX and make the font disks extensions of that disk. This enables others to use the fonts while FLIP is running.

The order of extensions used by FLIP to build the AFPINDEX must be the same for AFP API and Print Services Facility (PSF). If AFP API does not use the same order as specified when FLIP was run, formatting results are unpredictable, because the font metrics and contents may differ. If PSF uses a different order, the wrong font member may be selected if objects of the same name reside on different minidisks.

---

## Invoking the Font Library Index Program in MVS

In MVS, a font library is a partitioned data set; font objects are partitioned data set members (to a maximum of 8192); and the font library index will be created as a member named AFPINDEX.

FLIP in MVS accepts as input a data set name or a DD name (FONTLIB) to enable you to concatenate multiple data sets for the font library. A separate ddname, FONTLIBO, is used with FLIP to specify the single partitioned data set that will contain the AFPINDEX.

If concatenation is used for FONTLIB, you must use the FONTLIBO ddname to specify the single partitioned data set that will contain the AFPINDEX created by FLIP.

If your FONTLIB is not concatenated, FONTLIBO is ignored. The AFPINDEX will be created in the single partitioned data set referred to by FONTLIB.

If FONTLIB is a concatenated data set, only the first occurrence of each member is used. Subsequent (duplicate) members found in FONTLIB are ignored.

When you use concatenation for FONTLIB with FLIP, the same concatenation order must be used with AFP API and PSF. If AFP API does not use the same concatenation order as FLIP, formatting results are unpredictable.

For example, a small set of fonts in SYS1.FONT3820 might contain the following member names:

```
COB400NO  
COB400HO  
COB400FO  
COB400DO  
COB400BO  
COB40090  
COB30090  
COB500DO
```

To create the index, FLIP must examine each member of the library listed in the partitioned data set directory and, if the member is a font object, add an index entry for the font.

The following sample shows part of the JCL needed to create the font library index:

```
//INDEX JOB ...  
// EXEC PGM=FLIPMVS  
//STEPLIB DD DSN=***,****,***** ,DISP=SHR  
//SYSPRINT DD SYSOUT=A  
//FONTLIB DD DISP=OLD,DSN=SYS1.FONT3820  
//
```

**Note:** STEPLIB specifies the data set on the system where FLIPMVS is installed.

The FLIPMVS program adds two members to the font library data set in the example named SYS1.FONT3820:

- AFPINDEX, which AFP API needs at run time
- LISTING, which is a human-readable listing of the font library contents shown in Figure 24 on page 71

---

## Invoking the Font Library Index Program in a CICS/ESA Environment

When running AFP API in CICS/ESA environment, fonts must be stored in a key-sequenced VSAM data set defined to CICS/ESA with a file name of FONTLIB. After invoking FLIP, your system programmer must copy the font partitioned data set, including the index, to a VSAM data set.

To do this, AFP API provides program APQCIVSM and associated JCL in file APQCIFON. Your system programmer must modify and run the JCL in APQCIFON at installation and whenever the font library is modified.

---

## Invoking the Font Library Index Program in VSE

In VSE/AF Version 2, fonts are stored in a sublibrary. Font objects are library phases (to a maximum of 8192), and the font library index is a phase named AFPINDEX.

For example, a small set of Monotype Times New Roman fonts in the FONT3820.LIBRARY might contain the following phases:

```
COB400N0 FONT3820 g1  
COB400H0 FONT3820 g1  
COB400F0 FONT3820 g1  
COB400D0 FONT3820 g1  
COB400B0 FONT3820 g1  
COB40090 FONT3820 g1  
COB30090 FONT3820 g1  
COB500D0 FONT3820 g1
```

To create the index, FLIP must examine each member of the library listed in the library directory, and, if the member contains a font object, create an index entry for the font. The AFPINDEX phase composed of the index is produced by using the VSE linkage editor; APQFLVSE produces the input for the linkage editor.

In VSE/AF Version 2, the following job must be executed to create the font library index. In the example, *api.code.library* is the name of the library where the FLIP phase resides, and *afp.fontlib* is the name of the font library.

```
// JOB AFPINDEX
  LIBDEF PHASE,SEARCH=(api.code.library,afp.fontlib),
  CATALOG=afp.fontlib
// DLBL IJSYSPH,'AFP.SYSPCH.FILE1',0,SD
// EXTENT ,SYSWK1,,,150,50
ASSGN SYSPCH,3350,VOL=SYSWK1,SHR
// EXEC APQFLVSE,SIZE=AUTO
/*
CLOSE SYSPCH,PUNCH
// DLBL IJSYSIN,'AFP.SYSPCH.FILE1',99/365
// EXTENT SYSIPT,SYSWK1
ASSGN SYSIPT,3350,PERM,VOL=SYSWK1,SHR
// EXEC LIBR,PARM='A S=afp.fontlib'
/*
  LIBDEF OBJ,SEARCH=afp.fontlib
// OPTION CATAL
  PHASE AFPINDEX,*
  INCLUDE AFPINDEX
/*
// EXEC LNKEDT
CLOSE SYSIPT,UANCH
/*
/&
```

The sublibrary list specified in the VSE LIBDEF statement for the EXEC APQFLVSE job step must not contain any empty sublibraries. If an empty sublibrary is specified, APQFLVSE will terminate with a return code of 32.

---

<sup>9</sup> UA is an installation-dependent unit address of the SYSIPT before this job is executed. See your system programmer for the correct address.

---

## Font Library Index Program Return Codes

The Font Library Index Program sets a return code resulting from index processing. Table 3 lists these return codes.

<i>Table 3. Font Library Index Program Return Codes</i>	
<b>Return Code</b>	<b>Meaning</b>
0	Normal completion
4	Listing file OPEN error
8	No font objects in the font library
12	Font library OPEN error
16	Not enough storage available for processing
20	Font library read error
24	Unable to FIND font object in the font library
28	Unable to read or write to disk
32	Librarian macro services error
36	Fontlib data set blocksize too small
40	Overflow from the font-library member-name table, in MVS and VSE. The input font library is too large to be processed by FLIP. The library must be subdivided into two or more smaller font libraries.
44	Unable to access disk specified with file mode (VM)





---

## Appendix B. Return Codes and Severity Codes

This appendix uses the following format to provide information about the return codes for AFP API procedures:

1234

**A text description of the return code.**

**Severity:** The severity level of the return code. The five levels are:

- 0** Success. The AFP API procedure successfully completed.
- 4** Warning. The AFP API procedure placed data outside the depth boundary of the page or area. The AFP API procedure did not successfully complete. AFP API can continue.
- 8** Error. An error occurred, and the AFP API procedure did not successfully complete. AFP API can continue.
- 12** Severe. An error occurred, and the AFP API procedure did not successfully complete. AFP API cannot continue. This is probably an application programming logic error.
- 16** Fatal. An error occurred, and the AFP API procedure did not successfully complete. AFP API cannot continue. Contact your IBM service representative in the IBM Support Center, because this is probably an AFP API logic error.

**AFP API Procedures:** A list of the procedure calls that may have issued the return code.

**Response:** What you can do to correct the situation.

**Return Code Constants:** Return code constants for the COBOL and PL/1 languages. COBOL is shown; for PL/1, replace the hyphen (-) with an underscore (\_). These constants are included in your program if you include the copy file APQRCS COPY.

---

## AFP API Return Codes

**0000****No error.****Severity:** 0 (SUCCESS)**AFP API Procedures:** All**Response:** No response is necessary.**Return Code Constants:** None**0001****AFP API was unable to retrieve the specific error information.****Severity:** 16 (FATAL)**AFP API Procedures:** All**Response:** Contact your IBM service representative, and report this return code.**Return Code Constants:** ER-FAIL**0002****AFP API was in an invalid state when you issued a Query Current Attributes or Query Current Position procedure call. The state must be document, page, or area when you issue a Query Current Attributes procedure call; the state must be page or area when you issue a Query Current Position procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Query Current Attributes or Query Current Position**Response:** Ensure that the handle on the AFP API procedure call in error is valid.**Return Code Constants:** ER-QATTS**0003****AFP API is unable to retrieve the current attributes. Block attribute pointer is NULL.****Severity:** 16 (FATAL)**AFP API Procedures:** Query Current Attributes**Response:** Contact your IBM service representative, and report this return code.**Return Code Constants:** ER-NOATTS**0004****Invalid state for Terminate AFP API. The state must be start.****Severity:** 8 (ERROR)**AFP API Procedures:** Terminate AFP API**Response:** Verify that the handle on the AFP API procedure call in error is valid.**Return Code Constants:** ER-TERM

0005

Invalid handle.

**Severity:** 8 (ERROR)**AFP API Procedures:** All**Response:** Verify that the handle on the AFP API procedure call in error is valid.**Return Code Constants:** ER-NOTFOUND

0006

Invalid state for a Set Output Characteristics procedure call. The state must be start.

**Severity:** 8 (ERROR)**AFP API Procedures:** Set Output Characteristics**Response:** Verify that the handle on the AFP API procedure call in error is valid.**Return Code Constants:** ER-SETOUT

0007

Area must be ended before attempting to put the area on the page.

**Severity:** 8 (ERROR)**AFP API Procedures:** Put Area**Response:** Verify that the End Area procedure call is issued before the Put Area procedure call.**Return Code Constants:** ER-NOTENDED

0008

Invalid state for a Begin Document procedure call. A document is already started.

**Severity:** 8 (ERROR)**AFP API Procedures:** Begin Document**Response:** Verify that the End Document procedure call is issued prior to the Begin Document procedure call.**Return Code Constants:** ER-DOCEXISTS

0009

Invalid state for a Begin Page procedure call. A page is already started.

**Severity:** 8 (ERROR)**AFP API Procedures:** Begin Page**Response:** Verify that the End Page procedure call is issued prior to the Begin Page procedure call.**Return Code Constants:** ER-PAGEXISTS

0011

**Invalid state for the End AFP API procedure call. The state must be start.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End AFP API

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-END

0012

**NULL area depth parameter is specified in the End Area procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Area

**Response:** Verify that an area depth parameter is specified in the End Area procedure call.

**Return Code Constants:** ER-MBAREA

0013

**NULL rule thickness parameter is specified in the Query Current Attributes procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Verify that a rule thickness parameter is specified on the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVLTHICK

0014

**Invalid router block type.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-IVTYPE

0015

**Invalid formatter block type.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-BLKTYPE

0016

**Invalid color specified in the Set Color procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Color

**Response:** Change the color parameter on the Set Color procedure call to a valid value.

**Return Code Constants:** ER-IVCOLOR

0017

**Invalid units specified.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Units and Begin Document

**Response:** Change the units parameter on the AFP API procedure call in error to a valid value.

**Return Code Constants:** ER-IVUNITS

0018

**Invalid page orientation specified.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Document and Begin Page

**Response:** Change the page orientation parameter on the AFP API procedure call in error to a valid value.

**Return Code Constants:** ER-IVROTATE

0019

**Invalid number of rows on the Define Row procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Row

**Response:** Change the number of rows on the Define Row procedure call to a valid value.

**Return Code Constants:** ER-IVNUMROWS

0020

**Invalid alignment option specified in the Put Character String procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Character String

**Response:** Change the alignment option parameter on the Put Character String procedure call to a valid value.

**Return Code Constants:** ER-IVALIGN

0021

Invalid state for the **Begin Document** procedure call. The state must be start.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Document

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-DPARENT

0022

Invalid state for the **Begin Page** procedure call. The state must be document.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Page

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PPARENT

0023

Invalid state for the **Create Area** procedure call. The state must be document or page.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Create Area

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-APARENT

0024

Invalid state for the **Set Position** procedure call. The state must be page or area.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Position

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-NOCURSOR

0025

NULL font ID parameter specified in the **Query Current Attributes** procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that a font ID parameter is specified on the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVFONTID

0026

Invalid block.

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-IVBLOCK

0027

**Invalid application control.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-IVCONTROL

0028

**A procedure call attempted to write data to an area, using an invalid value for the Y position. The current Y position must be greater than the Y position of data previously written in the area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph, Begin Table, Include Page Segment, Put Box, Put Character String, and Put Rule

**Response:** Ensure that the Y position specified in the most recent Set Position procedure call is valid.

**Return Code Constants:** ER-BACK

0029

**Invalid coordinate specified in the Set Position procedure call. The coordinate must be a positive number.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Position

**Response:** Change the coordinate parameter on the Set Position procedure call to a positive value.

**Return Code Constants:** ER-NEGATIVE

0030

**NULL intercharacter space parameter specified in the Query Current Attributes procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that an intercharacter space parameter is specified in the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVCSPACEP

0031

**Out of memory when trying to allocate a new block.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-BLKMEN

0032

The previous call to an AFP API procedure caused a SEVERE or FATAL error.

**Severity:** 12 (SEVERE)

**AFP API Procedures:** All

**Response:** End the AFP API by using Terminate AFP API.

**Return Code Constants:** None

0034

NULL word space parameter specified in the Query Current Attributes procedure call

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that a word space parameter is specified in the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVWSPACE

0035

Numeric overflow on the Set Position procedure call. The specified relative value when added to the current position exceeds the maximum valid value.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Position

**Response:** Change the relative coordinate value on the Set Position procedure call so that when it is added to the current position, the result does not exceed the maximum valid value.

**Return Code Constants:** ER-OVERFLOW

0036

Invalid NULL handle was passed to an AFP API procedure.

**Severity:** 8 (ERROR)

**AFP API Procedures:** All

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-NULLPTR

0037

Invalid NULL application control block pointer.

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NULLCONTROL



0038

**Invalid formatter block state. The state must be initial.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTINIT

0039

**Invalid formatter block state. The state must be start.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTSTRT

0040

**Invalid formatter block state. The state must be either active or start.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACST

0041

**Invalid formatter block state. The state must be active.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT

0042

**Invalid formatter block state. The state must be active when attempting to update the formatter's current position.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Begin Paragraph, Begin Table, Put Character String, Put Box, Put Rule, and Include Page Segment

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-MOV

0043

**Invalid formatter block state. The state must be active when attempting to write a character string.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Character String

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-PUT

0044

**Invalid formatter block state. The state must be ended.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** End Document, End Page, End Area, End AFP API, End Paragraph, and End Table

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTEND

0045

**Out of memory when trying to allocate block attributes.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-ATTSMEM

0046

**No block attribute pointer when trying to copy the block attributes.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOATTPTR

0048

**Out of memory when trying to allocate the formatter and data stream generator interface buffer.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-DCFMEM

0049

**Out of memory when trying to allocate the application control block.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-APPLMEM

0050

**Out of memory when trying to allocate additional area pointers in the application control block.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Create Area

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-AREAMEM

0051

**Invalid state for a Put Character String procedure call. The state must be page or area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Character String

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PUTSTR

0052

**Out of memory when trying to allocate the formatter and data stream generator formatting environment.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-DCFFENV

0053

**Invalid state for a Put Area procedure call. The state must be page.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Area

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PUTAREA

0054

**Invalid rotation specified in a Put Area procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Area

**Response:** Change the rotation parameter on the Put Area procedure call to a valid value.

**Return Code Constants:** ER-IVAREAROT

0055

**Invalid formatter block state. The state must be active.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-PUTA

0056

**Invalid state for a Set Font procedure call. The state must be document, page, area, paragraph, or field.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Font

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETFONT

0057

**Invalid state for a Set Intercharacter Spacing procedure call. The state must be document, page, area, paragraph, or field.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Intercharacter Spacing

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETCSPAC

0058

**Invalid state for a Set Rule Thickness procedure call. The state must be document, page, or area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Rule Thickness

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETLTHCK

0059

**Invalid Y reference coordinate system specified in a Set Position procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Position

**Response:** Change the Y reference coordinate system parameter on the Set Position procedure call to a valid value.

**Return Code Constants:** ER-IVYREF

0060

**Parent font array pointer is NULL when attempting to copy a block.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOFONTPTR

0061

**Out of memory when trying to allocate a font array.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-FONTMEM

0062

**Out of memory when trying to allocate a font attribute structure.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Define Font

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-FONTATSMEM

0063

**An AFP API procedure call referenced a font that was not defined in a Define Font procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query String Width and Set Font

**Response:** Add a Define Font procedure call to your program to define the font, or change the font reference in the procedure call in error to reference a font that is defined.

**Return Code Constants:** ER-FONTNOTFND

0064

**Invalid formatter block state. The state must be active when attempting to define a font.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Define Font

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-DEF

0065

**Invalid formatter block state. The state must be active when attempting to activate a font.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Character String and Put Text

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-SET

0066

**Invalid state for a Put Rule procedure call. The state must be page or area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Rule

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PUTLINE

0067

**Invalid direction specified in a Put Rule procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Rule

**Response:** Change the direction parameter on the Put Rule procedure call to a valid value.

**Return Code Constants:** ER-IVDIRECTION

0068

**Invalid position returned from formatter and data stream generator.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Begin Paragraph, Begin Table, Put Character String, Put Box, Put Rule, and Include Page Segment

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-DCFPOS

0069

**Invalid formatter block state. The state must be active when attempting to include an overlay.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Include Page Overlay

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-INC

0070

**Invalid state for an Include Page Overlay procedure call. The state must be page or area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Page Overlay

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-INCOVLY

0071

**Invalid area handle specified in a Put Area procedure call. The area does not exist.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Area

**Response:** Change the area handle on the Put Area procedure call to an active area handle. Ensure that you have created the area with a Create Area procedure call and ended the area with an End Area procedure call before you place the area with a Put Area procedure call.

**Return Code Constants:** ER-AREANOTFND

0072

**Numeric overflow on a Begin Paragraph procedure call. The specified left margin, when added to the current inline position, exceeds the maximum valid value.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the left margin parameter on the Begin Paragraph procedure call, so that when it is added to the current inline position, the result does not exceed the maximum valid value.

**Return Code Constants:** ER-MARG-OVERF

0073

**Invalid mode specified in a Set Output Characteristics procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Output Characteristics

**Response:** Change the mode parameter on the Set Output Characteristics procedure call to a valid value.

**Return Code Constants:** ER-IVFMODE

0074

**Invalid code page specified in a Define Font procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Font**Response:** Change the code page parameter on the Define Font procedure call to a valid value.**Return Code Constants:** ER-IVCODEPG

0075

**Invalid descriptive name specified in a Define Font procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Font**Response:** Change the descriptive name parameter specified in the Define Font procedure call to a valid value. See "Selecting the Font You Want" on page 68 for determining the valid descriptive names installed on your system.**Return Code Constants:** ER-IVDESCNM

0076

**Invalid point size specified in a Define Font procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Font**Response:** Change the point size parameter specified in the Define Font procedure call to a valid value.**Return Code Constants:** ER-IVPTSIZE

0077

**Invalid weight specified in a Define Font procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Font**Response:** Change the weight parameter specified in the Define Font procedure call to a valid value.**Return Code Constants:** ER-IVWEIGHT

0078

**Invalid width specified in a Define Font procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Font**Response:** Change the width parameter specified in the Define Font procedure call to a valid value.**Return Code Constants:** ER-IVWIDTH



0079

**Invalid rotation specified in a Define Font procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Font

**Response:** Change the rotation parameter specified in the Define Font procedure call to a valid value.

**Return Code Constants:** ER-IVFONTROT

0080

**Invalid style specified in a Define Font procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Font

**Response:** Change the style parameter specified in the Define Font procedure call to a valid value.

**Return Code Constants:** ER-IVSTYLE

0081

**Invalid formatter block state. The state must be active when attempting to include a page segment.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Include Page Segment

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-INCPSP

0082

**Invalid formatter block state. The state must be active when attempting to draw a rule.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Rule

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-PUTL

0083

**Invalid state for a Set Color procedure call. The state must be document, page, area, paragraph, or field.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Color

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETCOLOR

0084

**Invalid bottom horizontal rule thickness specified in a Define Row procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Row

**Response:** Change the bottom horizontal rule thickness parameter specified in the Define Row procedure call to a valid value.

