

IBM Print Services Facility



AFP Conversion and Indexing User's Guide



Take Note!

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

First Edition (February 1996)

This edition applies to Advanced Function Presentation Conversion and Indexing Facility (program number 5648-062), which is shipped with Print Services Facility/MVS 2.1.1 (program number 5695-040), Print Services Facility/VM 2.1.1 (program number 5684-141), Print Services Facility/VSE 2.2.1 (program number 5686-040), and IBM Print Services Facility for AIX 2.1.0 (program number 5765-505). This edition applies to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. 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.

IBM welcomes your comments. For your convenience, a form for reader's comments appears at the back of this publication. You can either send your comments by fax to 1-800-524-1519 or mail comments to:

INFORMATION DEVELOPMENT
IBM PRINTING SYSTEMS COMPANY
DEPARTMENT 588 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, 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

Notices	ix
Programming Interface Information	ix
Disclaimer	x
Trademarks	x
About This Publication	xi
Who should use this Publication?	xi
How is this Publication Organized?	xii
What terms are used in this publication?	xiii
Why are some words highlighted, italicized, or underscored?	xiv
Highlighting	xiv
Notational Conventions	xiv
Where Can I Find Related Information?	xvi
How can I order additional copies of this publication?	xvii

Part 1. Information Common to AIX, MVS, VM, and VSE Environments

Chapter 1. Planning Your AFP Conversion and Indexing Facility (ACIF)	
Application	3
What Can I Do with ACIF?	3
Using ACIF for Different Tasks	5
Tasks You Can Do with ACIF	10
What Other IBM Products are Related to ACIF?	17
The Workbench Viewer	18
AFP Application Programming Interface	19
The IBM AFP Toolbox for Multiple Operating Systems	19
Document Composition Facility (DCF)	19
What are the Systems Considerations for ACIF?	20
System Limitations	20
System Prerequisites	20

Part 2. Using ACIF in the AIX Environment

Chapter 2. Using ACIF Parameters in AIX	25
Purpose	25
Syntax	25
Description	26
Flags and Values	29
Examples	50
Implementation Specifics	52
Files	52
NLS Messages	52
Suggested Reading	52
Chapter 3. Example of an ACIF Application in AIX	53
The Input File	55
Specifying ACIF Processing Parameters for ASCII Input Data	55
Using a Parameter File with ASCII Input Data	56
Specifying ACIF Processing Parameters for EBCDIC Input Data	58

Using a Parameter File with EBCDIC Input Data	58
Using the Shell with EBCDIC Literal Values	61
Identifying the Locations of the Resources	61
Determining the Form Definition and the Page Definition	61
Running the ACIF Job	62
ACIF Output	62
Concatenating ACIF Output Files	62
Accessing the Document File from the Workstation	63
Transferring the Document File to the Workstation	63
Mounting the AIX Directory on the Workstation	64
Chapter 4. User Exits and Attributes of the Input Print File in AIX	65
User Programming Exits	65
Input Record Exit	66
Index Record Exit	69
Output Record Exit	71
Resource Exit	73
Non-Zero Return Codes	75
Attributes of the Input Print File	75
Chapter 5. IBM AFP Fonts for ASCII Data	77

Part 3. Using ACIF in the MVS, VM, and VSE Environments 79

Chapter 6. Using ACIF in MVS, VM, and VSE	81
Using ACIF in the MVS Environment	81
Explaining the MVS JCL Statements	81
Using ACIF in the VM Environment	83
Explaining the VM CMS Commands	83
Using ACIF in the VSE Environment	85
Explaining the VSE JCL Statements	85
Chapter 7. Using ACIF Parameters in MVS, VM, and VSE	87
Syntax Rules for MVS, VM, and VSE Parameters	87
Chapter 8. Example: ACIF Application in MVS, VM, or VSE	111
Input File	113
JCL, CMS Commands, and ACIF Processing Parameters	113
MVS JCL to Invoke ACIF	114
VM CMS Commands to Invoke ACIF	115
VSE JCL to Invoke ACIF	116
ACIF Output	120
Concatenating Files	120
MVS JCL	120
VM CMS Commands	120
Chapter 9. User Exits and Attributes of the Input Print File in MVS, VM, and VSE	121
User Programming Exits	121
Input Record Exit	121
Index Record Exit	123
Output Record Exit	125
Resource Exit	126

User Exit Search Order	128
Non-Zero Return Codes	129
Attributes of the Input Print File	129
Chapter 10. ACIF Messages for MVS, VM, and VSE	131
Multiple Message Scenarios	132
General Messages	132

Part 4. Additional Information Common to the AIX, MVS, VM, and VSE

Environments	171
Appendix A. Helpful Hints	173
Working with control statements that contain numbered lines	173
Understanding how ACIF processes fonts	173
Understanding how ACIF processes unbounded box fonts (3800)	174
Placing TLEs in named groups	174
Working with file transfer and AIX	175
Understanding how ANSI and machine carriage controls are used	176
Understanding common methods of transferring files to AIX from other systems	177
Physical media	177
PC file transfer program	177
FTP	178
Other Considerations for Transferring Files to AIX	178
Invoke Medium Map (IMM) Structured Field	178
Indexing Considerations	179
Concatenating the Resource Group to the Document	180
Specifying the IMAGEOUT Parameter	180
Appendix B. Data Stream Information	181
Tag Logical Element (TLE) Structured Field	181
Format of the Resources File	183
Begin Resource Group (BRG) Structured Field	183
Begin Resource (BR) Structured Field	183
End Resource (ER) and End Resource Group (ERG) Structured Fields	184
Appendix C. Format of the Index Object File	185
Group-Level Index Element (IEL) Structured Field	186
Page-Level Index Element (IEL) Structured Field	187
Begin Document Index (BDI) Structured Field	187
Index Element (IEL) Structured Field	188
Tag Logical Element (TLE) Structured Field	189
End Document Index (EDI) Structured Field	189
Appendix D. Format of the Output Document File	191
Page Groups	194
Begin Document (BDT) Structured Field	194
Begin Named Group (BNG) Structured Field	194
Tag Logical Element (TLE) Structured Field	195
Begin Page (BPG) Structured Field	195
End Named Group (ENG), End Document (EDT), and End Page (EPG) Structured Fields	195
Output MO:DCA-P Data Stream	196

Composed Text Control (CTC) Structured Field	196
Map Coded Font (MCF) Format 1 Structured Field	196
Map Coded Font (MCF) Format 2 Structured Field	196
Presentation Text Data Descriptor (PTD) Format 1 Structured Field	196
Inline Resources	196
Page Definitions	196
Glossary	197
Source Identifiers	197
References	197
Index	203

Figures

1.	Advanced Function Presentation publications	xvi
2.	AIX publications	xvi
3.	Font publications	xvi
4.	MO:DCA-P publications	xvii
5.	MVS, VM, and VSE publications	xvii
6.	How ACIF Fits into Advanced Function Presentation	4
7.	Using ACIF to Prepare Files for Viewing	6
8.	Using ACIF to Prepare Files for Distributed Printing	8
9.	Using ACIF to Prepare Files for Archiving and Retrieving	10
10.	AFP Document with Index Tags and the Index Object File	13
11.	Example Bank Statement Input File	14
12.	ACIF Processing Parameters to Index the Bank Statement	15
13.	Workbench Viewer	18
14.	File Extensions for Resources	28
15.	Example of a Customer's Telephone Bill	54
16.	Line-Data Telephone Bill	55
17.	Example of a Parameter File for ASCII Input Data	56
18.	Example of a Parameter File for EBCDIC Input Data	59
19.	Sample Input Record Exit C Language Header	66
20.	Sample Index Record Exit C Language Header	69
21.	Sample Output Record Exit C Language Header	71
22.	Sample Resource Exit C Language Header	73
23.	Sample Print File Attributes C Language Header	75
24.	Font Mapping Table for Use with the chars Parameter	77
25.	Sample MVS JCL to Invoke ACIF	81
26.	Sample VM CMS Commands to Invoke ACIF	83
27.	Sample VSE JCL to Invoke ACIF	85
28.	ACIF Parameters, Tasks, and Operating Systems	88
29.	Example of a Customer's Phone Bill	112
30.	Line-Data Phone Bill	113
31.	Example of a Telephone Bill for an MVS ACIF Application	114
32.	Example of a Telephone Bill for a VM ACIF Application	115
33.	Example of a Telephone Bill for a VSE ACIF Application	116
34.	Library Names	119
35.	Sample Input Record Exit DSECT	122
36.	Sample Index Record Exit DSECT	123
37.	Sample Output Record Exit DSECT	125
38.	Sample Resource Exit DSECT	127
39.	Sample Print File Attributes DSECT	129
40.	Example of Code Containing Group-Level Indexing	192
41.	Example of Code Containing Group- and Page-Level Indexing	193