**Return Code Constants:** ER-SETPGOR

0085

**Invalid state for a Set Units procedure call. The state must be document, page, or area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Units

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETUNITS

0086

**Invalid state for a Set Word Spacing procedure call. The state must be document, page, area, paragraph, or field.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Word Spacing

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETWORDSP

0087

**Invalid state for a Define Field procedure call. The state must be document.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-DEFFIELD

0088

**Invalid state for a Define Font procedure call. The state must be document, page, or area.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Font

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-DEFFONT

0089

**Invalid state for a Define Row procedure call. The state must be document.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Row

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-DEFROW

0090

**Parent row array pointer is NULL when attempting to copy a block.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOROWPTR

0091

**Out of memory when trying to allocate a row array.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, Create Area, and Define Row

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-ROWMEM

0092

**Parent field array pointer is NULL when attempting to copy a block.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, and Create Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOFLDPTR

0093

**Out of memory when attempting to allocate a field array.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API, Begin Document, Begin Page, Begin Table, Begin Paragraph, Create Area, and Define Field

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-FLDMEM

0094

**Invalid format option specified in a Define Field procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Field**Response:** Change the format option on the Define Field procedure call to a valid value.**Return Code Constants:** ER-IVHOR

0095

**Invalid vertical field format specified in a Define Field procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Field**Response:** Change the vertical field format parameter on the Define Field procedure call to a valid value.**Return Code Constants:** ER-IVVER

0096

**Invalid shading pattern specified in a Define Field procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Field**Response:** Change the shading pattern parameter on the Define Field procedure call to a valid value.**Return Code Constants:** ER-IVSHADE

0097

**Invalid shading intensity specified in a Define Field procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Field**Response:** Change the shading intensity parameter on the Define Field procedure call to a valid value.**Return Code Constants:** ER-IVSHINT

0098

**Invalid field text orientation specified in a Define Field procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Define Field**Response:** Change the text orientation parameter on the Define Field procedure call to a valid value.**Return Code Constants:** ER-IVFLDOR

0099

Invalid top horizontal rule thickness specified in a Define Field procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the top horizontal rule thickness parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVFLDFR

0100

Invalid shading pattern specified in a Create Area procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Create Area

**Response:** Change the shading pattern parameter on the Create Area procedure call to a valid value.

**Return Code Constants:** ER-IVARSHADE

0101

Invalid shading intensity specified in a Create Area procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Create Area

**Response:** Change the shading intensity parameter on the Create Area procedure call to a valid value.

**Return Code Constants:** ER-IVARSHINT

0102

Invalid depth specified in a Define Row procedure call.

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Define Row

**Response:** Change the depth parameter on the Define Row procedure call to a valid value.

**Return Code Constants:** ER-IVDEPTH

0103

Invalid top horizontal rule thickness specified in a Define Row procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Row

**Response:** Change the top horizontal rule thickness parameter specified in the Define Row procedure call to a valid value.

**Return Code Constants:** ER-IVROWFR

0104

Out of memory when trying to allocate a field attribute structure.

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Define Field

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-FLDATSMEM

0105

Out of memory when trying to allocate a row attribute structure.

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Define Row

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-ROWATSMEM

0106

Invalid state for a Begin Table procedure call. The state must be page or area.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Table

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-CREATETABLE

0107

Invalid table rotation specified in a Begin Table procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Table

**Response:** Change the table rotation parameter on the Begin Table procedure call to a valid value.

**Return Code Constants:** ER-IVTABLEROT

0108

Invalid right vertical thickness specified in a Begin Table procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Table

**Response:** Change the right vertical rule thickness parameter on the Begin Table procedure call to a valid value.

**Return Code Constants:** ER-IVTBLRGHT

0110

**The row has not been previously defined. The row must be defined before it can be used within a table.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Row

**Response:** Add a Define Row procedure call to your program.

**Return Code Constants:** ER-ROWNOTFND

0111

**The field has not been previously defined. The field must be defined before it can be used.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Field or Define Row

**Response:** Add a Define Field procedure call to your program.

**Return Code Constants:** ER-FIELDNOTFND

0112

**Invalid alignment position specified in a Define Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the alignment position parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-NOTACT-PUTF

0113

**Invalid number of columns on a Define Row procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Row

**Response:** Change the number of columns parameter on the Define Row procedure call to a valid value.

**Return Code Constants:** ER-IVNUMCOLS

0114

**Invalid formatter block state. The state must be active when attempting to set the output characteristics.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Set Output Characteristics

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-OUT

0115

**Invalid output file ID specified in a Set Output Characteristics procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Output Characteristics

**Response:** Change the output file ID parameter on the Set Output Characteristics procedure call to a valid value.

**Return Code Constants:** ER-IVDDNAME

0116

**A NULL depth parameter was specified in an End Table procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Table

**Response:** Ensure that a table depth parameter is specified in the End Table procedure call.

**Return Code Constants:** ER-IVTABLDEP

0117

**Invalid formatter block state. The state must be active when attempting to set the color.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Set Color

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-SETCOL

0118

**Invalid left margin specified in a Define Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the left margin parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVLMAR

0119

**Invalid line spacing specified in a Define Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the line spacing parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVLINESP



0120

Invalid right margin specified in a Define Field procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the right margin parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVRMAR

0121

Invalid format for the field of a table.

**Severity:** 16 (FATAL)

**AFP API Procedures:** Begin Field and Begin Paragraph

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-IVFORMAT

0122

Invalid shading pattern specified in a Put Box procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Box

**Response:** Change the shading pattern parameter on the Put Box procedure call to a valid value.

**Return Code Constants:** ER-IVBXSHADE

0123

Invalid shading intensity specified in a Put Box procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Box

**Response:** Change the shading intensity parameter on the Put Box procedure call to a valid value.

**Return Code Constants:** ER-IVBXSHINT

0124

Invalid state for a Put Box procedure call. The state must be page or area.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Box

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PUTBOX

0125

Invalid page width specified.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Document and Begin Page

**Response:** Change the page width parameter specified in the Begin Document procedure call or the Begin Page procedure call to a valid value.

**Return Code Constants:** ER-IVPGWID

0126

Invalid page depth specified.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Document and Begin Page

**Response:** Change the page depth parameter specified in the Begin Document procedure call or the Begin Page procedure call to a valid value.

**Return Code Constants:** ER-IVPGDEP

0127

Invalid string length specified in a Put Character String procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Character String

**Response:** Change the string length parameter on the Put Character String procedure call to a valid value.

**Return Code Constants:** ER-IVSTRLEN

0129

Invalid state for a Begin Paragraph procedure call. The state must be page or area.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-CREATEPARA

0130

Invalid shading pattern specified in a Begin Paragraph procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the shading pattern parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPRSHADE

0131

**Invalid shading intensity specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the shading intensity parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPRSHINT

0132

**Invalid NULL pointer detected in an internal AFP API procedure.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-INULLPTR

0133

**Invalid format option specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the format option parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARAFORM

0134

**Invalid formatter block state. The state must be active when attempting to use a Put Text procedure call in a paragraph.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Text

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-PUTD

0135

**Invalid formatter state. The state of the block with the formatter must be active when attempting to start a variable depth box.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Create Area

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-SBOX

0136

Invalid width specified.

**Severity:** 8 (ERROR)**AFP API Procedures:** Create Area and Begin Table**Response:** Change the width parameter on the procedure call in error to a valid value.**Return Code Constants:** ER-IVAREAWID

0137

Invalid depth specified.

**Severity:** 8 (ERROR)**AFP API Procedures:** Create Area and Begin Table**Response:** Change the depth parameter on the procedure call in error to a valid value.**Return Code Constants:** ER-IVAREALEN

0138

Invalid X position specified in a Set Position procedure call.

**Severity:** 8 (ERROR)**AFP API Procedures:** Set Position**Response:** Change the X position parameter on the Set Position procedure call to a valid value.**Return Code Constants:** ER-IVXPOS

0139

Invalid Y position specified in a Set Position procedure call.

**Severity:** 8 (ERROR)**AFP API Procedures:** Set Position**Response:** Change the Y position parameter on the Set Position procedure call to a valid value.**Return Code Constants:** ER-IVYPOS

0140

Invalid rule thickness specified in a Set Rule Thickness procedure call.

**Severity:** 8 (ERROR)**AFP API Procedures:** Set Rule Thickness**Response:** Change the rule thickness parameter on the Set Rule Thickness procedure call to a valid value.**Return Code Constants:** ER-IVTHICK

0141

Invalid word spacing specified in a Set Word Spacing procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Word Spacing

**Response:** Change the word spacing parameter on the Set Word Spacing procedure call to a valid value.

**Return Code Constants:** ER-IVSPACE

0142

Invalid X reference coordinate system specified in a Set Position procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Position

**Response:** Change the X reference coordinate system parameter on the Set Position procedure call to a valid value.

**Return Code Constants:** ER-IVXREF

0143

Invalid descriptive name length specified in a Define Font procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Font

**Response:** Change the descriptive name length parameter on the Define Font procedure call to a valid value.

**Return Code Constants:** ER-IVDESCLEN

0144

Invalid first line offset specified in a Begin Paragraph procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the first line offset parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARAOFF

0145

Invalid left margin specified in a Begin Paragraph procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the left margin parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARAMAR

0146

**Invalid line length specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the line length parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARALEN

0147

**Invalid line spacing specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the line spacing parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARALSP

0148

**Invalid right vertical rule offset specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the right vertical rule offset parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARALOF

0149

**Invalid bottom rule offset specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the bottom rule offset parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARABOF

0150

**Invalid state for a Begin Paragraph procedure call. A paragraph already exists.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Ensure that the End Paragraph procedure call is issued prior to the Begin Paragraph procedure call.

**Return Code Constants:** ER-PARAEXISTS

0151

NULL font ID parameter specified in a Define Font procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Font

**Response:** Ensure that a font ID parameter is specified on the Define Font procedure call.

**Return Code Constants:** ER-IVFONT

0152

NULL row ID parameter specified in a Define Row procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Row

**Response:** Ensure that a row ID parameter is specified in the Define Row procedure call.

**Return Code Constants:** ER-IVROWID

0153

Invalid state for a Begin Table procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Table

**Response:** Ensure that the End Table procedure call is issued prior to the Begin Table procedure call.

**Return Code Constants:** ER-TABLEXISTS

0154

Invalid state for a Begin Row procedure call. The state must be table.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Row

**Response:** Ensure that the handle on the AFP API procedure call in error is valid and that the Begin Table procedure call is successfully issued before the Begin Row procedure call.

**Return Code Constants:** ER-BEGINROW

0155

Invalid state for an End Table procedure call. The state must be page or area.

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Table

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-ENDTABLE

0156

**Invalid handle for a Begin Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Field

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-BEGINFLD

0157

**Invalid state for a Begin Field procedure call. The state must be row.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Field

**Response:** Ensure that a Begin Row procedure call is issued before a Begin Field procedure call, or if a previous field exists in the row, ensure that an End Field procedure call is issued prior to this Begin Field procedure call.

**Return Code Constants:** ER-NOTACT-SFLD

0158

**Invalid state for an End Field procedure call. The state must be field.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Field

**Response:** Ensure that a Begin Field procedure call is issued before an End Field procedure call.

**Return Code Constants:** ER-NOTACT-EFLD

0159

**Invalid state for a Begin Row procedure call. The state must be table.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Row

**Response:** Ensure that a Begin Table procedure call is issued before a Begin Row procedure call.

**Return Code Constants:** ER-NOTACT-SROW

0160

**Invalid state for an End Row procedure call. The state must be row.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Row

**Response:** Ensure that a Begin Row procedure call is issued before an End Row procedure call.

**Return Code Constants:** ER-NOTACT-EROW



0161

**Invalid state for an End Table procedure call.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** End Table

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-ETBL

0162

**Invalid state for an End Paragraph procedure call. The state must be paragraph.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** End Paragraph

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-EPAR

0163

**Invalid handle for an End Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Field

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-ENDFLD

0164

**Invalid handle for an End Row procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Row

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-ENDROW

0165

**Invalid handle for an End Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Paragraph

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-ENDPARA

0166

**NULL paragraph depth parameter specified in an End Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Paragraph

**Response:** Ensure that a paragraph depth parameter is specified on the End Paragraph procedure call.

**Return Code Constants:** ER-IVPARADEP

0167

**Invalid box width specified in a Put Box procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Put Box**Response:** Change the box width parameter on the Put Box procedure call to a valid value.**Return Code Constants:** ER-IVBOXWIDTH

0168

**Invalid box depth specified in a Put Box procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Put Box**Response:** Change the box depth parameter on the Put Box procedure call to a valid value.**Return Code Constants:** ER-IVBOXDEPTH

0169

**Invalid rule length specified in a Put Rule procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Put Rule**Response:** Change the rule length parameter on the Put Rule procedure call to a valid value.**Return Code Constants:** ER-IVRULELEN

0170

**Invalid top horizontal rule thickness specified in a Begin Table procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Begin Table**Response:** Change the top horizontal rule thickness parameter on the Begin Table procedure call to a valid value.**Return Code Constants:** ER-IVTBLTOP

0171

**Invalid bottom horizontal rule thickness specified in a Begin Table procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Begin Table**Response:** Change the bottom horizontal rule thickness parameter on the Begin Table procedure call to a valid value.**Return Code Constants:** ER-IVTBLBOT

0172

**Invalid left vertical rule thickness specified in a Begin Table procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Table

**Response:** Change the left vertical rule thickness parameter on the Begin Table procedure call to a valid value.

**Return Code Constants:** ER-IVTBLLFT

0173

**Invalid bottom horizontal rule thickness specified in a Define Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the bottom horizontal rule thickness parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVFLDBOT

0174

**Invalid left vertical rule thickness specified in a Define Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the left vertical rule thickness parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVFLDLFT

0175

**Invalid right vertical rule thickness specified in a Define Field procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the right vertical rule thickness parameter on the Define Field procedure call to a valid value.

**Return Code Constants:** ER-IVFLDRGHT

0176

**Invalid column width specified in a Define Row procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Define Field

**Response:** Change the column width parameter on the Define Row procedure call to a valid value.

**Return Code Constants:** ER-IVCOLWID

0177

**Number of font definitions exceeded.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Define Font

**Response:** Remove unnecessary Define Font procedure calls from your program. You can define a maximum of 255 fonts.

**Return Code Constants:** ER-FONTDEFS

0178

**Invalid state for an End Page procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Page

**Response:** Ensure that the handle on the AFP API procedure call in error is valid. Also, ensure that no table or paragraph is active.

**Return Code Constants:** ER-ENDPAGE

0179

**Invalid state for an End Document procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Document

**Response:** Ensure that the handle on the AFP API procedure call in error is valid. Also, ensure that no page is active.

**Return Code Constants:** ER-ENDDOC

0180

**Invalid state for an Include Page Segment procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Page Segment

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-INCPSEG

0181

**A NULL units parameter was specified on a Query Current Attributes procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that a units of measure parameter is specified in the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVUNITP

0182

**A NULL X coordinate parameter was specified on a Query Current Attributes procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that a X coordinate parameter is specified in the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVXPOSP

0183

**A NULL Y coordinate parameter was specified on a Query Current Attributes procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that a Y coordinate parameter is specified in the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVYPOSP

0184

**A NULL color parameter was specified on a Query Current Attributes procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Current Attributes

**Response:** Ensure that a color parameter is specified in the Query Current Attributes procedure call.

**Return Code Constants:** ER-IVCOLORP

0185

**Invalid values were specified for the inline or reuse parameters.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Page Segment

**Response:** Change the inline or reuse parameters on the Include Page Segment procedure call to a valid value.

**Return Code Constants:** ER-IVINLINE

0186

**NULL handle.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Document, Begin Page, Begin Paragraph, and Begin Table

**Response:** Ensure that an output handle parameter is specified in the procedure call in error.

**Return Code Constants:** ER-IVBLKP

0187

**Invalid intercharacter spacing was specified in a Set Intercharacter Spacing procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Intercharacter Spacing

**Response:** Change the intercharacter spacing parameter on the Set Intercharacter Spacing procedure call to a valid value.

**Return Code Constants:** ER-IVCSPACE

0188

**Invalid formatter block state.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Character String and Put Text

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-SETWSP

0189

**Invalid formatter block state.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Character String and Put Text

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-SETISP

0190

**Invalid state for a Put Text procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Text

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PUTTEXT

0191

**Invalid first line indent specified in a Begin Paragraph procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the first line indent parameter on the Begin Paragraph procedure call to a valid value.

**Return Code Constants:** ER-IVPARAIND

0192

**Invalid state for an Invoke Medium Map procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Invoke Medium Map

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-INVMM

0193

**Invalid formatter block state.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Invoke Medium Map

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-INVMM

0194

**NULL current table depth parameter on an End Row procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Row

**Response:** Ensure that a current table depth parameter is specified in the End Row procedure call.

**Return Code Constants:** ER-IVROWDEP

0195

**Invalid formatter block state.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Set Resource Library Names

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-SLIBS

0196

**Invalid state for a Set Resource Library Names procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Resource Library Names

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-SETLIBS

0197

**NULL page segment library parameter on a Set Resource Library Names procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Resource Library Names

**Response:** Ensure that a page segment library parameter is specified on the Set Resource Library Names procedure call.

**Return Code Constants:** ER-IVPSEGLIB

0198

**NULL object library parameter on a Set Resource Library Names procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Resource Library Names

**Response:** Ensure that an object library parameter is specified on the Set Resource Library Names procedure call.

**Return Code Constants:** ER-IVOBLIB

0199

**NULL font library parameter on a Set Resource Library Names procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Resource Library Names

**Response:** Ensure that a font library parameter is specified on the Set Resource Library Names procedure call.

**Return Code Constants:** ER-IVFONTLIB

0200

**The formatter could not obtain storage with a GETMAIN request.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** All

**Response:** Request a larger region (MVS), a larger virtual machine size (VM), or try running your program in a larger partition (VSE).

**Return Code Constants:** ER-NO-STORAGE

0201

**Error reading resource library. The formatter could not read a required resource library (font, page segment, or object).**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Set Font, Include Page Segment, and Include Object

**Response:** Ensure that the DD name or file name specified on the Set Resource Library Names procedure call is correct and matches your JCL, and that you have read access to the files. In a CICS/ESA environment, ensure that your program does *not* issue the Include Object procedure call. If none of these conditions caused the error, have your system programmer check for problems with the font, page segment, or object library. This return code doesn't apply to the VSE operating system.

**Return Code Constants:** ER-READ-LIB

0202

**A procedure call tried to place text or rules in an area, page, paragraph, or table field, but the space is not wide enough.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Character String, Put Text, Begin Paragraph, and Begin Table

**Response:** If you are placing text with a Put Character String procedure call, either use a smaller font, place fewer characters, or make the area or paragraph wider. If you are placing text in a paragraph, ensure that the paragraph first-line indent plus its length does not exceed the area, paragraph, or page width. If you are beginning a paragraph, ensure that the line length and right rule offset do not exceed the width of the page or



area. If you are building a table, ensure that the table width is not greater than the page or area width. If you place text in a table field with a AFPPTXT (Put Text) procedure call, ensure that the largest text word fits in the table field. To determine the width required for a character string to print, you can issue the AFPQSTR (Query String) procedure call prior to issuing the AFPPTXT (Put Text) procedure call.