Notices

References in this publication to products, programs, 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, programs, or services of IBM may be used. Noninfringing equivalents may 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. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, USA.

Programming Interface Information

This publication is intended to help you program and run AFP Conversion and Indexing Facility (ACIF) applications. This publication primarily documents General-Use Programming Interface and Associated Guide Information provided by ACIF.

General-Use programming interfaces allows you to write programs that obtain the services of ACIF.

However, this publication also documents Product-Sensitive Programming Interface and Associated Guidance Information provided by ACIF.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning ACIF. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-Sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Product-Sensitive Programming Interface

Product-Sensitive Programming Interface and Associated Guidance Information...

End of Product-Sensitive Programming Interface

ACIF provides no macros that allow a customer installation to write programs that use the services of ACIF.

Attention: Do not use any ACIF macros as programming interfaces.

Disclaimer

All examples are for illustration only. Any references to existing businesses or individuals is unintentional and does not imply any connection or relationship of any kind between the businesses, individuals, and IBM.

Trademarks

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

- Advanced Function Presentation
- Advanced Function Printing
- AFP
- AIX
- AIX/6000
- AT
- Bar Code Object Content Architecture
- BCOCA
- BookManager
- IBM
- IPDS
- MVS/DFP
- MVS/ESA
- MVS/SP
- Operating System/2
- OS/2
- OS/400
- Personal System/2
- Presentation Manager
- Print Services Facility
- Proprinter
- PSF
- Quietwriter
- RACF
- RISC System/6000
- System/370
- System/390
- S/390
- The IBM Printing Systems Company
- VM/ESA
- VM/XA
- WIN-OS/2

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

- Adobe, ATM-1. and PostScript are trademarks of Adobe Systems, Inc.
- Microsoft, Microsoft Windows, and Windows are trademarks of Microsoft, Inc.
- NFS is a trademark of SUN Microsystems, Inc.
- UNIX is a trademark of UNIX System Laboratories, Inc.

About This Publication

This publication describes Advanced Function Presentation Conversion and Indexing Facility (ACIF), a transform program that allows you to format and print System/370 line data and unformatted ASCII files with IBM Print Services Facility in the following environments:

- AIX
- MVS
- VM
- VSE

ACIF also provides indexing and resource retrieval capabilities that allow you to view, archive, and retrieve document files. This section identifies who should use this publication, how it is organized, the formatting conventions used, and a list of publications related to the product.

Who should use this Publication?

Application programmers should use this publication as a guide and reference when they are developing ACIF applications. ACIF is a batch application development utility enabling creation of documents that can be printed, viewed, distributed, archived, and retrieved with fidelity across systems and platforms. Using ACIF, you can:

- Convert line-format print files to MO:DCA-P documents
- Add indexing tags to documents; create a separate index object file from the indexing tags in a MO:DCA-P document
- Retrieve and package AFP resources needed for printing or viewing a MO:DCA-P document

This publication assumes that you are familiar with Advanced Function Presentation (AFP) concepts as well as the parameters that you specify when printing with Print Services Facility (PSF). If you are not familiar with AFP concepts, refer to *Guide to Advanced Function Presentation*. If you are not familiar with the PSF print parameters, refer to either: *IBM Print Services Facility for AIX: Print Submission* or the PSF application programming guide for your operating system, as listed in "Where Can I Find Related Information?" on page xvi.