**Return Code Constants:** ER-TOO-WIDE

**0203**

**Shade definition not found. A request for shading was made, but the shade definition was not previously defined.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Box, Begin Paragraph, Create Area, and Begin Field

**Response:** This error is caused by an interface problem between the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-NO-SHADE

**0204**

**Invalid request from AFP API to formatter. The AFP API code built a request to the formatter that contains invalid parameters.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** This error is caused by an interface problem between the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-IVREQUEST

**0206**

**AFP API attempted to query the line spacing of a font that is not defined.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Set Position

**Response:** This error is caused by an interface problem between the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-QFONT-NOTFOUND

**0208**

**Missing internal work area address. An internal address that is required for processing is zero.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** This error is caused by an interface problem between the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-NO-FORMATTER-HANDLE

**0209**

**A table row contains more data than will fit in the remaining or maximum table depth or on the page.**

**Severity:** 4 (WARNING)

**AFP API Procedures:** End Row

**Response:** Increase the table depth on the Define Table procedure call, decrease the amount of row contents, position the table further up the page, or start a new page.

**Return Code Constants:** ER-ROW-TOO-DEEP

**0210**

**Error writing to output file. The formatter could not write to the output file specified with the Set Output Characteristics procedure call.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** End Page, End Document, and Terminate AFP API

**Response:** Ensure that you have WRITE authority to the data set. Ensure that the name specified in the Set Output Characteristics procedure call (or the default) matches the DD statement in your JCL (for MVS) or matches the file name specified on the DLBL in your JCL (for VSE).

**Return Code Constants:** ER-WRITE-OUTPUT

**0211**

**An internal request for storage was too large to be processed.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** This error is caused by an interface problem between the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-TOO-BIG

**0212**

**Font index not found or not usable. The formatter requires the font index file in the font library in order to continue processing. The file was not found or could not be read.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API and Set Font

**Response:** Ensure that the font index file is in the font library specified with the Set Resource Library Names procedure call or the default font library if the Set Resource Library Names procedure call was not issued. The index file is a member named AFPINDEX or DCFINDEX. If the member is not found, you must run the Font Library Index Program provided with the AFP API. See your system programmer for assistance.

**Note:** See Appendix A, "Font Library Indexing Program (FLIP)" on page 203 for more information.

**Return Code Constants:** ER-FONTINDEX

## 0213

**The maximum depth specified in the area, page, or table was exceeded; the object will not fit.**

**Severity:** 4 (WARNING)

**AFP API Procedures:** Put Character String, Put Text, Put Box, Put Rule, Include Object, Include Page Segment, and Begin Paragraph

**Response:** Increase the area or table dimensions, position the data further up the page, or start a new page.

**Return Code Constants:** ER-DEPTH-EXCEEDED

## 0214

**The formatter cannot start the requested font. This error occurs when the font is set, not when it is defined.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Font

**Response:** Check the font definition specified with the Define Font by Attributes procedure call. The Font Library Index Program (FLIP) index describes the fonts that are available. See "Font Library Indexing Program" on page 69 for information about the FLIP listing.

If the definition appears correct, ensure that the font is available in the font library. Also, check that the font library name specified in the Set Resource Library Names procedure call (or the default font library if a Set Resource Library Names procedure call was not issued) is correct.

If the font is available, ensure that the font does not contain a fractional point size by checking the font listing produced by the Font Library Index Program. You cannot use a font with a fractional point size, such as 24.9.

Ensure that the code page has a blank character defined at position X'40'.

**Note:** See Appendix A, "Font Library Indexing Program (FLIP)" on page 203 for more information.

**Return Code Constants:** ER-STARTFONT

## 0215

**A procedure call attempted to begin an area, table, row, or field that was not previously defined.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Area, Begin Row, and Begin Field

**Response:** This error is caused by an interface problem between the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-NO-DEFINITION

## 0216

**The resource requested by an Include Page Segment procedure call or an Include Object procedure call was not found.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Page Segment and Include Object

**Response:** Ensure that the resource specified in the procedure call in error exists in the library specified with the Set Resource Library Names procedure call or in the default library, if a Set Resource Library Names procedure call was not issued.

**Return Code Constants:** ER-NO-OBJECT

0217

**The font specified cannot be used by AFP API for one of the following reasons:**

- The font contains invalid AFP objects.
- The font is missing required structured fields.
- The font pattern technology setting does not match previous fonts used in this document.
- The specified character rotation could not be found.
- The default coded font was not found in the font library. The default coded font is X0N2100C.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Font and Begin Document

**Response:** Ensure that you have defined the font correctly, that it is available in your font library, and that the pattern technology setting for all fonts in your document is the same.

Also, ensure that the descriptive name specified with the Define Font by Attributes procedure call matches *exactly* the description in the FLIP listing. Note that the descriptive name is case sensitive; all descriptive names are currently defined in uppercase.

Ensure that the font library contains a coded-font member named X0N2100C even if you are not using the default font. This coded font can point to any valid character set and code page combination that you want, but the member name must be X0N2100C. If you are formatting for a 3800 printer, copy a valid coded font member with zero rotation (that is, a coded font that begins with X1) and rename it to X0N2100C.

**Return Code Constants:** ER-INVFONT

0218

**The code page you requested contains characters that are not in the current character set.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Set Font

**Response:** Select a code page that matches the character set you are using. Refer to *IBM AFP Fonts: Technical Reference for IBM Expanded Core Fonts* for the fonts you are using for more information about selecting code pages.

Ensure that the descriptive name specified with the Define Font by Attributes procedure call matches *exactly* the description in the FLIP listing. Notice that descriptive names are case sensitive; all descriptive names are currently defined in uppercase.

Ensure that the code page is not an ASCII code page; ASCII code pages are not supported.

Ensure that all character sets have unique typeface and attribute combinations. If multiple character sets within a typeface have the same attributes, AFP API selects the first character set found (in the order shown in the FLIP listing).

**Return Code Constants:** ER-CODEPAGE

0219

**A page segment or included object on the page is too wide or too deep and extends beyond the dimensions of the logical page.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Page

**Response:** Print the page in order to determine which object is too large. Either make the object smaller or position it at a different location so it will fit. For page segments and included objects, "white space" in the object may be causing this error.

**Return Code Constants:** ER-OFF-PAGE

0220

**The page may not be printable because of the number or size of the fonts selected.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Page

**Response:** If the page won't print, reduce the number or size of the fonts used on this page.

**Return Code Constants:** ER-FONTSIZE

0221

**The contents of the area extend beyond the dimensions of the logical page. The area is either too wide or too deep. Whether it is the width or depth that is too large depends on the rotation of the area and where you have positioned it relative to the edge of the page.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Area

**Response:** Reduce the size (width or depth) of the area, or move its position (using the Set Position procedure call) so that the area will fit.

**Return Code Constants:** ER-AREA-OFF-PAGE

0222

**A page segment or object contains invalid structured fields.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Page Segment and Include Object

**Response:** Ensure that the segment is valid for a 3820-type device and that it contains valid AFP data stream structured fields. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured fields.

**Return Code Constants:** ER-INVPSEG

0223

**During initialization, the AFP API module could not be loaded into the system.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Initialize AFP API

**Response:** Check with your system programmer to ensure that the AFP API module has been installed. Check your JCL (MVS or VSE) or disk access (VM) to ensure that you have access to the APQTKMOD (MVS, VM, and VSE) and APQIOMOD (MVS only) modules.

**Return Code Constants:** ER-LOADMOD

## 0224

**File exists, and REPLACE was not specified. You requested output to a file that already exists and specified (or took the default) FALS for the REPLACE parameter on the Set Output Characteristics procedure call.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Set Output Characteristics

**Response:** Specify TRU for the REPLACE parameter on the Set Output Characteristics procedure call if you want the file to be replaced, or select a different output file name.

**Return Code Constants:** ER-REPLACE

## 0225

**A Begin Field procedure call was issued, but the field was not part of the Begin Row procedure call currently in effect.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Field

**Response:** Ensure that the field ID specified on the Begin Field procedure call matches the ID for the current Define Field procedure call. Also ensure that the field ID was specified as part of the arrangement on the Define Row procedure call.

**Return Code Constants:** ER-FIELDNDEF

## 0226

**Invalid Begin Group request. You attempted to start a group, but one is already active. Nested groups are not valid; you must end the previous group before beginning a new one.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Group

**Response:** End the previous group with an End Group procedure call before beginning this one.

**Return Code Constants:** ER-NESTGRPS

## 0227

**Invalid End Group request. An End Group procedure call was issued, but a group of that name is not active.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Group

**Response:** Ensure that the group name you specified matches the name specified in the previous Begin Group request.

**Return Code Constants:** ER-NOBEGGRP

0228

**Invalid Put Tag request. A Put Tag procedure call attempted to put a tag in document state (between pages), but no group is active. Tags cannot be placed between pages unless they are preceded by a Begin Group request.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Tag

**Response:** Start a group using the Begin Group procedure call before you place group-level tags with the Put Tag procedure call. Or, to create a page-level tag, start the page so that the Put Tag procedure call comes after the Begin Page procedure call.

**Return Code Constants:** ER-NOACTGRP

0229

**A Begin Field procedure call was issued, but the field was rotated, and no subrow depth was specified. The Define Row procedure call contained AFP-DEFAULT for the SUBROW-DEPTH parameter.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Field

**Response:** Specify a value other than AFP-DEFAULT for the SUBROW-DEPTH parameter of the Define Row procedure call that references the field.

**Return Code Constants:** ER-INVSUBROW

0255

**A logic error in the formatter code occurred.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** All

**Response:** This error is caused by an internal problem in the AFP API internal procedures. Contact your IBM service representative, and report this return code to report the problem.

**Return Code Constants:** ER-FORMATTER-ABEND

0260

**Invalid state for an Include Object procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Object

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-INCOBJ

0262

**Invalid object area width specified in an Include Object procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Include Object

**Response:** Change the object area width parameter specified in the Include Object procedure call to a valid value.

**Return Code Constants:** ER-IVOBJWIDTH

0263

**Invalid object area depth specified in an Include Object procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Include Object**Response:** Change the object area depth parameter specified in the Include Object procedure call to a valid value.**Return Code Constants:** ER-IVOBJDEPTH

0264

**Invalid object area rotation specified in an Include Object procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Include Object**Response:** Change the object area rotation parameter specified in the Include Object procedure call to a valid value.**Return Code Constants:** ER-IVOBJROT

0265

**Invalid object area mapping option specified in an Include Object procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Include Object**Response:** Change the object area mapping option parameter specified in the Include Object procedure call to a valid value.**Return Code Constants:** ER-IVOBJMAP

0266

**Invalid object X offset specified in an Include Object procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Include Object**Response:** Change the object X offset parameter specified in the Include Object procedure call to a valid value.**Return Code Constants:** ER-IVOBJXPOS

0267

**Invalid object Y offset specified in an Include Object procedure call.****Severity:** 8 (ERROR)**AFP API Procedures:** Include Object**Response:** Change the object Y offset parameter specified in the Include Object procedure call to a valid value.**Return Code Constants:** ER-IVOBJYPOS



0268

**Invalid formatter state. The state of the block with the formatter must be active when attempting to include an object.**

**Severity:** 16 (FATAL)

**AFP API Procedures:** Include Object

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-INCOBJ

0269

**Invalid state for a Begin Group procedure call. The state must be document.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Group

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-BEGGRP

0270

**Invalid group name specified in a Begin Group procedure call or an End Group procedure call.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Group and End Group

**Response:** Change the group name parameter on the procedure call in error to a valid value.

**Return Code Constants:** ER-IVGRPNAME

0271

**Invalid state for an End Group procedure call. The state must be document.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** End Group

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-ENDGRP

0272

**Invalid state for a Put Tag procedure call. The state must be document or page.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Tag

**Response:** Ensure that the handle on the AFP API procedure call in error is valid.

**Return Code Constants:** ER-PUTTAG

0273

Invalid attribute name specified in a Put Tag procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Tag

**Response:** Change the attribute name parameter on the Put Tag procedure call to a valid value.

**Return Code Constants:** ER-IVTAGNAME

0274

Invalid attribute value specified in a Put Tag procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Put Tag

**Response:** Change the attribute value parameter on the Put Tag procedure call to a valid value.

**Return Code Constants:** ER-IVTAGVALUE

0275

Invalid formatter state. The state must be active when attempting to begin a group.

**Severity:** 16 (FATAL)

**AFP API Procedures:** Begin Group

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-BGRP

0276

Invalid formatter state. The state must be active when attempting to end a group.

**Severity:** 16 (FATAL)

**AFP API Procedures:** End Group

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-EGRP

0277

Invalid formatter state. The state must be active when attempting to put a tag.

**Severity:** 16 (FATAL)

**AFP API Procedures:** Put Tag

**Response:** Contact your IBM service representative, and report this return code.

**Return Code Constants:** ER-NOTACT-PTAG

0278

Invalid state for a Terminate AFP API procedure call.

**Severity:** 8 (ERROR)

**AFP API Procedures:** Terminate AFP API

**Response:** Ensure that the handle on the Terminate AFP API procedure call in error is valid.

**Return Code Constants:** ER-TERMINATE

0279

**Numeric overflow on a Begin Paragraph procedure call. The specified line length, when added to the current inline position and the left margin, exceeds the maximum valid value.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Begin Paragraph

**Response:** Change the line length parameter on the Begin Paragraph procedure call, so that when it is added to the current inline position and left margin, the result does not exceed the maximum valid value.

**Return Code Constants:** ER-LINELEN-OVERF

0280

**Invalid state for Get Output Buffer procedure call. The state must be document.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Get Output Buffer

**Response:** Verify that the document handle on the AFPGBUF procedure call is valid.

**Return Code Constants:** ER-GETOUT

0281

**Buffered output was not requested on the AFPSOUT procedure call. The AFPGBUF procedure call cannot return output to your program.**

**Severity:** 12 (SEVERE)

**AFP API Procedures:** Get Output Buffer

**Response:** Ensure that the BUFFERED constant is specified in the Output Filename parameter on the AFPSOUT procedure call.

**Return Code Constants:** ER-NOTACT-GBUF

0282

**The record (structured field) is larger than the size of the buffer provided for the record. AFP API truncated the record to fit into the buffer. AFPGBUF returns the actual size of the record in the Buffer Length parameter.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Get Output Buffer

**Response:** Ensure that the maximum output record size specified on the AFPSOUT procedure call is large enough to hold the record.

**Return Code Constants:** ER-IVBUFFER

0284

**Invalid state for a Query Character String Size procedure call. The state must be page, area, paragraph, or field.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Character String Size

**Response:** Ensure that the current handle on the AFPQSTR procedure call is valid.

**Return Code Constants:** ER-QSTR

0285

**Invalid length of the character string specified for a Query Character String Size procedure call. The length of the character string must be greater than zero.**

**Severity:** 8 (ERROR)

**AFP API Procedures:** Query Character String Size

**Response:** Ensure that the length specified in the String Length parameter on the AFPQSTR procedure call is greater than zero.

**Return Code Constants:** ER-QSTR-IVSTRLEN

---

## Appendix C. Shade Patterns and Types

AFP API provides 32 shading intensities in two patterns: **STANDARD** (the default) and **SCREEN**. After specifying the shade pattern, select the shade intensity by specifying one of the percentages shown in Figure 80 or Figure 81.