The publication also assumes that you are somewhat familiar with the MO:DCA-P architecture and structured fields. You may want to order *Mixed Object Document Content Architecture Reference* and *Advanced Function Presentation: Programming Guide and Line Data Reference* to read about these topics.

Note: This publication provides ACIF messages for the MVS, VM, and VSE environments, which contain instructions for the system programmers responsible for maintaining the operating system and the PSF program running on it. You may need to show these messages to your system programmer for assistance from time to time.

How is this Publication Organized?

This publication contains information pertaining to ACIF support for AIX, MVS, VM, and VSE operating environments. However, because AIX users can invoke the PSF MSG command to view or print AIX messages online, the AIX messages have not been duplicated in this publication.

- Part one contains information common to AIX, MVS, VM, and VSE.
 - Chapter 1, “Planning Your AFP Conversion and Indexing Facility (ACIF) Application” presents an overview of tasks you can do with the ACIF product, describes several related products, and describes system considerations for using ACIF.
- Part two contains information specific to AIX.
 - Chapter 2, “Using ACIF Parameters in AIX” describes the **acif** command, including syntax rules, parameters, and their values.
 - Chapter 3, “Example of an ACIF Application in AIX” describes the steps for developing an ACIF application for AIX.
 - Chapter 4, “User Exits and Attributes of the Input Print File in AIX” describes the exits available in AIX for customizing ACIF.
 - Chapter 5, “IBM AFP Fonts for ASCII Data” provides a list of the fonts for use with unformatted ASCII input data in AIX.
- Part three contains information specific to MVS, VM, and VSE.
 - Chapter 6, “Using ACIF in MVS, VM, and VSE” provides sample code for invoking ACIF in the MVS, VM, and VSE environments.
 - Chapter 7, “Using ACIF Parameters in MVS, VM, and VSE” describes the parameters used for ACIF processing in the MVS, VM, and VSE environments.
 - Chapter 8, “Example: ACIF Application in MVS, VM, or VSE” describes the steps for developing an ACIF application.
 - Chapter 9, “User Exits and Attributes of the Input Print File in MVS, VM, and VSE” describes the exits available for customizing ACIF.
 - Chapter 10, “ACIF Messages for MVS, VM, and VSE” provides the ACIF messages, with suggestions for responding to the errors.
- Part four through the end of this publication contains more information common to AIX, MVS, VM, and VSE.
 - Appendix A, “Helpful Hints” describes some considerations of using ACIF as a front-end preprocessor for viewing, archiving, and retrieving information.
 - Appendix B, “Data Stream Information” describes the structured-field information for indexing.
 - Appendix C, “Format of the Index Object File” describes the file that enables applications to determine the location of a page group or page within the MO:DCA-P print file, based on the indexing tags.
 - Appendix D, “Format of the Output Document File” shows the three separate output files that ACIF can produce.

What terms are used in this publication?

The following are terms this publication uses, plus a description of how those terms apply to your system:

Document In all systems:

A file that contains AFP structured fields in Mixed Object:
Document Content Architecture - Presentation (MO:DCA-P) format

File In AIX and OS/2:

- A collection of related data

In MVS:

- A sequential data set
- A member of a partitioned data set
- A name of a DD card

In VM:

- A CMS file (filename filetype filemode)

In VSE:

- A sequential (SAM) file

Library In AIX:

- A directory in which AFP resources are stored

In OS/2:

- A directory
- A list of files stored on a disk or diskette

In MVS:

- A partitioned data set
- A series of concatenated data sets

In VM:

- A collection of CMS files, generally with the same file type

In VSE:

- A library.sublibrary

Why are some words highlighted, italicized, or underscored?

This publication uses consistent conventions for the following:

- Highlighting
- Notational conventions

Highlighting

This publication uses the following highlighting conventions:

- Bold** Identifies commands, keywords, files, directories, and other items, whose names are predefined by the system or must be entered as is, such as **acif**.
- Italic* Identifies parameters whose actual names or values you supply. Italics also identify the names of publications.
- Monospace Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Notational Conventions

This publication uses the following notational conventions:

- Italics within a command represent variables for which you must supply a value. For example:

cpgid=*code page identifier*

means that you enter **cpgid=** as shown and then replace the variable *code page identifier* with a value that represents any valid code page, which is three-character decimal value (for example, 395) that defines an IBM-registered code page.

- Do not enter the following symbols as part of the command:

Bar	
Braces	{ }
Brackets	[]
Underscore	_

These symbols have the following meanings:

- A vertical bar, |, between values indicates that you can only enter one of the values with the command. For example:

cc={ yes | no }

means that when you enter **cc=**, you can specify either **yes** or **no** as the value, but not both.

Note: In AIX, sometimes the vertical bar, |, acts as a pipe. When the pipe symbol appears between commands, it indicates that the output from the first command becomes the input to the second command. For example:

acif inputdd=myfile | enq -P3825A

means that the output generated by the **acif** command is the input to the AIX **enq** command, which prints the file.

- Braces, { }, around values indicate a required value. For example:

cc={ yes | no }

means that when you enter **cc=**, you must also enter **yes** or **no**.

- Brackets, [], around parameters indicate that they are optional. For example:

[cc=value] [cctype=value]

means that you do not have to enter either **cc=value** or **cctype=value**.

- An underscore, _, indicates the default value, which ACIF uses if you do not specify the parameter with a non-default value. For example:

cc={ yes | no }

means that if the **cc** parameter is not entered, ACIF uses the default value of **yes** for the **cc** parameter.

Where Can I Find Related Information?

The publications listed below might be helpful to you when you are using ACIF. The titles and the order numbers for publications can change from time to time. To verify the current title or order number for a publication, contact your IBM Printing Systems Company representative.

<i>Figure 1. Advanced Function Presentation publications</i>	
Title	Order Number
<i>Guide to Advanced Function Presentation</i> This publication provides an overview of Advanced Function Presentation concepts and products.	G544-3876
<i>Advanced Function Presentation: Programming Guide and Line Data Reference</i> This publication describes the format of S/370 line data and the AFP structured fields that can be used to format that data.	S544-3884
<i>AFP Workbench for Windows and OS/2: Using the Viewer Application</i> This publication describes how to use AFP Workbench.	G544-3813
<i>AFP Application Programming Interface: Programming Guide and Reference</i> This publication describes how to use the AFP Application Programming Interface.	S544-3872
<i>AFP Toolbox for Multiple Operating Systems User's Guide</i>	G544-5292
<i>Advanced Function Printing: Diagnosis Guide</i> This publication helps you determine which of several AFP programs is the source of an error.	LH40-0201
<i>Printing and publishing: Collection Kit</i> This CD-ROM contains copies of many AFP publications in BookManager format.	SK2T-2921

<i>Figure 2. AIX publications</i>	
Title	Order Number
<i>Facts About PSF for AIX</i>	G544-5305
<i>IBM Print Services Facility for AIX: AIX for Print Services Facility Users</i>	G544-3766
<i>IBM Print Services Facility for AIX: Print Administration</i>	S544-3817
<i>IBM Print Services Facility for AIX: Print Submission</i>	S544-3878
<i>IBM Print Services Facility for AIX: PSF Direct Configuration Guide</i>	G544-3766
<i>IBM Print Services Facility for AIX: Guide for Printer and COM Operators</i>	S544-5286
<i>IBM Page Printer Formatting Aid/6000: User's Guide Version 2.1</i>	S544-3918
<i>IBM Print Services Facility for AIX: Licensed Program Specifications</i>	G544-3815
<i>AIX and Related Products Documentation</i>	SC23-2456

<i>Figure 3. Font publications</i>	
Title	Order Number
<i>IBM AFP Fonts: Technical Reference for Code Pages</i> Use this publication as a font reference.	S544-3802
<i>IBM AFP Fonts: Font Summary</i>	G544-3810