**Note:** The results may vary on different printers.

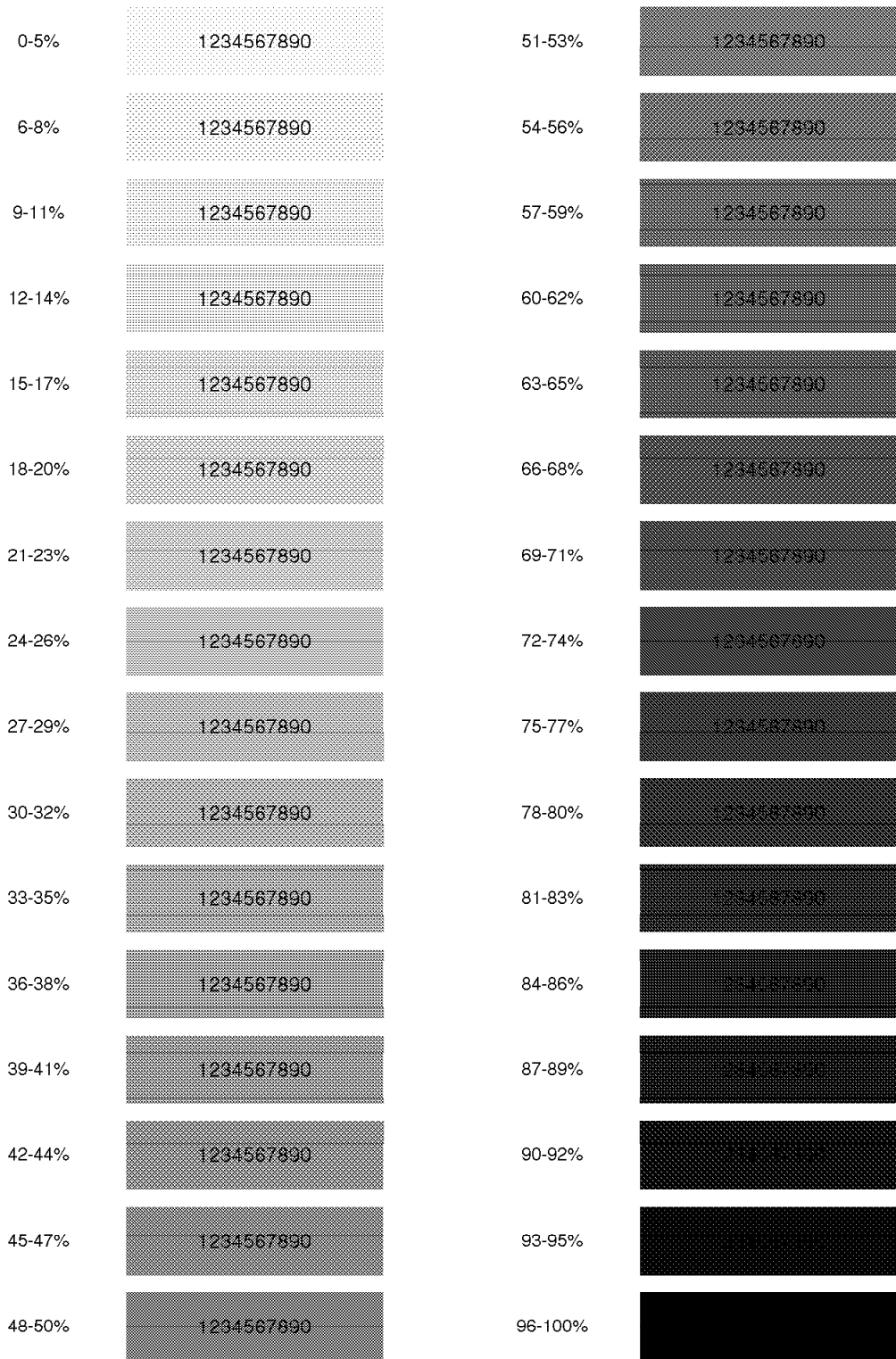


Figure 80. Shade Pattern—STANDARD

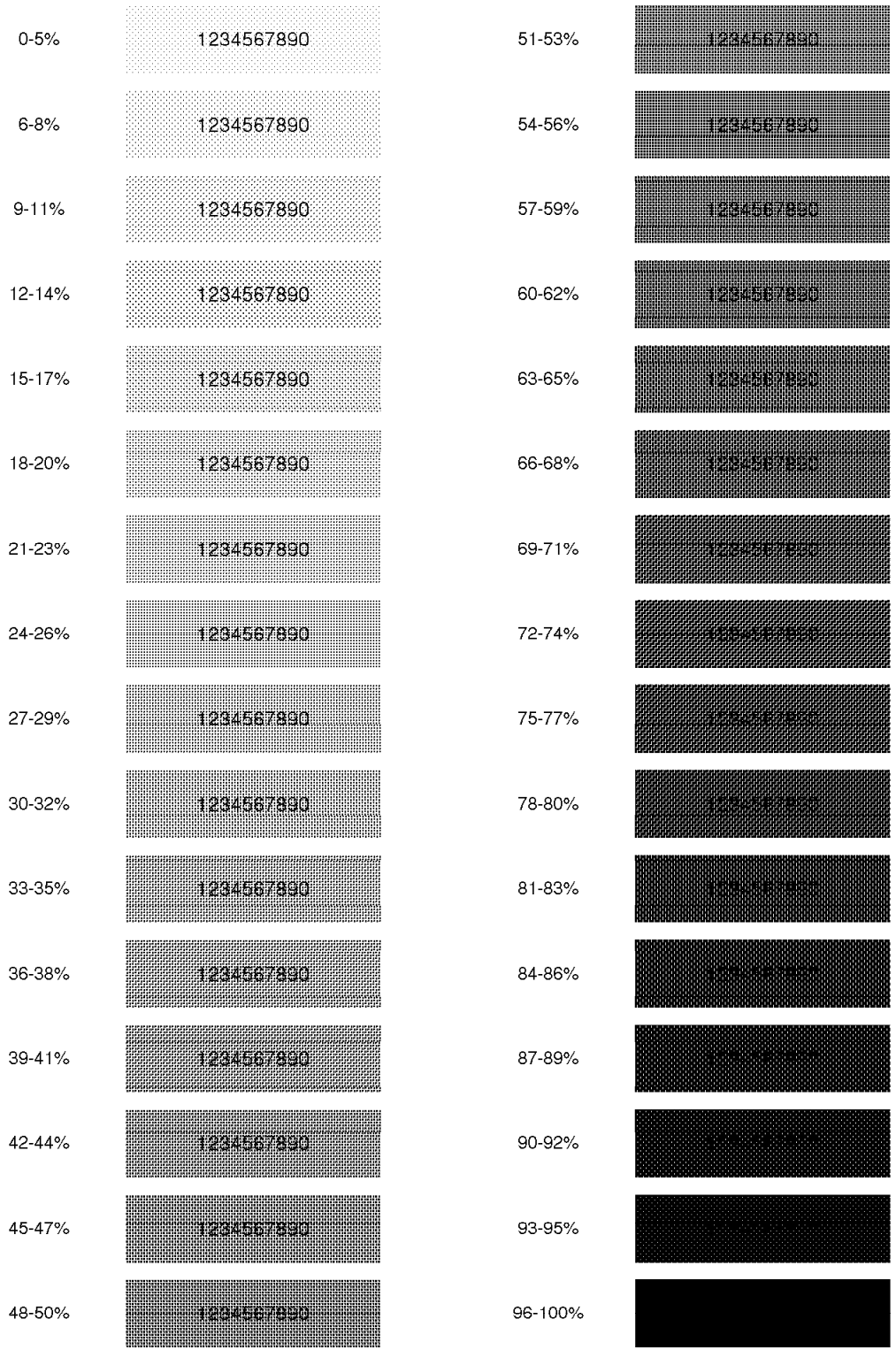


Figure 81. Shade Pattern—SCREEN





---

## Appendix D. Creating an Executable Program under MVS

This appendix contains reference information for building the COBOL and PL/1 load modules and running your COBOL and PL/1 programs under MVS.

---

### MVS JCL for Compiling and Link-Editing a COBOL Application

Figure 82 shows the general format of the JCL for compiling and link-editing a COBOL object with the AFP API code. This JCL is based on the JCL distributed with AFP API in member APQCOCOB. After link-editing, run the job with the JCL shown in Figure 83 on page 267.

```
//APQCOCOB JOB 'acct info','name',MSGLEVEL=(1,1)
/*
/* COMPILE AND LINK-EDIT A COBOL PROGRAM WITH AFP API
/*
//COB2 EXEC PGM=IGYCRCTL,PARM='OBJECT,LIST,LIB,RENT,RES,Q',
// REGION=1024K
/* MODIFY THE FOLLOWING FOR YOUR COBOL LIBRARY
//STEPLIB DD DSNAME=cob2.V131.COB2COMP,DISP=SHR
//SYSPRINT DD SYSOUT=A
/* MODIFY THE FOLLOWING FOR YOUR COBOL PROGRAM
//SYSIN DD DSNAME=PSF.AFPAPI.SAPQSAMI(APQSAMP),DISP=SHR
//SYSLIN DD DSNAME=&&LOADSET,UNIT=SYSDA,DISP=(MOD,PASS),
// SPACE=(TRK,(3,3)),
// DCB=(BLKSIZE=80,LRECL=80,RECFM=FB)
/* MODIFY THE FOLLOWING FOR YOUR COBOL LIBRARY
//SYSLIB DD DSNAME=PSF.AFPAPI.SAPQSAMI,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//LKED EXEC PGM=IEWL,
// PARM='LIST,AMODE=31,RMODE=ANY,RENT,CALL',
// COND=(5,LT,COB2),
// REGION=512K
/* MODIFY THE FOLLOWING FOR YOUR PROGRAM TEXT LIBRARY
//SYSLMOD DD DSNAME=PSF.AFPAPI.SAPQMOD1,
// DISP=(OLD,KEEP)
/* MODIFY THE FOLLOWING FOR YOUR COBOL LIBRARIES
//SYSLIB DD DSNAME=cob2.V131.COB2LIB,DISP=SHR
//APQSTUB DD DSNAME=PSF.AFPAPI.SAPQMOD2,DISP=SHR
//SYSLIN DD DSNAME=&&LOADSET,DISP=(OLD,DELETE)
// DD *
INCLUDE APQSTUB(APQBDOC)
INCLUDE APQSTUB(APQBFLD)
INCLUDE APQSTUB(APQBGRP)
INCLUDE APQSTUB(APQBPAR)
INCLUDE APQSTUB(APQBPAR)
INCLUDE APQSTUB(APQBPAR)
INCLUDE APQSTUB(APQBROW)
INCLUDE APQSTUB(APQBTBL)
INCLUDE APQSTUB(APQCARE)
```

Figure 82 (Part 1 of 2). JCL to Compile and Link-Edit a COBOL Application in an MVS System

```

INCLUDE  APQSTUB(APQDFLD)
INCLUDE  APQSTUB(APQDFNT)
INCLUDE  APQSTUB(APQDROW)
INCLUDE  APQSTUB(APQEARE)
INCLUDE  APQSTUB(APQEDOC)
INCLUDE  APQSTUB(APQEFLD)
INCLUDE  APQSTUB(APQEGRP)
INCLUDE  APQSTUB(APQEND)
INCLUDE  APQSTUB(APQEPAG)
INCLUDE  APQSTUB(APQEPAR)
INCLUDE  APQSTUB(APQEROW)
INCLUDE  APQSTUB(APQETBL)
INCLUDE  APQSTUB(APQGBUF)
INCLUDE  APQSTUB(APQINIT)
INCLUDE  APQSTUB(APQINVM)
INCLUDE  APQSTUB(APQIOBJ)
INCLUDE  APQSTUB(APQIOVL)
INCLUDE  APQSTUB(APQIPSG)
INCLUDE  APQSTUB(APQPARE)
INCLUDE  APQSTUB(APQPBOX)
INCLUDE  APQSTUB(APQPCHS)
INCLUDE  APQSTUB(APQPRUL)
INCLUDE  APQSTUB(APQPRTAG)
INCLUDE  APQSTUB(APQPXTX)
INCLUDE  APQSTUB(APQQATT)
INCLUDE  APQSTUB(APQQPOS)
INCLUDE  APQSTUB(APQQSTR)
INCLUDE  APQSTUB(APQSCLR)
INCLUDE  APQSTUB(APQSFNT)
INCLUDE  APQSTUB(APQSICS)
INCLUDE  APQSTUB(APQSLIB)
INCLUDE  APQSTUB(APQSOUT)
INCLUDE  APQSTUB(APQSPOS)
INCLUDE  APQSTUB(APQSRTH)
INCLUDE  APQSTUB(APQSUNI)
INCLUDE  APQSTUB(APQSWSP)
INCLUDE  APQSTUB(APQXARE)
INCLUDE  APQSTUB(APQXFREE)
INCLUDE  APQSTUB(APQXGET)
INCLUDE  APQSTUB(APQXLOAD)
INCLUDE  APQSTUB(APQXSRVI)
INCLUDE  APQSTUB(APQXSRVN)
INCLUDE  APQSTUB(APQXUNLD)
ENTRY   <Your cobol program name>
NAME    <Your cobol program name>(R)
//SYSUT1 DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=*
/*

```

Figure 82 (Part 2 of 2). JCL to Compile and Link-Edit a COBOL Application in an MVS System

---

## MVS JCL for Running a COBOL Application

Figure 83 shows the general format of the JCL for running the AFP API job from a load module after you link-edit a COBOL object with the AFP API code. This JCL is based on the JCL distributed with AFP API in member APQIVCOB.

```
//APQIVCOB JOB 'acct info','name',MSGLEVEL=(1,1)
/*
/* RUN A COBOL PROGRAM
/*
/* MODIFY THE FOLLOWING FOR YOUR PROGRAM
//GO EXEC PGM=<your cobol program name>
/* MODIFY THE FOLLOWING FOR YOUR COBOL LIBRARY
//STEPLIB DD DSNAME=cob2.V131.COB2LIB,DISP=SHR
// DD DSNAME=PSF.AFPAPI.SAPQMOD1,DISP=SHR
/* MODIFY THE FOLLOWING FOR YOUR PAGE SEGMENT LIBRARY
//PSEGDD DD DSNAME=PSF.AFPAPI.SAPQULIB,DISP=SHR
/* MODIFY THE FOLLOWING FOR YOUR CORE FONT LIBRARY
//FONTDD DD DSNAME=SYS1.FONTLIBB,DISP=SHR
/* MODIFY THE FOLLOWING FOR YOUR OUTPUT DATASET
//APQSAMP DD DSNAME=userid.APQSAMP.LISTAFP,
// DISP=(NEW,CATLG,CATLG),
// UNIT=SYSDA,
// SPACE=(CYL,(2,1),RLSE),
// DCB=(RECFM=VB,LRECL=8205,BLKSIZE=8209)
/* MODIFY THE FOLLOWING FOR YOUR INPUT DATA
//DATAFILE DD DSNAME=PSF.AFPAPI.SAPQSAMI(APQDATA),DISP=SHR
//SYSABOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSDBOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
/*
```

Figure 83. JCL to Execute a COBOL Program in an MVS System

## MVS JCL for Compiling and Link-Editing a PL/1 Application

Figure 84 shows the general format of the JCL for compiling and link-editing a PL/1 object with the AFP API code. This JCL is based on the JCL distributed with AFP API in member APQCOPLI. After link-editing, run the job with the JCL shown in Figure 85 on page 271.

```
//APQCOPLI JOB 'acct no.', 'name', MSGLEVEL=(1,1)
//*
//* MODIFY THE FOLLOWING FOR YOUR PL/1 LIBRARIES
//JOBLIB DD DSN=p1i.V2R3MO.PLICOMP,DISP=SHR
// DD DSN=p1i.V2R3MO.PLITASK,DISP=SHR
// DD DSN=p1i.V2R3MO.PLIBASE,DISP=SHR
// DD DSN=p1i.V2R3MO.SIBMBASE,DISP=SHR
// DD DSN=p1i.V2R3MO.PLILINK,DISP=SHR
// DD DSN=p1i.V2R3MO.SIBMLINK,DISP=SHR
//*
//*****
//* THIS JOB STEP COMPILES PL/1 PROGRAMS *
//*****
//*
//STEP3 EXEC PGM=IELOAA,
// PARM='OBJECT,SOURCE,XREF,INCLUDE',
// REGION=512K
//*
//DDSYS DD DSN=PSF.AFPAPI.SAPQSAM1,DISP=SHR
//*
//SYSLIN DD DSNAME=&&LOADSET(APQPSAMP),
// UNIT=SYSDA,DISP=(MOD,PASS),
// SPACE=(TRK,(30,3,20)),
// DCB=(BLKSIZE=80,LRECL=80,RECFM=FB)
//SYSUT1 DD DSN=&&SYSUT1,
// UNIT=SYSDA,
// SPACE=(1024,(200,20)),
// DCB=BLKSIZE=1024
//*
//SYSIN DD DSN=PSF.AFPAPI.SAPQSAM1(APQPSAMP),DISP=SHR
//*
//SYSPRINT DD SYSOUT=*
//*
//*****
//* THIS STEP LINKEDITS THE PROGRAMS *
//*****
//*
//STEP4 EXEC PGM=IEWL,
// COND=(9,LT,STEP3),
// PARM='XCAL,LIST',
// REGION=512K
//* MODIFY THE FOLLOWING FOR YOUR PL/1 AND C LIBRARIES
//SYSLIB DD DSN=p1i.V2R3MO.PLIBASE,DISP=SHR
// DD DSN=p1i.V2R3MO.SIBMBASE,DISP=SHR
// DD DSN=p1i.V2R3MO.SIBMLINK,DISP=SHR
// DD DSN=p1i.V2R3MO.PLILINK,DISP=SHR
//* MODIFY THE DATA SET NAME AND VOLSER FOR YOUR INSTALLATION
//SYSLMOD DD DSN=userid.API.TEMPOUT(APQPSAMP),
// UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(5,1,1)),VOL=SER=TEMPnn,
// DCB=(BLKSIZE=13000,LRECL=256,RECFM=U)
//*
```

Figure 84 (Part 1 of 3). JCL to Compile and Link-Edit a PL/1 Application in an MVS System

```

//SYSUT1 DD DSN=&&SYSUT1,
//          UNIT=SYSDA,
//          SPACE=(1024,(200,20)),
//          DCB=BLKSIZE=1024
//*
//TXTLIB DD DSN=PSF.AFPAPI.SAPQMOD2,DISP=SHR
//*
//OBJLIB DD DSN=&&LOADSET,DISP=(OLD,PASS)
//*
//SYSPRINT DD SYSOUT=*
//*
//SYSLIN DD *
INCLUDE OBJLIB(APQPSAMP)
INCLUDE TXTLIB(APQBDOC)
INCLUDE TXTLIB(APQBPAQ)
INCLUDE TXTLIB(APQBFLD)
INCLUDE TXTLIB(APQBGRP)
INCLUDE TXTLIB(APQBROW)
INCLUDE TXTLIB(APQBTBL)
INCLUDE TXTLIB(APQBPAR)
INCLUDE TXTLIB(APQCARE)
INCLUDE TXTLIB(APQDFLD)
INCLUDE TXTLIB(APQDFNT)
INCLUDE TXTLIB(APQDROW)
INCLUDE TXTLIB(APQEARE)
INCLUDE TXTLIB(APQEDOC)
INCLUDE TXTLIB(APQEFLD)
INCLUDE TXTLIB(APQEGRP)
INCLUDE TXTLIB(APQEND)
INCLUDE TXTLIB(APQEPAG)
INCLUDE TXTLIB(APQEPAR)
INCLUDE TXTLIB(APQEROW)
INCLUDE TXTLIB(APQETBL)
INCLUDE TXTLIB(APQGBUF)
INCLUDE TXTLIB(APQINIT)
INCLUDE TXTLIB(APQINVM)
INCLUDE TXTLIB(APQIOBJ)
INCLUDE TXTLIB(APQIOVL)
INCLUDE TXTLIB(APQIPSG)
INCLUDE TXTLIB(APQPARE)
INCLUDE TXTLIB(APQPBOX)
INCLUDE TXTLIB(APQPATG)
INCLUDE TXTLIB(APQPXTX)
INCLUDE TXTLIB(APQQATT)
INCLUDE TXTLIB(APQPRUL)
INCLUDE TXTLIB(APQPCHS)
INCLUDE TXTLIB(APQQPOS)
INCLUDE TXTLIB(APQQSTR)
INCLUDE TXTLIB(APQSCLR)
INCLUDE TXTLIB(APQSFNT)
INCLUDE TXTLIB(APQSICS)
INCLUDE TXTLIB(APQSLIB)
INCLUDE TXTLIB(APQSOUT)
INCLUDE TXTLIB(APQSPOS)
INCLUDE TXTLIB(APQSRTH)
INCLUDE TXTLIB(APQSUNI)
INCLUDE TXTLIB(APQSWSP)
INCLUDE TXTLIB(APQTERM)
INCLUDE TXTLIB(APQXARE)
INCLUDE TXTLIB(APQXGET)
INCLUDE TXTLIB(APQXFREE)
INCLUDE TXTLIB(APXLOAD)
INCLUDE TXTLIB(APQXSRVI)
INCLUDE TXTLIB(APQXSRVN)
INCLUDE TXTLIB(APQXUNLD)
INCLUDE SYSLIB(IBMOPAA)

```

Figure 84 (Part 2 of 3). JCL to Compile and Link-Edit a PL/1 Application in an MVS System

```
INCLUDE SYSLIB(IBMBLIIA)
MODE AMODE(31),RMODE(ANY)
NAME <your PL/1 program name> (R)
/*
```

*Figure 84 (Part 3 of 3). JCL to Compile and Link-Edit a PL/1 Application in an MVS System*

## MVS JCL for Running a PL/1 Application

Figure 83 on page 267 shows the general format of the JCL for running the AFP API job from a load module after you link-edit a PL/1 object with the AFP API code. This JCL is based on the JCL distributed with AFP API in member APQIVPLI.

```
//APQIVPLI JOB 'acct no.', 'name', MSGLEVEL=(1,1)
//*
/** MODIFY THE FOLLOWING FOR YOUR PL/1 LIBRARIES
/** ALSO, MODIFY FOR THE TEMPORARY OBJECT DATA SET
/** CREATED FROM JOB APQCOPLI
//JOBLIB DD DSN=pli.V2R3MO.SIBMLINK,DISP=SHR
// DD DSN=pli.V2R3MO.PLILINK,DISP=SHR
// DD DSN=pli.V2R3MO.PLILINK,DISP=SHR
// DD DSN=PSF.AFPAPI.SAPQMOD1,DISP=SHR
// DD DSN=userid.API.TEMPOUT,DISP=SHR
/**
/*****
/** THIS STEP DELETES THE EXISTING DATASETS *
/*****
/**
//STEP1 EXEC PGM=IEFBR14
/**
/**
//FILE1 DD DSN=userid.APQPSAMP.LISTAFP,
// DISP=(MOD,DELETE,DELETE),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1),RLSE)
/**
//FILE2 DD DSN=userid.APQPSAMP.SYSPRINT,
// DISP=(MOD,DELETE,DELETE),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1),RLSE)
/**
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
/**
/*****
/** THIS STEP RUNS A PL/1 PROGRAM THAT USES THE AFP API *
/*****
/**
//STEP2 EXEC PGM=APQPSAMP,REGION=4098K
/**
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
/**
//FONTDD DD DSN=SYS1.FONTLIBB,DISP=SHR
/**
//PSEGDD DD DSN=PSF.AFPAPI.SAPQLIB,DISP=SHR
/**
//APQSAMP DD DSN=userid.APQPSAMP.LISTAFP,
// DISP=(NEW,CATLG,CATLG),
// UNIT=SYSDA,
// SPACE=(CYL,(2,1),RLSE),
// DCB=(RECFM=VB,LRECL=8205,BLKSIZE=8209)
/**
//DATAIN DD DSN=PSF.AFPAPI.SAPQSAM1(APQDATA),DISP=SHR
/**
```