<i>Figure 4. MO:DCA-P publications</i>	
Title	Order Number
<i>Mixed Object Document Content Architecture Reference</i> This publication contains information about the MO:DCA-P architecture and about AFP structured fields. You should order this publication to help you work with the structured fields.	SC31-6802

<i>Figure 5. MVS, VM, and VSE publications</i>	
Title	Order Number
<i>Print Services Facility/MVS: System Programming Guide</i> This publication provides information about maintaining, tuning, and modifying PSF/MVS.	S544-3672
<i>MVS/XA: Supervisor Services and Macro Instructions</i>	GC28-1154
<i>MVS/ESA SPL: Application Development Macro</i>	GC28-1857
<i>MVS/ESA Application Development Guide: Authorized Assembler Language Programs</i>	GC28-1645
<i>Print Services Facility/MVS: Application Programming Guide</i> This publication shows how to use JCL to print data with PSF/MVS.	S544-3673
<i>Print Services Facility/VM: System Programming Guide</i> This publication provides information about maintaining, tuning, and modifying PSF/VM.	S544-3680
<i>Print Services Facility/VM: Application Programming Guide</i> This publication shows how to use PSF and CMS commands to print data with PSF/VM.	S544-3677
<i>Print Services Facility/VSE: System Programming Guide</i> This publication provides information about maintaining, tuning, and modifying PSF/VSE.	S544-3665
<i>Print Services Facility/VSE: Application Programming Guide</i> This publication shows how to use JCL to print data with PSF/VSE.	S544-3666
<i>MVS/Extended Architecture Data Administration</i>	GC26-4140

You can use any of the following methods to send comments about the publications:

- Reader's Comment Form in each publication
- Internet id: pennant_pubs@vnet.ibm.com
- IBM Mail Exchange id: IEA USIB4TDB
- Fax number: 1-800-524-1519

How can I order additional copies of this publication?

To order additional printed copies of this publication, use order number *S544-5285*.

Part 1. Information Common to AIX, MVS, VM, and VSE Environments

Chapter 1. Planning Your AFP Conversion and Indexing Facility (ACIF) Application

This chapter provides:

- A description of ACIF and the tasks you can perform with it
- A brief description of related products you can use with ACIF
- A list of system prerequisites for using ACIF

What Can I Do with ACIF?

AFP Conversion and Indexing Facility (ACIF) is a batch application development utility you can use to:

- Convert line data or mixed data into MO:DCA-P data, which is an architected, device-independent data stream used for interchanging documents between different platforms
- Index a document to enhance your ability to view, archive, or retrieve individual pages or groups of pages from large documents; create a separate *index object file* from the indexing tags
- Retrieve and package AFP resources needed for printing or viewing a document and place them in a separate file, so that you can print and view the exact document, possibly years after its creation

Figure 6 on page 4 shows a high-level overview of how the ACIF application development utility fits into an installation's AFP process for creating, indexing, viewing, and printing documents.