Figure 85 (Part 1 of 2). JCL to Execute a PL/1 Program in an MVS System

```
//SYSPRINT DD DSN=userid.APQPSAMP.SYSPRINT,  
//          DISP=(NEW,CATLG,CATLG),  
//          UNIT=SYSDA,  
//          SPACE=(TRK,(5,5),RLSE),  
//          DCB=(RECFM=FB,LRECL=133,BLKSIZE=13300)  
/*  
//SYSERR DD SYSOUT=*  
//SYSABOUT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SYSIN DD DUMMY  
/*
```

Figure 85 (Part 2 of 2). JCL to Execute a PL/1 Program in an MVS System



## MVS JCL for Compiling and Link-Editing a COBOL Application in a CICS/ESA Environment

Figure 86 shows the general format of the JCL for translating, compiling, and link-editing your COBOL program with the AFP API code in a CICS/ESA environment using the DFHEITVL procedure. This JCL is based on the JCL distributed with AFP API in member APQCOSMB.

```
//DOIT      EXEC PROC=DFHEITVL
//*
/* TRANSLATE, COMPILE, AND LINK-EDIT WITH AFP API
/*
/*  MODIFY THE FOLLOWING FOR YOUR COBOL PROGRAM
//TRN.SYSIN DD DSN=PSF.AFPAPI.SAPQSAMI(APQCISMB),DISP=SHR
/*  MODIFY THE FOLLOWING FOR YOUR COBOL PROGRAM
//COB.SYSLIB DD
//          DD
//          DD DSN=PSF.AFPAPI.SAPQSAMI,DISP=SHR
/*  MODIFY THE FOLLOWING FOR YOUR PROGRAM TEXT LIBRARY
//LKED.SYSLMOD DD DSN=PSF.AFPAPI.SAPQMOD1,DISP=SHR
//LKED.SYSIN DD *
INCLUDE  APQSTUB(APQBDOC)
INCLUDE  APQSTUB(APQBFLD)
INCLUDE  APQSTUB(APQBGRP)
INCLUDE  APQSTUB(APQBPAG)
INCLUDE  APQSTUB(APQBPAR)
INCLUDE  APQSTUB(APQBROW)
INCLUDE  APQSTUB(APQBTBL)
INCLUDE  APQSTUB(APQCARE)
INCLUDE  APQSTUB(APQCEND)
INCLUDE  APQSTUB(APQCFREE)
INCLUDE  APQSTUB(APQCGET)
INCLUDE  APQSTUB(APQCINIT)
INCLUDE  APQSTUB(APQCLOAD)
INCLUDE  APQSTUB(APQCTERM)
INCLUDE  APQSTUB(APQCUNLD)
INCLUDE  APQSTUB(APQDFLD)
INCLUDE  APQSTUB(APQDFNT)
INCLUDE  APQSTUB(APQDROW)
INCLUDE  APQSTUB(APQEARE)
INCLUDE  APQSTUB(APQEDOC)
INCLUDE  APQSTUB(APQEFLD)
INCLUDE  APQSTUB(APQEGRP)
INCLUDE  APQSTUB(APQEPAG)
INCLUDE  APQSTUB(APQEPAR)
INCLUDE  APQSTUB(APQEROW)
INCLUDE  APQSTUB(APQETBL)
INCLUDE  APQSTUB(APQGBUF)
INCLUDE  APQSTUB(APQINVM)
INCLUDE  APQSTUB(APQIOBJ)
INCLUDE  APQSTUB(APQIOVL)
INCLUDE  APQSTUB(APQIPSG)
INCLUDE  APQSTUB(APQPARE)
INCLUDE  APQSTUB(APQPBOX)
INCLUDE  APQSTUB(APQPCHS)
INCLUDE  APQSTUB(APQPRUL)
INCLUDE  APQSTUB(APQPTAG)
INCLUDE  APQSTUB(APQP TXT)
INCLUDE  APQSTUB(APQQATT)
INCLUDE  APQSTUB(APQQPOS)
INCLUDE  APQSTUB(APQQSTR)
```

Figure 86 (Part 1 of 2). JCL to Create an Executable Program in a CICS/ESA Environment

```
INCLUDE APQSTUB(APQSCLR)
INCLUDE APQSTUB(APQSFNT)
INCLUDE APQSTUB(APQSICS)
INCLUDE APQSTUB(APQSLIB)
INCLUDE APQSTUB(APQSOUT)
INCLUDE APQSTUB(APQSPOS)
INCLUDE APQSTUB(APQSRTH)
INCLUDE APQSTUB(APQSUNI)
INCLUDE APQSTUB(APQSWSP)
INCLUDE APQSTUB(APQXARE)
INCLUDE APQSTUB(APQXSRVI)
INCLUDE APQSTUB(APQXSRVN)
ENTRY <Your cobol program name>
NAME <Your cobol program name>(R)
/*
//LKED.APQSTUB DD DSNAME=SYS1.SAPQMOD2,DISP=SHR
```

*Figure 86 (Part 2 of 2). JCL to Create an Executable Program in a CICS/ESA Environment*

## Appendix E. Creating an Executable Program under VM

This appendix contains reference information for building COBOL and PL/1 load modules and running your COBOL and PL/1 programs under VM.

### VM EXEC for Compiling a COBOL Application

Figure 87 shows how to compile a COBOL program in a VM system. This EXEC is distributed with AFP API in file VMCOB. After compiling, use the EXEC shown in Figure 88 on page 276 to create an executable module and run it.

```

/*****/
/* FUNCTION NAME: APQCOCOB */
/* */
/* DESCRIPTION: This exec invokes the COBOL compiler to compile a */
/*              single COBOL source program. */
/* */
/* INPUTS: The filename of the Cobol program to be compiled. */
/* */
/* RETURNS: None */
/* */
/* NOTES: */
/* */
/*****/
  PARSE UPPER ARG compfn
  if (compfn = ' ')
  then do
    say 'Incorrect invocation of APQCOCOB EXEC: ';
    say 'APQCOCOB <fname>';
    return
  end;

/*****/
/* Link to the disk with the Cobol compiler. */
/*****/
  say 'Have you accessed the disk with the COBOL libraries?';
  PULL response
  if ((response = 'YES') & (response = 'Y'))
  then do;
    say 'Access to the COBOL libraries is required.';
    say 'APQCOCOB terminated.';
    exit;
  end;

/*****/
/* Issue a GLOBAL MACLIB for the AFP API copy books. */
/*****/
  GLOBAL MACLIB APQTEXT

/*****/
/* Invoke the Cobol compiler. */
/*****/
  'COBOL2 ' compfn '(LIB RENT RES TERM'

/*****/
/* Clear the GLOBAL MACLIB */
/*****/
  GLOBAL MACLIB

```

Figure 87. VM EXEC to Compile a COBOL Application

## VM EXEC for Link-Editing and Running a COBOL Application

Figure 88 shows how to build an executable module and run it in a VM system. This EXEC is distributed with AFP API in file VMIVCOB.

```

/*****/
/* FUNCTION NAME: APQIVCOB */
/* */
/* DESCRIPTION: This EXEC builds a module from a COBOL text deck, */
/*              loads the AFP API into the nucleus, and starts */
/*              execution of the COBOL program. */
/* */
/* INPUTS: The filename of the COBOL text deck. */
/* */
/* RETURNS: None */
/* */
/* NOTES: */
/* */
/* */
/*****/
PARSE UPPER ARG linkfn
if (linkfn = ' ')
then do
    say 'Incorrect invocation of APQIVCOB EXEC: ';
    say 'APQIVCOB <fname>';
    return
end;
/*****/
/* Link to the disk with the Cobol libraries. */
/*****/
say 'Have you accessed the disk with the COBOL libraries?';
PULL response
if ((response = 'YES') & (response = 'Y'))
then do;
    say 'Access to the COBOL libraries is required.';
    say 'APQIVCOB terminated.';
    exit;
end;

/*****/
/* Issue a GLOBAL LOADLIB for the COBOL load library. */
/*****/
GLOBAL LOADLIB VSC2LOAD

if (RC = 0)
then do;
    say 'APQIVCOB terminated.';
    exit;
end;

/*****/
/* Issue a GLOBAL TXTLIB for the COBOL library, and AFP API */
/* library. */
/*****/
GLOBAL TXTLIB VSC2LTX APQSTUBS

if (RC = 0)
then do;
    say 'APQIVCOB terminated.';
    exit;
end;
```

Figure 88 (Part 1 of 2). VM EXEC to Link-Edit a COBOL Application and Run it

```

/*****
/* Link and load the Cobol program.
/*****
'LOAD' linkfn '(CLEAR MAP AUTO LIBE RLD'

if (RC = 0)
then do;
    say 'LOAD failed for 'linkfn;
    say 'APQIVCOB terminated.';
    exit;
end;
/*****
/* Generate a module.
/*****
'GENMOD' linkfn

if (RC = 0)
then do;
    say 'GENMOD failed for 'linkfn;
    say 'APQIVCOB terminated.';
    exit;
end;
/*****
/* Clear the GLOBAL LOADLIB
/*****
GLOBAL LOADLIB

/*****
/* Issue a FILEDEF for the AFP API trace file (if tracing)
/*****
'FILEDEF STDERR DISK ' linkfn 'STDERR A (RECFM V DSORG PS'

/*****
/* Issue application-specific FILEDEFS (if necessary)
/*****
'FILEDEF DATAFILE DISK APQDATA DATA * (RECFM F LRECL 80 DSORG PS'

/*****
/* Load the AFP API module as a nucleus extension
/*****
'NUCXLOAD APQTKMOD APQTKMOD (SYSTEM'

/*****
/* Start execution of the program
/*****
if (RC = 0) | (RC = 1)
then linkfn
else do;
    say 'APQIVCOB terminated.';
    exit;
end;

```

Figure 88 (Part 2 of 2). VM EXEC to Link-Edit a COBOL Application and Run it

## VM EXEC for Compiling a PL/1 Application

Figure 89 shows how to compile a PL/1 program in a VM system. This EXEC is distributed with AFP API in file VMCOPLI. After compiling, use the EXEC shown in Figure 90 on page 279 to create an executable module and run it.

```

/*****/
/* FUNCTION NAME: APQCOPLI                               */
/*                                                     */
/* DESCRIPTION: This EXEC compiles a single PL/1 source program. */
/*                                                     */
/* INPUTS: The filename of the PL/1 source program.      */
/*                                                     */
/* RETURNS: None                                         */
/*                                                     */
/* NOTES:                                                */
/*                                                     */
/*****/
  PARSE UPPER ARG compfn
  if (compfn = ' ')
  then do
    say 'Incorrect invocation of APQCOPLI EXEC: ';
    say 'APQCOPLI <fname>';
    return
  end;
/*****/
/* Link to the disk with the PL/1 compiler.             */
/*****/
  say 'Have you accessed the disk with the PL/1 libraries?';
  PULL response
  if ((response = 'YES') & (response = 'Y'))
  then do;
    say 'Access to the PL/1 libraries is required.';
    say 'APQCOPLI terminated.';
    exit;
  end;

/*****/
/* Issue a FILEDEF for the AFP API copy book MACLIB.    */
/*****/
  FILEDEF DDSYS DISK APQPLI MACLIB

/*****/
/* Invoke the PL/1 compiler.                             */
/*****/
  'PLIOPT ' compfn '( INC SOURCE'
```

Figure 89. VM EXEC to Compile a PL/1 Application

## VM EXEC for Link-Editing and Running a PL/1 Application

Figure 90 shows how to build an executable module and run it in a VM system. This EXEC is distributed with AFP API in file VMIVPLI.

```

/*****
/* FUNCTION NAME: APQIVPLI
/*
/* DESCRIPTION: This exec builds a module from a PL/1 text deck,
/*             loads the AFP API into the nucleus, and starts
/*             execution of the PL/1 program.
/*
/* INPUTS: The filename of the PL/1 text deck.
/*
/* RETURNS: None
/*
/* NOTES:
/*
/*
/*****
PARSE UPPER ARG linkfn
if (linkfn = ' ')
then do
    say 'Incorrect invocation of APQIVPLI EXEC: ';
    say 'APQIVPLI <fname>';
    return
end;

/*****
/* Link to the disk with the PL/1 libraries.
/*****
say 'Have you accessed the disk with the PL/1 libraries?';
PULL response
if ((response = 'YES') & (response = 'Y'))
then do;
    say 'Access to the PL/1 libraries is required.';
    say 'APQIVPLI terminated.';
    exit;
end;

/*****
/* Issue a GLOBAL TXTLIB for the PL/1 libraries, and AFP API
/* library.
/*****
GLOBAL TXTLIB PLILIB CMSLIB IBMLIB APQSTUBS
if (RC = 0)
then do;
    say 'APQIVPLI terminated.';
    exit;
end;

/*****
/* Link and load the PL/1 program.
/*****
'LOAD' linkfn '(CLEAR MAP AUTO LIBE RLD'
if (RC = 0)
then do;
    say 'APQIVPLI terminated.';
    exit;
end;

```

Figure 90 (Part 1 of 2). VM EXEC to Link-Edit a PL/1 Application and Run it

```

/*****
/* Generate a module.
/*****
'GENMOD' linkfn '(from PLISTART'
  if (RC = 0)
  then do;
    say 'APQIVPLI terminated.';
    exit;
  end;

/*****
/* Issue a FILEDEF for the AFP API trace file (if tracing)
/*****
'FILEDEF STDERR DISK ' linkfn 'STDERR A (RECFM V DSORG PS'

/*****
/* Issue application-specific FILEDEFS (if necessary)
/*****
'FILEDEF DATAIN DISK APQDATA DATA * (RECFM F LRECL 80 DSORG PS'

/*****
/* Load the AFP API module as a nucleus extension
/*****
'NUCXLOAD APQTKMOD APQTKMOD (SYSTEM'

/*****
/* Start execution of the program
/*****
  if (RC = 0) | (RC = 1)
  then linkfn
  else do;
    say 'APQIVPLI terminated.';
    exit;
  end;

```

Figure 90 (Part 2 of 2). VM EXEC to Link-Edit a PL/1 Application and Run it



---

## Appendix F. Creating an Executable Program under VSE

This appendix contains reference information for building a COBOL load module and running your COBOL program under VSE.

---

### VSE JCL for Compiling and Link-Editing a COBOL Application

Figure 91 shows the general format of the JCL for compiling and link-editing a COBOL object with the AFP API code. This JCL is based on the JCL distributed with AFP API in file VSECOCOB. After link-editing, run the job with the JCL shown in Figure 92 on page 283.

```
$$ JOB JNM=APQCOCOB,CLASS=0,DISP=D
// JOB APQCOCOB
*
*   COMPILE A COBOL PROGRAM
*
// DLBL IJSYSPH,'file-id'
// EXTENT SYSPCH,volser,,reltrk,25
  ASSGN SYSPCH,DISK,VOL=volser,SHR
// LIBDEF *,SEARCH=(PRD2.AFP,your cobol libraries)
// OPTION DECK,NOLINK
// EXEC IGYCRCTL,SIZE=IGYCRCTL,PARM=' LIB,RENT,RES,NAME(ALIAS)'
BASIS <your cobol program>
*
*
*   REASSIGN SYSPCH TO THE NORMAL PARTITION ASSIGNMENT
*
  CLOSE SYSPCH,PUNCH
*
*   CATALOG THE OBJECT DECK IN file-id SPECIFIED ABOVE IN IJSYSPCH
*
// DLBL IJSYSIN,'file-id'
// EXTENT SYSIPT,volser
  ASSGN SYSIPT,DISK,VOL=volser,SHR
*
*   ACCESS THE SUBLIB THAT WILL CONTAIN THE OBJECT FILE
*
// EXEC LIBR,PARM=' ACCESS SUBLIB=lib.sublib'
*
*
*   LINK EDIT YOUR COBOL PROGRAM WITH THE AFP API STUBS
*   'file-id' IS THE NAME OF THE FILE THAT CONTAINS THE
*   OBJECT DECK. THIS FILE-ID REFERS TO THE
*   LIB.SUBLIB SPECIFIED ABOVE.
*   'cobol object library' IS THE LIB.SUBLIB DEFINED ABOVE THAT
*   CONTAINS THE OBJECT DECK.
*   'your cobol libraries' IS THE NAME OF THE COBOL COMPILER AND
*   RUNTIME LIBRARIES.
*   'CATALOG=lib.sublib' IS THE LIB.SUBLIB WHERE THE LINK EDITED
*   PHASE WILL BE CATALOGED. THIS CAN BE THE
*   SAME AS THE LIB.SUBLIB THAT CONTAINS THE
*   OBJECT DECK.
*
*   A RETURN CODE OF 4 IS NORMAL FOR THIS JOB. YOU WILL RECEIVE
*   3 WXTERN REFERENCES FOR IGZETUN, IGZEOPT, ILBDMNSO
*
```

Figure 91 (Part 1 of 2). JCL to Compile and Link-Edit a COBOL Application in a VSE System

```

// DLBL filename,'file-id'
// EXTENT ,volser
// LIBDEF *,SEARCH=(cobol object library,PRD2.AFP,your cobol libraries)
// LIBDEF PHASE,CATALOG=lib.sublib
// OPTION CATAL
ACTION MAP
PHASE <your cobol program>,*
INCLUDE <your cobol program>
INCLUDE APQBDOC
INCLUDE APQBFLD
INCLUDE APQBGPR
INCLUDE APQBPAR
INCLUDE APQBROW
INCLUDE APQBTBL
INCLUDE APQCARE
INCLUDE APQDFLD
INCLUDE APQDFNT
INCLUDE APQDROW
INCLUDE APQEARE
INCLUDE APQEDOC
INCLUDE APQEFLD
INCLUDE APQEGRP
INCLUDE APQEND
INCLUDE APQEPAG
INCLUDE APQEPAR
INCLUDE APQEROW
INCLUDE APQETBL
INCLUDE APQGBUF
INCLUDE APQINIT
INCLUDE APQINVM
INCLUDE APQIOBJ
INCLUDE APQIOVL
INCLUDE APQIPSG
INCLUDE APQPARE
INCLUDE APQPBOX
INCLUDE APQPCHS
INCLUDE APQPRUL
INCLUDE APQPTAG
INCLUDE APQPTXT
INCLUDE APQQATT
INCLUDE APQQPOS
INCLUDE APQQSTR
INCLUDE APQSCLR
INCLUDE APQSFNT
INCLUDE APQSICS
INCLUDE APQSLIB
INCLUDE APQSOUT
INCLUDE APQSPOS
INCLUDE APQSRTH
INCLUDE APQSUNI
INCLUDE APQSWSP
INCLUDE APQTERM
INCLUDE APQXARE
INCLUDE APQXFREE
INCLUDE APQXGET
INCLUDE APQXLOAD
INCLUDE APQXSRVI
INCLUDE APQXSRVN
INCLUDE APQXUNLD
ENTRY <your cobol program name>
// EXEC LNKEDT,PARM='AMODE=24,RMODE=24'
*
* REASSIGN SYSIPT TO THE NORMAL PARTITION ASSIGNMENT
*
CLOSE SYSIPT,uuu
/&
$$ EOJ

```

Figure 91 (Part 2 of 2). JCL to Compile and Link-Edit a COBOL Application in a VSE System

## VSE JCL for Running a COBOL Application

Figure 92 shows the general format of the JCL for running the AFP API job from a load module after you link-edit a COBOL object with the AFP API code. This JCL is based on the JCL distributed with AFP API in file VSEIVCOB.

```
$$ JOB JNM=APQIVCOB,CLASS=0,DISP=D
// JOB APQIVCOB
*
* EXECUTE A COBOL PROGRAM
*
*   MODIFY THE FOLLOWING TO
*   ACCESS YOUR COBOL PROGRAM
*
// DLBL filename,'file-id'
// EXTENT ,volser
*
*   MODIFY THE FOLLOWING TO
*   ACCESS THE INPUT DATA FOR YOUR PROGRAM
*
// DLBL DATAFIL,'file-id',0,SD,BLKSIZE=80
// EXTENT SYSxxx,volser,,,reltrk,20
// ASSGN SYSxxx,DISK,VOL=volser,SHR
*
*   MODIFY THE FOLLOWING TO
*   ACCESS THE FILE TO CONTAIN THE AFP API OUTPUT
*
// DLBL APQSAMP,'file-id'
// EXTENT SYSxxx,volser,,,reltrk,20
// ASSGN SYSxxx,DISK,VOL=volser,SHR
*
* 'cobol object library' IS THE COBOL OBJECT LIBRARY DEFINED
*   ABOVE THAT CONTAINS YOUR COBOL OBJECT
* 'your cobol library' IS THE COBOL RUNTIME LIBRARY
*
// LIBDEF *,SEARCH=(cobol object library,PRD2.AFP,your cobol library)
// EXEC <your cobol program name>,SIZE=<your cobol program name>
/*
/&
$$ EOJ
```

Figure 92. JCL to Run a COBOL Application in a VSE System



---

## Appendix G. AFP API Macros Used as Programming Interfaces

---

### General-Use Programming Interfaces

The macros identified in this appendix are provided as programming interfaces for customers by AFP API.

**Warning:** Do not use as programming interfaces any AFP API macros other than those identified in this appendix.

End of General-Use Programming Interfaces

---

### General-Use Programming Interfaces

Following are the macros that are general-use programming interfaces. These macros are documented in *AFP Application Programming Interface: COBOL Language Reference* and *AFP Application Programming Interface: PL/1 Language Reference*.

<b>APQRCS</b>	COBOL return codes
<b>APQCONST</b>	COBOL constants
<b>APQPERF</b>	COBOL performs
<b>APQVARS</b>	COBOL variables
<b>APQTRIM</b>	COBOL trim subprogram
<b>APQSTRL</b>	COBOL string length subprogram
<b>APQPRCS</b>	PL/1 return codes
<b>APQPCON</b>	PL/1 constants
<b>APQPPRF</b>	PL/1 performs
<b>APQPVAR</b>	PL/1 variables



## Appendix H. Related Publications

The following publications may help you understand the licensed programs used with the data streams described in this publication. Also, the text of this publication refers to many of these publications.

### Advanced Function Presentation

Title	Order Number
<i>Guide to Advanced Function Presentation</i> Contains an overview of AFP concepts and products	G544-3876
<i>AFP Application Programming Interface: COBOL Language Reference</i> Contains COBOL language bindings for using the AFP Application Programming Interface	S544-3873
<i>AFP Application Programming Interface: PL/1 Language Reference</i> Contains PL/1 language bindings for using the AFP Application Programming Interface	S544-3874
<i>Advanced Function Presentation: Printer Information</i> Contains details about AFP printers	G544-3290
<i>Page Printer Formatting Aid/370: User's Guide and Reference</i> Contains information about the PPFA/370 product used to create AFP page definitions and form definitions	G544-3181
<i>AFP Workbench for Windows: Using the Viewer Application</i> Contains information about using Workbench with AFP API	G544-3813
<i>AFP Conversion and Indexing Facility: Application Programming Guide</i> Contains information about using AFP Conversion and Indexing Facility	G544-3824
<i>Printing and Publishing Collection Kit</i> Contains the BookManager versions of many AFP publications	SK2T-2921

### Fonts

Title	Order Number
<i>IBM AFP Fonts: Introduction to Typography</i>	G544-3122
<i>IBM AFP Fonts: Technical Reference for Code Pages</i>	S544-3802
<i>IBM AFP Fonts: Technical Reference for IBM Expanded Core Fonts</i>	S544-5228
<i>IBM AFP Fonts: Font Samples</i>	S544-3792
<i>Advanced Function Printing: Host Font Data Stream Reference</i>	S544-3289

### Architecture

Title	Order Number
<i>Mixed Object Document Content Architecture Reference</i> Contains the definition of the Mixed Object Document Content Architecture and its functions and elements.	SC31-6802
<i>Advanced Function Presentation: Programming Guide and Line Data Reference</i> Contains information about processing line and mixed data, page definitions, and the X'5A' prefix on structured fields.	S544-3884
<i>Font Object Content Architecture Reference</i>	S544-3285
<i>Image Object Content Architecture Reference</i>	SC31-6805
<i>Intelligent Printer Data Stream Reference</i>	S544-3417
<i>Graphics Object Content Architecture Reference</i>	SC31-6804
<i>Presentation Text Object Content Architecture Reference</i>	SC31-6803

### PSF/MVS

Title	Order Number
<i>Print Services Facility/MVS: Application Programming Guide</i>	S544-3673
<i>Print Services Facility/MVS: System Programming Guide</i>	S544-3672
<i>Program Directory for Print Services Facility/MVS</i>	None

### PSF/VM

Title	Order Number
<i>Print Services Facility/VM: Application Programming Guide</i>	S544-3677
<i>Print Services Facility/VM: System Programming Guide</i>	S544-3680
<i>Program Directory for Print Services Facility/VM</i>	None

### PSF/VSE

Title	Order Number
<i>Print Services Facility/VSE: Application Programming Guide</i>	S544-3666
<i>Print Services Facility/VSE: System Programming Guide</i>	S544-3665
<i>Program Directory for Print Services Facility/VSE</i>	None



**CICS/ESA Version 4 Release 1**

Title	Order Number
<i>CICS/ESA Customization Guide</i>	SC33-1165
<i>CICS/ESA Resource Definition Guide</i>	SC33-1166
<i>CICS/ESA Operations and Utilities Guide</i>	SC33-1167
<i>CICS/ESA Application Programming Guide</i>	SC33-1169
<i>CICS/ESA Application Programming Reference</i>	SC33-1170

## Related Publications

---

# Glossary

---

## Source Identifiers

This publication includes terms and definitions from:

- The *American National Dictionary for Information Processing Systems*, copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary*, developed by the Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Committee (ISO/IEC JTC1/SC1).

Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/IEC JTC1/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

---

## References

This glossary uses the following cross-references:

<b>Deprecated term for</b>	Indicates that the term should not be used (because it is obsolete, misleading, ambiguous, or jargonistic) and refers to the preferred term. For a deprecated term, the commentary contains only this reference; the deprecated term is not defined.
<b>Synonymous with</b>	Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning. The commentaries of the less desirable or less specific terms refer back to the preferred term with the <i>Synonym for</i> reference words.
<b>Synonym for</b>	Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.
<b>Contrast with</b>	Refers to a term that has an opposite or substantively different meaning.
<b>See</b>	Refers to a multiple-word term in which this term appears.
<b>See also</b>	Refers to related terms that have similar (but not synonymous) meanings.

## A

**abend.** An abnormal end of task before the task's completion because of an error condition.

**absolute positioning.** The establishment of a position within a coordinate system as an offset from the coordinate system origin. Contrast with *relative positioning*.

**addressable position.** A position in a presentation space or on a physical medium that can be identified by a coordinate from the coordinate system of the presentation space or physical medium. See also *picture element*. Synonymous with *position*.

**addressable point.** For page printers, any point in a presentation surface that can be addressed. Synonymous with *picture element*.

**Advanced Function Presentation (AFP).** A set of licensed programs that use the all-points-addressable concept to view and print text and graphics.

**Advanced Function Printing data stream (AFPDS).** The data stream supported by IBM's Advanced Function Presentation products.

**AFP.** (1) Advanced Function Presentation.  
(2) Advanced Function Printing.

**AFPDS.** Advanced Function Printing data stream.

**all points addressable (APA).** (1) The ability to address, reference, and position text, overlays, and images at any defined position or pel on the printable area of the paper. This capability depends on the ability of the hardware to address and to display each picture element. (2) In computer graphics, pertaining to the ability to address and display or not display each picture element (pel) on a display surface. (3) See also *picture element*.

**American National Standard Code for Information Interchange (ASCII).** The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

**APA.** *All-points addressable*.

**application program.** (1) A program that performs a particular data processing task, such as inventory control or payroll. (2) A program that produces the print file.

**area.** An AFP API element that can be defined and used repeatedly in a document or page.

**ASCII.** American National Standard Code for Information Interchange.

**attribute.** A property or characteristic of one or more entities. (T)

## B

**background.** (1) The part of a presentation space that is not occupied with object data. (2) In GOCA, that portion of a drawing primitive that is mixed into the presentation space under the control of the current values of the background mix and background color attributes. (3) In GOCA, that portion of a character cell that does not represent a character. (4) The space of a bar code symbol.

**bar code.** A code representing characters by sets of parallel bars of varying thickness and separation that are read optically by transverse scanning. (I)

**bar code object.** The object containing the structured fields required to present bar code information on a page, a page segment, or an overlay.

**baseline.** The imaginary line on which successive characters are aligned in the inline direction.

**baseline axis.** The axis along which successive lines of text are placed.

**baseline direction.** The direction in which successive lines of text appear on a logical page.

**baseline increment.** The distance between successive sequential baselines.

**bin.** A paper supply on cut-sheet printers. See also *cassette*.

**boilerplate.** (1) A frequently used segment of stored text that can be combined with other text to create a new document. (T) (2) In word processing and desktop publishing, text that is stored for repeated use in various documents; for example, the wording of an edition notice.

## C

**carriage control character.** An optional character in an input data record that specifies a write, space, or skip operation.

**cassette.** In a cut-sheet printer, a movable paper storage enclosure. See also *bin*.

**channel code.** A number from 1 to 12 that identifies a position in the forms control buffer or a page definition. A carriage control character can select a position defined by a particular channel code.

**CHAR.** A data type for architecture syntax, indicating one or more bytes to be interpreted as character information.

**character.** (1) A member of a set of elements that is used for the representation, organization, or control of data. Characters may be letters, digits, punctuation marks, or other symbols. (T) (2) A character is often represented in the form of a spatial arrangement of adjacent or connected strokes or in the form of other physical conditions in data media. (3) A letter, numeral, punctuation mark, or special graphic used for the production of text. (4) A byte of data. (5) See also *graphic character*.

**character baseline.** A reference line within a character box on which the character reference point lies. The character baseline is used to orient and position the initial point of a character.

**character data.** Data in the form of letters and special characters, such as punctuation marks. See also *numeric data*.

**character increment.** The distance between the current print position and the next print position.

**character metrics.** Measurement information that defines individual character values such as height, width, and space. Character metrics may be expressed in specific fixed units, such as pels, or in relative units that are independent of both the resolution and size of the font. Character metrics are often included as part of the general term font metrics. See also *font metrics*.

**character rotation.** The alignment of a character with respect to its character baseline, measured in degrees in a clockwise direction. Examples are 0°, 90°, 180°, and 270°. 0° character rotation exists when a character is in its customary alignment with the baseline.

**character set.** (1) A collection of characters that is composed of some descriptive information and the character shapes themselves. (2) A group of characters used for a specific reason, for example, the set of characters a keyboard contains. (3) Synonym for *font character set*. (4) See also *coded font*.

**character string.** A sequence of characters.

**CICS/ESA.** Customer Information Control System/Enterprise System Architecture.

**CODE.** A data type for architecture syntax, indicating an architected constant to be interpreted as defined by the architecture.

**coded font.** An AFP font that associates a code page and a font character set.

**code page.** Part of an AFP font that associates code points and character identifiers. A code page also identifies undefined code points. See also *coded font*.

**code point.** A 1-byte code representing one of 256 potential characters.

**composed text.** Deprecated term for *presentation text*.

**compression algorithm.** An algorithm used to compress image data. Compression of image data can decrease the volume of data required to represent an image.

**continuous forms.** A series of connected forms that feed continuously through a printing device. The connection between the forms is perforated, enabling the user to tear them apart. Before printing, the forms are folded in a stack arrangement with the folds along the perforations. Contrast with *cut-sheet paper*.

**control character.** A character that starts, changes, or stops any operation that affects recording, processing, transmitting, or interpreting data (such as carriage return, font change, and end of transmission).

**coordinate system.** A Cartesian coordinate system. An example is the image coordinate system that uses the fourth quadrant with positive values for the Y-axis. The origin is the upper left-hand corner of the fourth quadrant. A pair of values in the X and Y axes corresponds to one image point that corresponds to a single image data element.

**coordinates.** A pair of values that specify a position in a coordinate space.

**copies.** See *copy group*.

**copy files.** Files shipped with AFP API to aid in developing user programs. The files contain such items as AFP API variables, return codes, constants for variables, paragraphs that invoke AFP API procedures, and other programs or subprograms.

**copy group.** A subset of a form definition that allows different modifications to be made to multiple copies of the input data. Modifications can include suppression of some data fields from printing as well as the use of different overlays.

**core interchange font.** See *IBM Core Interchange font*.

**current position.** The position identified by the current presentation space coordinates. For example, the coordinate position reached after the execution of a drawing order. See also *current baseline presentation coordinate* and *current inline presentation coordinate*. Contrast with *given position*.

**Customer Information Control System (CICS).** An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It includes facilities for building, using, and maintaining databases.

**cut-sheet paper.** The medium that is cut into uniform-size sheets before it is loaded into the printer. Contrast with *continuous forms*.

## D

**data control block (DCB).** A control block used by access method routines in storing and retrieving data.

**data map.** An internal object in a page definition that specifies fonts, page segments, fixed text, page size, and the placement and orientation of text. Synonymous with *page format*.

**data stream.** A continuous stream of data that has a defined format. An example of a defined format is a structured field.

**DCB.** Data control block.

**DCF.** Document Composition Facility.

**default.** Pertaining to an attribute, value, or option that is assumed when none is explicitly specified. (I)

**direction.** In GOCA, an attribute controlling the direction in which a character string grows relative to the inline direction. Values are: left-to-right, right-to-left, top-to-bottom, and bottom-to-top.

**disabled.** A condition of the printer (physically selected) in which the printer is not available to the host processor. Contrast with *enabled*.

**document.** A file containing an AFP data stream document. An AFP data stream document is bounded by Begin Document and End Document structured fields and can be created using a text formatter such as AFP API.

**document fidelity.** Synonym for *fidelity*.

**Document Composition Facility (DCF).** An IBM licensed program that provides a text formatter called SCRIPT/VS. SCRIPT/VS can process files marked up with a unique set of controls and tags.

**duplex printing.** Printing on both sides of a sheet of paper. Contrast with *simplex printing*. See also *normal duplex printing* and *tumble duplex printing*.

## E

**EBCDIC.** Extended binary-coded decimal interchange code.

**electronic forms.** A collection of constant data that is electronically composed in the host processor and that can be merged with variable data on a page during printing.

**electronic overlay.** A collection of constant data, such as lines, shading, text, boxes, or logos, that is electronically composed in the host processor and stored in a library, and that can be merged with variable data during printing. Contrast with *page segment*. See also *page overlay* and *medium overlay*.

**element.** (1) A bar or space in a bar code character or a bar code symbol. (2) A structured field in a document-content architecture data stream.

**enabled.** (1) Pertaining to a state of the processing unit that allows the occurrence of certain types of interruptions. (2) A condition of the printer (physically selected) in which the printer is available to the host processor for normal work. (3) Contrast with *disabled*.

**exception.** A condition that exists when the printer:

- Detects an invalid or unsupported command, order, control, or parameter value from the host
- Finds a condition requiring host-system notification
- Detects a condition that requires the host system to resend data

**extended binary-coded decimal interchange code (EBCDIC).** A coded character set consisting of eight-bit coded characters.

## F

**FCB.** Forms control buffer.

**fidelity.** The ability to replicate faithfully the appearance of document content on presentation surfaces. Synonymous with *document fidelity*.

**FLIP.** Font Library Indexing Program.

**font.** (1) A family or assortment of characters of a given size and style; for example, 9 point Bodoni Modern. (A) (2) One size and one typeface in a particular type family, including letters, numerals, punctuation marks, special characters, and ligatures.

**font character set.** Part of an AFP font that contains the raster patterns, identifiers, and descriptions of characters. Synonymous with *character set*. See also *coded font*.

**font metrics.** Measurement information that defines individual character values such as height, width, and space, as well as overall font values, such as averages and maximums. Font metrics may be expressed in specific fixed units, such as pels, or in relative units that are independent of both the resolution and size of the font. See also *character metrics*.

**font object.** A resource object that contains the description of a font.

**Font Object Content Architecture (FOCA).** An architected collection of constructs used to describe fonts and to interchange those font descriptions.

**font width.** The average character width expressed in 1440th of an inch. For proportionally spaced fonts, the font width is 1/3 of the point size converted to 1440th of an inch (or the width of the average character escapement box). For fixed pitch fonts, the font width is calculated by dividing 1440 by the pitch. For mixed pitch fonts, the font width is the width of the space character (usually 120). See also *pitch*.

**form.** The paper on which output data is printed by a printer. Synonymous with *physical page, medium, and sheet*.

**format.** (1) The shape, size, and general makeup of a printed document. (2) To prepare a document for printing. (3) The arrangement of text on the page.

**formatter.** A computer program that prepares a source document for printing.

**form definition.** A resource used by PSF that defines the characteristics of the form that includes overlays to be used (if any), text suppression, the position of page data on the form, and the number and modifications of a page.

**forms control buffer (FCB).** A buffer for controlling the vertical format of printed output. The forms control buffer is a line-printer control that is similar to the punched-paper, carriage-control tape used on IBM 1403 printers.

## G

**GDDM.** Graphical Data Display Manager.

**GOCA.** Graphic Object Content Architecture.

**graphic character.** A visual representation of a character, other than a control character, that is normally produced by writing, printing, or displaying. (T)

**Graphical Data Display Manager.** An IBM licensed program containing utilities for creating, saving, editing, and displaying visual data such as page segments, charts, images, vector graphics,

composites (of text, graphics, and images), and scanned data.

**graphics data.** Data containing lines, arcs, markers, and other constructs that describe a picture.

**Graphics Object Content Architecture (GOCA).** An architecture that provides a collection of graphics values and control structures used to interchange and present graphics data.

**group.** An object in a file used to define a named, logical grouping of sequential pages.

## H

**handle.** An ID that identifies the session and the state of the document in which AFP API is operating. AFP API uses handles to keep track of which state in the document it is processing at any given time.

**hanging indent.** When the first line of a paragraph begins at the left margin and all subsequent lines are indented from the left margin by the specified amount.

**hexadecimal.** Pertaining to a numbering system with base of 16; valid numbers use the digits 0 through 9 and characters A through F, where A represents 10 and F represents 15.

## I

**IBM Core Interchange font.** (1) A uniformly spaced, typographic font with specialized characters for different languages. (2) A group of fonts supplied with Print Services Facility Version 2. These fonts include the Courier, Helvetica, and Times New Roman type families. Using the IBM Core Interchange fonts increases the fidelity of documents exchanged between different systems and applications.

**image.** An electronic representation of a picture. An image can also be generated directly by software without reference to an existing picture.

**image data.** A pattern of bits with 0 and 1 values that define the pels in an image. (A 1-bit is a toned pel.)

**image object.** An object that contains image data.

**Image Object Content Architecture (IOCA).** An architected collection of constructs used to interchange and present images.

**image segment.** A set of image data parameters and image data that appear between begin and end segment self-defining fields.

**IM image.** One of the two types of images used in AFP data streams. An IM image is device- and resolution-dependent, bi-level, uses pel-to-pel

mapping for presentation, and cannot be compressed or scaled. Contrast with *IO image*.

**index object file.** An index-information file created by AFP Conversion and Indexing Facility that contains structured fields, which allows indexed information to be retrieved from storage, based on selected attributes.

**inline.** (1) The direction of successive characters in a line of text. Synonymous with *inline direction*. (2) A resource contained within the print file so that a reference to a resource library is not needed.

**inline axis.** The axis along which successive characters in a line are placed.

**inline direction.** (1) The direction in which successive characters appear in a line of text. (2) In GOCA, the direction specified by the character angle attribute.

**inline margin.** The inline coordinate that identifies the initial addressable position for a line of text.

**interface.** A shared boundary. An interface can be a hardware component used to link two devices, or it can be a portion of storage or registers accessed by two or more computer programs. (A)

**IO image.** An image object containing IOCA constructs.

**IOCA.** Image Object Content Architecture.

## J

**JAN.** Japanese Article Numbering. A type of bar code.

## L

**landscape.** Pertaining to a display or hardcopy of a page with greater width than height. Contrast with *portrait*.

**library.** A data file that contains files and control information that allows them to be accessed individually.

**line data.** Data prepared for printing on a line printer, such as a 3800 Model 1.

**line printer.** A device that prints a line of characters as a unit. (I) (A) Contrast with *page printer*.

**local identifier.** An identifier that is mapped by the environment to a named resource.

**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.** A presentation space. One or more object areas or data blocks may be mapped to a logical page. A logical page has specifiable characteristics. Examples of specifiable page characteristics are size, shape, orientation, and offset. The shape of a page is the shape of a rectangle. Orientation and offset are specified relative to a medium coordinate system. Contrast with *physical page*.

**logical page origin.** The point on the logical page that represents  $X_p=0$ ,  $Y_p=0$  in the  $X_p$ ,  $Y_p$  coordinate system.

## M

**magnetic ink character recognition (MICR).** Recognition of characters printed with ink that contains particles of a magnetic material. (I) (A)

**media.** Plural of medium. See also *medium*.

**medium.** A base coordinate space from which all other coordinate spaces either directly or indirectly originate. A medium is mapped to a presentation surface in a device-dependent manner. Synonymous with *form*, *physical page*, and *sheet*.

**medium map.** An internal object in a form definition that controls the modifications to a form, page placement, and overlays. Synonymous with *copy group*.

**media origin.** A page is placed on a medium, or form, relative to the media origin. The media origin is located at the very top-left corner of the form.

**medium overlay.** An electronic overlay that is invoked by the medium map of a form definition for printing at a fixed position on the form. See also *page overlay* and *overlay*.

**MICR.** Magnetic ink character recognition.

**module.** In a bar code, the basic element of width. Actual bar code elements can be a module width or a multiple of a module width. The multiple need not be an integer. The smallest element is also called "unit" or "x dimension" in some bar code specifications.

**Multiple Virtual Storage (MVS).** Multiple Virtual Storage, consisting of MVS/System Product Version 1 and the MVS/370 Data Facility Product operating on a System/370 processor.

**MVS.** Multiple Virtual Storage.



## N

**no operation (NOP).** A construct whose execution causes a product to do nothing other than to proceed to the next instruction to be processed.

**NOP.** No operation.

**normal duplex printing.** Printing on both sides of the paper so that the sheets can be bound on the long edge of the paper. Contrast with *simplex printing*. See also *tumble duplex printing*.

**numeric data.** (1) Data represented by numerals. (I) (A) (2) Data in the form of numerals and special characters. For example, a date represented as 91/01/01. (3) See also *character data*.

## O

**object.** A collection of structured fields. The first structured field provides a begin-object function and the last structured field provides an end-object function. The object may contain one or more other structured fields whose content consists of one or more data elements of a particular data type. An object may be assigned a name, which may be used to reference the object. Examples of objects are text, font, graphics, and image objects.

**object area.** A rectangular area on a logical page into which a data object is mapped. Examples are graphics block and image block.

**object data.** A collection of related data elements that have been bundled together. Examples of data elements are graphic characters, image data elements, and drawing orders.

**offset stacking.** A function that allows the printed output pages to be offset for easy separation of print jobs.

**OGL/370.** Overlay Generation Language/370.

**option.** (1) A specification in a statement that may be used to influence the execution of the statement. (2) A choice offered from a list of possibilities.

**order.** (1) In IOCA, an image data parameter construct. (2) In GOCA, a graphics construct that defines part of a picture or its visual attributes. See also *drawing order*.

**orientation.** The number of degrees an object is rotated relative to a reference; for example, the orientation of an overlay relative to the page origin. Usually applies to blocks of information. Character rotation applies to individual characters. See also *text orientation*.

**origin.** A pel position from which the placement and orientation of text, images, and page segments are specified. For example, pages, overlays, and page segments have origins.

**overlay.** A collection of predefined, constant data such as lines, shading, text, boxes, or logos, that is electronically composed and stored as an AFP resource file than can be merged with variable data on a page while printing or viewing.

**Overlay Generation Language/370 (OGL/370).** An IBM licensed program you can use to design objects for electronic overlays, such as lines, boxes, shadings, and irregular shapes, to create graphics.

## P

**page.** Part of an AFP document bracketed by a pair of Begin Page and End Page structured fields.

**page definition.** A resource used by PSF that defines the rules of transforming line data into composed pages and text controls.

**page format.** Synonym for *data map*.

**page origin.** The point on the logical page that represents  $X_p=0$ ,  $Y_p=0$  in the  $X_p$ ,  $Y_p$  coordinate system. The page origin is relative to the top left of the form. Contrast with *media origin*.

**page overlay.** An electronic overlay that can be invoked for printing and positioned at any point on the page by an Include Page Overlay structured field in the print data. See also *medium overlay* and *overlay*.

**page printer.** A device that prints one page as a unit. (I) (A) Contrast with *line printer*.

**Page Printer Formatting Aid/370 (PPFA/370).** An IBM licensed program that you can use to create and store form definitions and page definitions.

**page segment.** A resource that can contain text and images and can be included on any addressable point on a page or electronic overlay. A page segment assumes the environment of an object in which it is included.

**parameter.** A variable that is given a constant value for a specified application and that may denote the application.

**partial page.** A page that does not contain all the intended data. Partial pages can be printed after an error is sensed.

**pattern.** A symbol used repeatedly to fill an area.

**pel.** Picture element.

**physical page.** Synonymous with *form*, *medium*, and *sheet*. See also *logical page*.

**picture element (pel).** (1) In computer graphics, the smallest element of a display surface that can be independently assigned color and intensity. (T)  
(2) The addressable unit on a page printer. See also *all-points-addressable* and *raster*.

**pitch.** A unit of width of type, based on the number of characters that can be placed in a linear inch. For example, 10-pitch type has ten characters per inch.

**point.** A unit of about 1/72 inch used in measuring type. Contrast with *pitch*.

**point size.** The height of a font in points.

**portrait.** Pertaining to a display or hardcopy of a page with a greater height than width. Contrast with *landscape*.

**position.** A position in a presentation space or on a presentation surface that can be identified by a coordinate from the coordinate system of the presentation space or presentation surface. See also *picture element*. Synonymous with *addressable position*.

**PPFA/370.** Page Printer Formatting Aid/370.

**presentation space.** A conceptual address space with a specified coordinate system and a set of addressable positions. The coordinate system and addressable positions may coincide with those of a presentation surface. Examples of presentation spaces are medium, page, and object area.

**presentation text.** (1) Text data and text control information that dictates the format, placement, and appearance of data to be printed. (2) Print data that has been composed into pages. Text formatting programs such as DCF can produce presentation text consisting entirely of structured fields.  
(3) Synonymous with *composed text*.

**print file.** A file that is created for the purpose of printing data.

**print job.** The data that the user submits to PSF to be printed. A print job can request the printing of multiple data sets.

**print quality.** (1) The measure of printed output against existing standards and in comparison with jobs printed previously. (2) The ability of some page printers to print data at more than one level of print quality, such as "draft" and "near-letter" quality.

**Print Services Facility (PSF).** An IBM licensed program that manages and controls the input data stream and output data stream required by supported

AFP printers. PSF combines print data with other resources and printing controls to produce AFP output.

**PSF.** Print Services Facility.

## R

**raster.** Computer graphics in which a display image is composed of an array of pels arranged in rows and columns.

**relative positioning.** Establishing a position within a coordinate system as an offset from the current position. Contrast with *absolute positioning*.

**resolution.** In computer graphics, a measure of the sharpness of an image, expressed as the number of lines and columns on the display screen or the number of pels per unit of linear measure.

**resource.** A collection of printing instructions consisting entirely of structured fields. Coded fonts, font character sets, code pages, page segments, overlays, form definitions, and page definitions are all resources.

**rotation.** The alignment of a character with respect to its character baseline, measured in degrees in a clockwise direction. Examples are 0°, 90°, 180°, and 270°. 0° character rotation exists when a character is in its customary alignment with the baseline. Synonymous with *character rotation*.

**rule.** A solid or patterned line of any weight, extending horizontally or vertically across a column, row, or page.

## S

**scale.** To enlarge or reduce all or part of a display image.

**scaling.** Making a picture, or part of it, smaller or larger. Scaling is done by multiplying the coordinate values of the picture by a constant amount.

**segment.** A grouping of graphic drawing orders that can appear in a picture.

**semantics.** The part of a construct's description that describes the function of the construct.

**shading.** A darkened area on the displayed page. Usually used to highlight an area containing text. In AFP documents, image data is used to produce shading.

**sheet.** A physical entity on which information is printed. An example of a sheet is one piece of paper. See also *physical sheet*, *medium*, and *form*.

**simplex printing.** Printing on only one side of the paper. Contrast with *duplex printing*.

**single-byte font.** A font having one byte per code point.

**skip.** (1) To ignore one or more instructions in a sequence of instructions. (A) (2) A move of the current print position to another location.

**state.** The portion of the document in which AFP API is operating, for example, document state or page state.

**structured field.** A self-identifying string of bytes and its data or parameters.

**subset.** Within a hierarchy structure, a portion of architecture represented by a particular level.

**symbol.** (1) A visual representation of something by reason of relationship, association, or convention. (2) In GOCA, the sub-picture referenced as a character definition within a font character set and used as a character, marker, or fill pattern. A bitmap may also be referenced as a symbol for use as a fill pattern.

**syntax.** The part of a construct's description that describes the structure of the construct.

## T

**tag.** A type of structured field used for indexing in an AFP document. Tags associate an index attribute - value pair with a specific page or group of pages in a document.

**text.** A graphic representation of information. Text can consist of alphanumeric characters and symbols arranged in paragraphs, tables, columns, and other shapes.

**text control.** Structured-field data that controls the format, placement, and appearance of text.

**text orientation.** A description of the appearance of text as a combination of inline direction and baseline direction. See also *inline direction* and *baseline direction*.

**trace.** A record of the execution of a computer program. It exhibits the sequences in which the instructions were executed. (A)

**transaction.** In CICS, one of more application programs that can be used by a display station operator. A given transaction can be used concurrently from one or more display stations.

**tumble duplex printing.** Duplex printing for sheets that are to be bound on the short edge of the paper regardless of whether the printing is portrait or landscape. Contrast with *normal duplex printing*.

**TYPE.** A table heading for architecture syntax. The entries under this heading indicate the types of data present in a construct. The data type will be one of the following: BITS, CHAR, CODE, SBIN, UBIN.

**type font.** A collection of characters sharing the same type family, typeface, and size.

**typographic fonts.** Fonts in which the distance between characters varies. The distance from one character to another is adjusted to improve the visual flow of text by eliminating excess space.

## V

**value.** A quantity assigned to a constant, a variable, a parameter, or a symbol in a command.

**variable space.** A method used to assign a character increment dimension of varying size to space characters. The space characters are used to distribute white space within a text line. The white space is distributed by expanding or contracting the dimension of the variable space character's increment, dependent upon the amount of white space to be distributed. See also *variable space character*.

**variable space character.** The code point assigned by the data-stream for which the character increment will be varied according to the semantics and pragmatics of the variable space function. This code point is not presented, but its character increment parameter is used to provide spacing.

**virtual machine (VM).** A functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system. Each virtual machine is controlled by an operating system. VM controls concurrent execution of multiple virtual machines on a single system.

**VM.** Virtual machine.

**Virtual Storage Extended (VSE).** An operating system that is an extension of Disk Operating system/Virtual Storage. A VSE system consists of licensed VSE/Advanced Functions support and any programs required to meet the data processing needs of the user. VSE and the hardware it controls form a complete computing system.

**VSE.** Virtual Storage Extended.

## W

**white space.** The portion of a line that is not occupied by characters when the characters of all words that can be placed on a line and spaces between those words are assembled or formatted on a line. When a line is justified, the white space is distributed between the words, characters, or both on the line in some specified manner.

**window.** A predefined part of a graphics presentation space.

## X

**X-axis.** In printing, an axis perpendicular to the direction in which the paper moves through the printer. See also *Y-axis*.

**X-extent.** The width a page or overlay along the X-axis or the size of a page or overlay in the X-direction (horizontal).

**Xm, Ym coordinate system.** The media coordinate system.

**Xp, Yp coordinate system.** The logical page coordinate system.

## Y

**Y-axis.** In printing, an axis parallel with the direction in which the paper moves through the printer. See also *X-axis*.

**Y-extent.** A measurement along the Y-axis.

---

# Index

## A

abends 89  
ACIF (AFP Conversion and Indexing Facility)  
  description 14, 79  
Advanced Function Presentation concepts 3  
AFP API handles 77  
AFP API procedure calls  
  AFBDOC (Begin Document) 100  
  AFBFLD (Begin Field) 104  
  AFBGRP (Begin Group) 106  
  AFBPAG (Begin Page) 108  
  AFBPAR (Begin Paragraph) 112  
  AFBROW (Begin Row) 116  
  AFBTBL (Begin Table) 118  
  AFPCARE (Create Area) 121  
  AFPDFLD (Define Field) 124  
  AFPDFNT (Define Font By Attributes) 128, 129  
  AFPDROW (Define Row) 131  
  AFPEARE (End Area) 134  
  AFPEDOC (End Document) 136  
  AFPEFLD (End Field) 137  
  AFPEGRP (End Group) 138  
  AFPEND (End AFP API) 140  
  AFPEPAG (End Page) 141  
  AFPEPAR (End Paragraph) 143  
  AFPEROW (End Row) 145  
  AFPETBL (End Table) 147  
  AFPGBUF (Get Output Buffer) 149  
  AFPINIT (Initialize AFP API) 151  
  AFPINVM (Invoke Medium Map) 152  
  AFPIOBJ (Include Object) 154  
  AFPIOVL (Include Page Overlay) 158  
  AFPIPSG (Include Page Segment) 160  
  AFPPARE (Put Area) 162  
  AFPPBOX (Put Box) 164  
  AFPPCHS (Put Character String) 166  
  AFPPRUL (Put Rule) 169  
  AFPPTAG (Put Tag) 171  
  AFPPTXT (Put Text) 173  
  AFPQATT (Query Current Attributes) 175  
  AFPQPOS (Query Current Position) 177  
  AFPQSTR (Query Character String Size) 179  
  AFPSCLR (Set Color) 181  
  AFPSFNT (Set Font) 183  
  AFPSICS (Set Intercharacter Spacing) 185  
  AFPSLIB (Set Resource Library Names) 187  
  AFPSOUT (Set Output Characteristics) 190  
  AFPSPOS (Set Position) 193  
  AFPSRTH (Set Rule Thickness) 195  
  AFPSUNI (Set Units) 197  
  AFPSWSP (Set Word Spacing) 199  
  AFPTERM (Terminate AFP API) 201  
  AFPXARE (Destroy Area) 202  
  format of 98

AFP API procedure calls (*continued*)  
  handles 75, 98  
  output, description of 98  
  syntax of 98  
AFP API session  
  AFPEND (End AFP API) procedure call 140  
  AFPGBUF (Get Output Buffer) procedure call 149  
  AFPINIT (Initialize AFP API) procedure call 151  
  ending a session 26  
  performance consideration for 87  
AFP Conversion and Indexing Facility 14  
AFP data stream 4  
AFP Workbench for Windows 14, 79  
alignment  
  AFPPCHS (Put Character String) procedure  
  call 166  
  of fields of a table (AFPDFLD call) 124  
all-points addressability 3  
application programming interface 93  
archiving online documents  
  AFBGRP (Begin Group) procedure call 106  
  AFPEGRP (End Group) procedure call 138  
  AFPPTAG (Put Tag) procedure call 171  
  description 14  
areas  
  AFPCARE (Create Area) procedure call 121  
  AFPEARE (End Area) procedure call 134  
  AFPPARE (Put Area) procedure call 162  
  AFPXARE (Destroy Area) procedure call 202  
  coding tip for using 88  
  description of 47  
  example of 42  
  performance consideration for 87  
  rotation 162  
ASCII code pages 128  
attributes  
  AFPDFNT (Define Font By Attributes) procedure  
  call 128  
  AFPQATT (Query Current Attributes) procedure  
  call 175  
  querying 74  
  setting 73

## B

Begin Document procedure call 100  
Begin Field procedure call 104  
Begin Group procedure call 106  
Begin Page procedure call 108  
Begin Paragraph procedure call 112  
Begin Row procedure call 116  
Begin Table procedure call 118  
beginning a document 23  
boxes  
  AFPPBOX (Put Box) procedure call 164

boxes (*continued*)  
  shading 164  
buffered output  
  abend 0C1 when using 89  
  description of 66  
  obtaining records with AFPGBUF call 149  
  requesting with AFPSOUT call 190

## C

carriage control characters 5  
changing page layout 82  
channel codes 5  
character string  
  AFPCHS (Put Character String) procedure  
    call 166  
  AFPQSTR (Query Character String Size) procedure  
    call 179  
  AFPSICS (Set Intercharacter Spacing) procedure  
    call 185  
    description 28  
CHKSUCC routine, modifying 90  
CICS/ESA environment  
  AFPIOBJ call, not supported in 154  
  AFPSLIB call, ignored in 187  
  AFPSOUT call to define output queue in 190  
  defining font and page segment data sets for 86  
  description of support for 85  
  JCL to link-edit COBOL program with AFP API 273  
  modifying CHKSUCC for 86  
  sample programs for 85  
  transactions for sample programs 85  
  using IOCA and GOCA objects in 85  
  writing output to a temporary storage queue 85  
code page  
  ASCII 128  
  description 68  
  example of using 68  
coding tips 88  
color  
  AFPQATT (Query Current Attributes) procedure  
    call 175  
  AFPSCLR (Set Color) procedure call 181  
    default 73  
columns in tables  
  See tables  
compiling  
  error during 89  
  MVS JCL for 265, 268  
  MVS JCL for, in a CICS/ESA environment 273  
  VM EXEC for 275  
  VSE JCL for 281  
coordinate systems  
  logical page 8  
  physical page (medium) 9  
copy group 84  
Create Area procedure call 121  
current handle description 78

current position  
  AFPQPOS (Query Current Position) procedure  
    call 177  
  after AFPEPAR 143  
  after AFPETBL 147  
  after AFPIOBJ 154  
  after AFPIOVL 158  
  after AFPIPSG 160  
  after AFPPARE 162  
  after AFPPBOX 164  
  after AFPPCHS 166  
  after AFPPRUL 169  
  after AFPPTXT 173  
  when drawing rules 169  
  when drawing rules in an area 122, 169

## D

data object 11  
debugging your program 89  
defaults, list of 73  
Define Field procedure call 124  
Define Font By Attributes procedure call 128  
Define Row procedure call 131  
depth of character string, querying 179  
Destroy Area AFP API procedure call 202  
determining page breaks 82  
document  
  AFPBDOC (Begin Document) procedure call 100  
  AFPEDOC (End Document) procedure call 136  
  description 4  
  elements of 18  
  performance consideration for 87  
  sample 18

## E

End AFP API procedure call 140  
End Area procedure call 134  
End Document procedure call 136  
End Field procedure call 137  
End Group procedure call 138  
End Page procedure call 141  
End Paragraph procedure call 143  
End Row procedure call 145  
End Table procedure call 147  
ending a session 26  
environment, defining for a document 65  
errors, finding in your API program 89  
example  
  See sample document

## F

fields in tables  
  See tables  
FLIP  
  See Font Library Indexing Program

Font Library Indexing Program 69

fonts

- AFPDFNT (Define Font By Attributes) procedure call 128
- AFPSFNT (Set Font) procedure call 183
- coding tips for using 88
- default 73
- defining and using 67
- defining data set in CICS/ESA environment 86
- definition of 12
- determining if a font is on your system 69, 203
- Font Library Indexing Program 69, 203
- performance consideration for 87
- rotation 68, 129
- selecting them for your application 68
- style 129
- using 12

form definition

- definition of 12
- description and use 84
- using 13

## G

Get Output Buffer procedure call 149

getting started 23

GOCA 11

graphics 4

graphics object 11

groups, for indexing

- See indexing for softcopy

## H

handles

- current handle 78
- description 75, 77
- diagram of using handles 78
- format 98

## I

image object 11

- in a CICS/ESA environment 85

Include Object procedure call 63, 154

Include Page Overlay procedure call 158

Include Page Segment procedure call 160

indenting a paragraph 112

indexing for softcopy

- AFPBGRP (Begin Group) procedure call 106
- AFPEGRP (End Group) procedure call 138
- AFPPTAG (Put Tag) procedure call 171
- description 14, 79
- tags 79

Initialize AFP API procedure call 151

initializing an AFP API session

- AFPGBUF (Get Output Buffer) procedure call 149
- AFPINIT (Initialize AFP API) procedure call 151
- program template 24

initializing an AFP API session (*continued*)

- sample code 23
- setting up AFP API 24

intercharacter spacing

- AFPQATT (Query Current Attributes) procedure call 175, 185
- AFPSICS (Set Intercharacter Spacing) procedure call 185
- default 73

invoke medium map

- AFPINVM (Invoke Medium Map) procedure call 152
- description 34
- error using 89

Invoke Medium Map procedure call 84, 152

IOCA 11

## J

JCL

- running AFP API 265, 268
- running AFP API in CICS/ESA environment 273
- running the Font Library Indexing Program 204

## L

library

- See resource libraries

line break

- error when using AFPPTXT 89
- when placing data in a table (AFPPCHS call) 166
- when using AFPPTXT 174

line data 5

line spacing

- in fields of a table (AFPDFLD call) 124
- in paragraphs 112
- querying (AFPQSTR call) 179

lines, drawing them

- See rules

link-editing

- MVS JCL for 265, 268
- VM EXEC for 275
- VSE JCL for 281

logical page

- coordinate system 8
- defining 8, 25
- in Begin Page call 108
- offset 10
- orientation (rotation) 100, 109

logos 4

## M

margins

- for a paragraph (AFBPBAR call) 112
- in the fields of a table (AFPDFLD call) 124

medium coordinate system 9

medium map

- AFPINVM (Invoke Medium Map) procedure call 152

medium map (*continued*)  
description and use 84  
error using 89  
messages, error 89  
MVS  
invoking the Font Library Indexing Program 204  
running AFP API 265, 268  
running AFP API in CICS/ESA environment 273

## N

navigating through a soft-copy document 14, 79

## O

objects  
AFPIOBJ (Include Object) procedure call 154  
description of 11  
graphics 11  
image 11  
in a CICS/ESA environment 85  
using include object (AFPIOBJ) 63  
orientation  
areas 162  
logical page 109  
logical page and physical page 100  
table rotation 119  
output  
AFPSOUT (Set Output Characteristics) procedure  
call 190  
description 66  
obtaining records (AFPGBUF call) 149  
writing in a CICS/ESA environment 85  
writing to an output buffer 66, 149, 190  
overlays  
AFPIOVL (Include Page Overlay) procedure  
call 158  
definition of 12  
description 34  
using 13

## P

page  
AFBPAG (Begin Page) procedure call 108  
AFPEPAG (End Page) procedure call 141  
determining breaks 82  
indexing 14  
layout 82  
orientation (rotation) 100  
page breaks 82  
page definition  
definition of 12  
using 5, 13  
page layout, changing 82  
page segments  
AFIPSG (Include Page Segment) procedure  
call 160  
defining data set in CICS/ESA environment 86

page segments (*continued*)  
definition of 12  
example 33  
library 160  
using 12  
paragraphs  
AFBPAG (Begin Paragraph) procedure call 112  
AFPEPAR (End Paragraph) procedure call 143  
description of 40  
example 36  
parameters (in AFP API procedure calls)  
AFP API handle (AFPEND call) 140  
AFP API handle (AFPINIT call) 151  
AFP API Handle (AFPTERM call) 201  
alignment character (AFPPCHS call) 166  
alignment option (AFPPCHS call) 166  
alignment position (AFPDFLD call) 124  
area depth (AFPEARE call) 134  
area frame (AFPCARE call) 121  
area handle (AFPCARE call) 121  
area handle (AFPEARE call) 134  
area handle (AFPPARE call) 162  
area handle (AFPXARE call) 202  
area rotation (AFPPARE call) 162  
area width (AFPCARE call) 121  
attribute name (AFPPTAG call) 171  
attribute value (AFPPTAG call) 171  
bottom rule offset (AFBPAG call) 112  
bottom thickness (AFPDFLD call) 124  
bottom thickness (AFPDROW call) 131  
box depth (AFPPBOX call) 164  
box width (AFPPBOX call) 164  
buffer (AFPGBUF call) 149  
buffer length (AFPGBUF call) 149  
buffered output (AFPSOUT call) 190  
character spacing (AFPQATT call) 175  
character spacing (AFPSICS call) 185  
character string (AFPPCHS call) 166  
character string (AFPPTXT call) 173  
character string (AFPQSTR call) 179  
code page (AFPDFNT call) 128  
color (AFPQATT call) 175  
color (AFPSCLR call) 181  
column width array (AFPDROW call) 131  
concatenate (AFPPTXT call) 173  
current table depth (AFPEROW call) 145  
descriptive name (AFPDFNT call) 128  
descriptive name length (AFPDFNT call) 128  
direction (AFPPRUL call) 169  
document handle (AFPEDOC call) 136  
document handle (AFPEGRP call) 138  
document handle (AFPGBUF call) 149  
document handle (AFPINVM call) 152  
document page depth (AFPBDOC call) 100  
document page width (AFPBDOC call) 100  
field ID (AFPDFLD call) 124  
field ID (AFPDROW call) 104  
first line indent (AFBPAG call) 112



parameters (in AFP API procedure calls) *(continued)*

first line offset (AFBPAR call) 112  
 font ID (AFPDFNT call) 128  
 font ID (AFPQATT call) 175  
 font ID (AFPSFNT call) 183  
 font library (AFPSLIB call) 187  
 format option (AFBPAR call) 112  
 format option (AFPDFLD call) 124  
 group name (AFBGRP call) 106  
 group name (AFPEGRP call) 138  
 left margin (AFBPAR call) 112  
 left margin (AFPDFLD call) 124  
 left thickness (AFBPTBL call) 118  
 left thickness (AFPDFLD call) 124  
 line length (AFBPAR call) 112  
 line spacing (AFBPAR call) 112  
 line spacing (AFPDFLD call) 124  
 line spacing (AFPQSTR call) 179  
 max table depth (AFBPTBL call) 118  
 maximum area depth (AFPCARE call) 121  
 measured width (AFPQSTR call) 179  
 medium map name (AFPINVM call) 152  
 minimum subrow depth array (AFPDROW call) 131  
 more records (AFPGBUF call) 149  
 number of columns (AFPDROW call) 131  
 number of subrows (AFPDROW call) 131  
 object depth (AFPIOBJ call) 155  
 object library (AFPSLIB call) 187  
 object mapping option (AFPIOBJ call) 155  
 object name (AFPIOBJ call) 155  
 object rotation (AFPIOBJ call) 155  
 object width (AFPIOBJ call) 155  
 object X-offset (AFPIOBJ call) 155  
 object Y-offset (AFPIOBJ call) 155  
 output buffer (AFPGBUF call) 149  
 output file mode (AFPSOUT call) 190  
 output file name (AFPSOUT call) 190  
 output file type (AFPSOUT call) 190  
 output record size (AFPSOUT call) 190  
 overlay name (AFPIOVL call) 158  
 page depth (AFBPAG call) 108  
 page handle (AFBPAG call) 108  
 page handle (AFPEPAG call) 141  
 page inline option (AFPIPSG call) 160  
 page orientation (AFBDOC call) 100  
 page orientation (AFBPAG call) 108  
 page reuse option (AFPIPSG call) 160  
 page segment library (AFPSLIB call) 187  
 page segment name (AFPIPSG call) 160  
 page width (AFBPAG call) 108  
 paragraph depth (AFPEPAR call) 143  
 paragraph frame (AFBPAR call) 112  
 paragraph handle (AFBPAR call) 112  
 paragraph handle (AFPEPAR call) 143  
 point size (AFPDFNT call) 128  
 position option (AFPPCHS call) 166  
 remaining length (AFPPTXT call) 173

parameters (in AFP API procedure calls) *(continued)*

remaining string (AFPPTXT call) 173  
 replace (AFPSOUT call) 190  
 right margin (AFPDFLD call) 124  
 right rule offset (AFBPAR call) 112  
 right thickness (AFBPTBL call) 118  
 right thickness (AFPDFLD call) 124  
 rotation (AFPDFNT call) 128  
 row arrange array (AFPDROW call) 131  
 row ID (AFPBROW call) 116  
 rule length (AFPPRUL call) 169  
 rule thickness (AFPQATT call) 175  
 rule thickness (AFPSRTH call) 195  
 shading intensity (AFBPAR call) 112  
 shading intensity (AFPCARE call) 121  
 shading intensity (AFPDFLD call) 124  
 shading intensity (AFPPBOX call) 164  
 shading pattern (AFBPAR call) 112  
 shading pattern (AFPCARE call) 121  
 shading pattern (AFPDFLD call) 124  
 shading pattern (AFPPBOX call) 164  
 string length (AFPPCHS call) 166  
 string length (AFPQSTR call) 179  
 style (AFPDFNT call) 128  
 table depth (AFPETBL call) 147  
 table handle (AFPBROW call) 116  
 table handle (AFBPTBL call) 118  
 table handle (AFPDROW call) 104  
 table handle (AFPEFLD call) 137  
 table handle (AFPEROW call) 145  
 table handle (AFPETBL call) 147  
 table rotation (AFBPTBL call) 118  
 table width (AFBPTBL call) 118  
 temporary storage queue (AFPSOUT call) 190  
 text orientation (AFPDFLD call) 124  
 top thickness (AFBPTBL call) 118  
 top thickness (AFPDFLD call) 124  
 top thickness (AFPDROW call) 131  
 trace (AFPINIT call) 151  
 underline (AFPPCHS call) 166  
 underline (AFPPTXT call) 173  
 unit of measure (AFPSUNI call) 197  
 units of measure (AFPQATT call) 175  
 vertical format (AFPDFLD call) 124  
 weight (AFPDFNT call) 128  
 width (AFPDFNT call) 128  
 word spacing (AFPQATT call) 175  
 word spacing (AFPSWSP call) 199  
 X coordinate (AFPQATT call) 175  
 X coordinate (AFPQPOS call) 177  
 X coordinate (AFPSPOS call) 193  
 X reference coordinate system (AFPSPOS call) 193  
 Y coordinate (AFPQATT call) 175  
 Y coordinate (AFPQPOS call) 177  
 Y coordinate (AFPSPOS call) 193  
 Y reference coordinate system (AFPSPOS call) 193

- pels 3
- performance considerations 87
- physical page
  - coordinate system 9
  - defining it 25
  - in the AFPINVM (Invoke Medium Map) call 152
- position
  - AFPQPOS (Query Current Position procedure call 177
  - AFPSPOS (Set Position) procedure call 193
- program template 24
- publications, related 287
- Put Area procedure call 162
- put box example 62
- Put Box procedure call 164
- Put Character String procedure call 166
- put rule example 31
- Put Rule procedure call 169
- Put Tag procedure call 171
- Put Text procedure call 173
- putting data on the page 26

## Q

- Query Character String Size procedure call 179
- Query Current Attributes procedure call 175
- Query Current Position procedure call 177
- querying attributes 73

## R

- reference information for procedure calls 93
- related publications 287
- resource libraries
  - AFPSLIB (Set Resource Library Names) procedure call 187
  - defining in a CICS/ESA environment 86
- resources
  - AFPSLIB (Set Resource Library Names) procedure call 187
  - defining in a CICS/ESA environment 86
  - description 12
  - understanding resource objects 11
- return codes and severity codes
  - definitions 209
  - description 65
  - modifying CHKSUCC routine 90
  - use of the severity code 84
- rotation
  - of areas (AFPPARE call) 162
  - of fonts (AFPDFNT call) 128
  - of objects (AFPIOBJ call) 155
  - of tables (AFPBTL call) 118
- rows in tables
  - See tables
- rules
  - AFPPRUL (Put Rule) procedure call 169
  - AFPQATT (Query Current Attributes) procedure call 175

- rules (*continued*)
  - AFPSRTH (Set Rule Thickness) procedure call 195
  - default rule thickness 73
  - drawing vertical rules in an area 122, 169
  - example 31

## S

- sample document
  - areas 42
  - box 62
  - character string 28
  - description 18
  - ending the session 26
  - getting started 23
  - include object 63
  - page segment 33
  - paragraphs 36
  - put rule 31
  - putting data on the page 26
  - template 24
- sample programs for CICS/ESA 85
- screen shading pattern 263
- session
  - See AFP API session
- Set Color procedure call 181
- Set Font procedure call 183
- Set Intercharacter Spacing procedure call 185
- Set Output Characteristics procedure call 190
- Set Position procedure call 193
- Set Resource Library Names procedure call 187
- Set Rule Thickness procedure call 195
- Set Units procedure call 197
- Set Word Spacing procedure call 199
- setting attributes 73
- setting up a document
  - See initializing an AFP API session
- severity codes 84
- shading
  - pattern and intensity 261
  - pattern and intensity for boxes 62, 164
- size of character string, querying 179
- spacing
  - AFPQATT (Query Current Attributes) procedure call 175
  - AFPSICS (Set Intercharacter Spacing) procedure call 185
  - AFPSWSP (Set Word Spacing) procedure call 199
- standard shading pattern 263
- starting a document 23
- states
  - description 75
  - diagram showing hierarchy of states 75
  - table of calls in each state 94
- syntax of procedure calls 98

## T

### tables

- AFPBFLD (Begin Field) procedure call 104
- AFPBROW (Begin Row) procedure call 116
- AFPBTBL (Begin Table) procedure call 118
- AFPDFLD (Define Field) procedure call 124
- AFPDROW (Define Row) procedure call 131
- AFPEFLD (End Field) procedure call 137
- AFPEROW (End Row) procedure call 145
- AFPETBL (End Table) procedure call 147
- definition of 58
- fields, in rows of a table 58
- performance consideration for 87
- rows in a table 58
- setting up 48

### tags, indexing

- AFPPTAG (Put Tag) procedure call 171
- defined 14
- group-level 14

### template, program (for the sample document) 24

### temporary storage queue

- naming (AFPSOUT call) 190
- writing output to in CICS/ESA 85

### Terminate AFP API procedure call 201

### text and AFPPTXT (Put Text) procedure call 173

### tips for coding your program 88

### transactions, CICS/ESA 85

### troubleshooting your program 89

## U

### underline

- AFPPCHS (Put Character String) procedure call 166
- AFPPTXT (Put Text) procedure call 173

### understanding handles 77

### units of measure

- AFPQATT (Query Current Attributes) procedure call 175
- AFPSUNI (Set Units) procedure call 197
- default 73
- used in AFP API output 73

## V

### vertical rules in an area 122, 169

### viewing documents online

- AFPBGRP (Begin Group) procedure call 106
- AFPEGRP (End Group) procedure call 138
- AFPPTAG (Put Tag) procedure call 171
- description 14, 79

### VM

- invoking the Font Library Indexing Program 203
- running AFP API 275

### VSE

- invoking the Font Library Indexing Program 205
- running AFP API 281, 285

## W

### width of character string, querying 179

### word spacing

- AFPQATT (Query Current Attributes) procedure call 175
- AFPQSTR (Query Character String Size) procedure call 179
- AFPSWSP (Set Word Spacing) procedure call 199
- default 73



---

# Readers' Comments — We'd Like to Hear from You

**Advanced Function Presentation  
Application Programming Interface:  
Programming Guide and Reference  
Publication No. S544-3872-02**

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

**Note:** IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

	Yes	No
• Does the publication meet your needs?	_____	_____
• Did you find the information:		
Accurate?	_____	_____
Easy to read and understand?	_____	_____
Easy to retrieve?	_____	_____
Organized for convenient use?	_____	_____
Legible?	_____	_____
Complete?	_____	_____
Well illustrated?	_____	_____
Written for your technical level?	_____	_____
• Do you use this publication:		
As an introduction to the subject?	_____	_____
As a reference manual?	_____	_____
As an instructor in class?	_____	_____
As a student in class?	_____	_____
• What is your occupation?	_____	_____

---

Thank you for your input and cooperation.

**Note:** You may either send your comments by fax to 1-800-524-1519, or mail your comments. If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

**Comments:**

---

Name

---

Address

---

Company or Organization

---

Phone No.



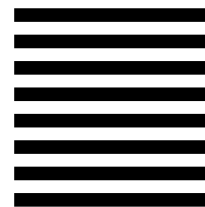
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES



# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Information Development  
The IBM Printing Systems Company  
Department H7FE Building 003G  
P O Box 1900  
BOULDER CO 80301-9817



Fold and Tape

Please do not staple

Fold and Tape





File Number: S370-40



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.



S544-3872-02