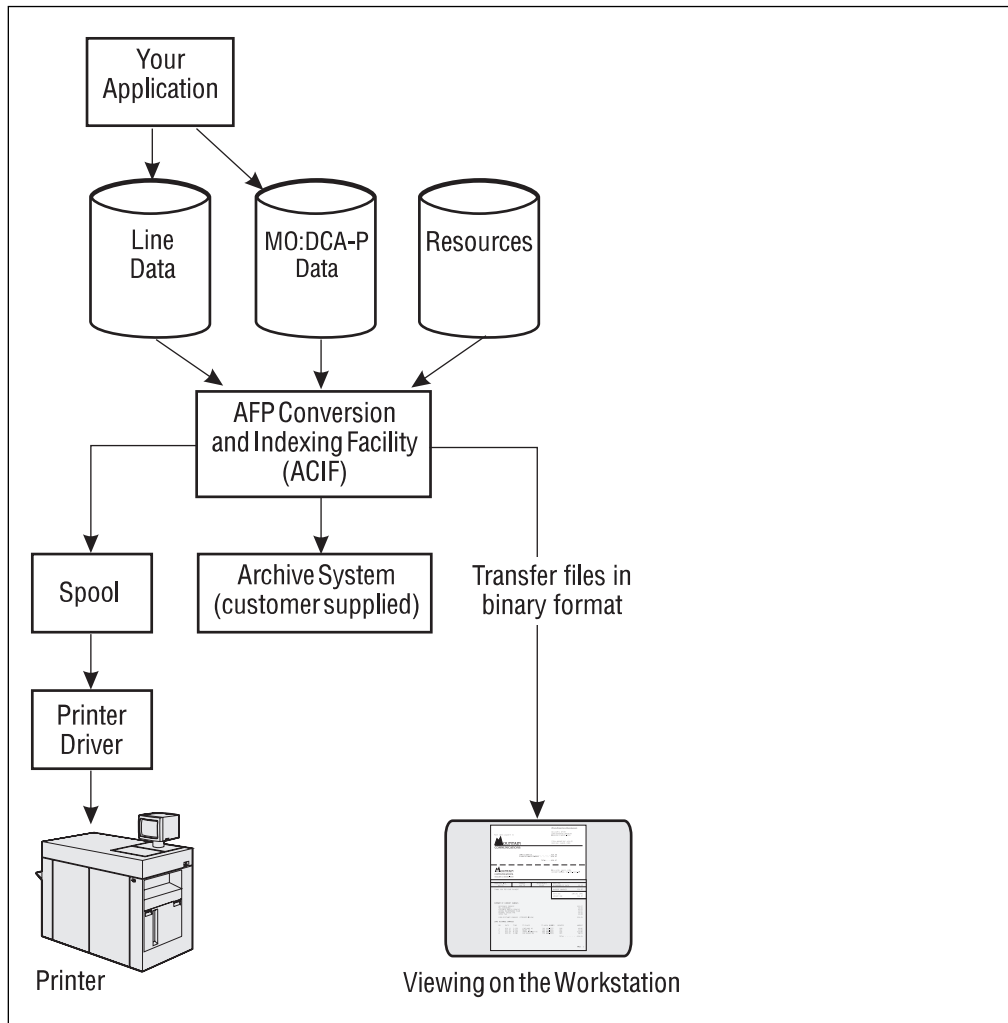


Figure 6. How ACIF Fits into Advanced Function Presentation

The figure shows the resources and the text data, which can be provided and used by various AFP and AFP-compatible products. With ACIF, data and resources can feed into ACIF for processing and can be sent to a customer-supplied archival and retrieval system, to the spool, or to the Viewer application of AFP Workbench for viewing.

ACIF accepts data from your application in the following formats:

- AFP data
- MO:DCA-P data
- S/370 line-mode data
- Mixed-mode data
- Unformatted ASCII data (AIX only)

Put simply, ACIF can process application print data and AFP resources to produce three types of files:

1. An AFP document file
2. An AFP resource file
3. An AFP index object file

With the files that ACIF creates, you can do the following:

- Use PSF to print the AFP document file. If you have specified resources in the AFP document file, PSF for AIX references the AFP resource file for the names and locations of the resources. The AFP document file must be concatenated to the end of the resource file before the file is printed.
- Use the Viewer application of AFP Workbench (hereafter referred to as “Workbench Viewer”) to view the AFP document file. Workbench Viewer takes MO:DCA-P data and resources as input to produce output that can be viewed.
- Store report files and the index file entries created by ACIF in a document archival system, such as IBM OnDemand for AIX. OnDemand operates in a client/server environment and supports small office environments as well as large enterprise installations with hundreds of system users. OnDemand provides a server to store report files and other types of business documents. End-users can search for and retrieve files from the server with client programs that run under Microsoft Windows and MVS CICS/ESA. OnDemand supports full fidelity viewing and reprinting of report files on local and remote printers.
- Use your own archive system to store the ACIF-created files.
- Use your own retrieval system to access information in the ACIF files, using retrieval information in the index object file.

Using ACIF for Different Tasks

ACIF can process your files to allow:

- Viewing with Workbench Viewer
- Printing locally and on other systems
- Selective archiving and retrieval

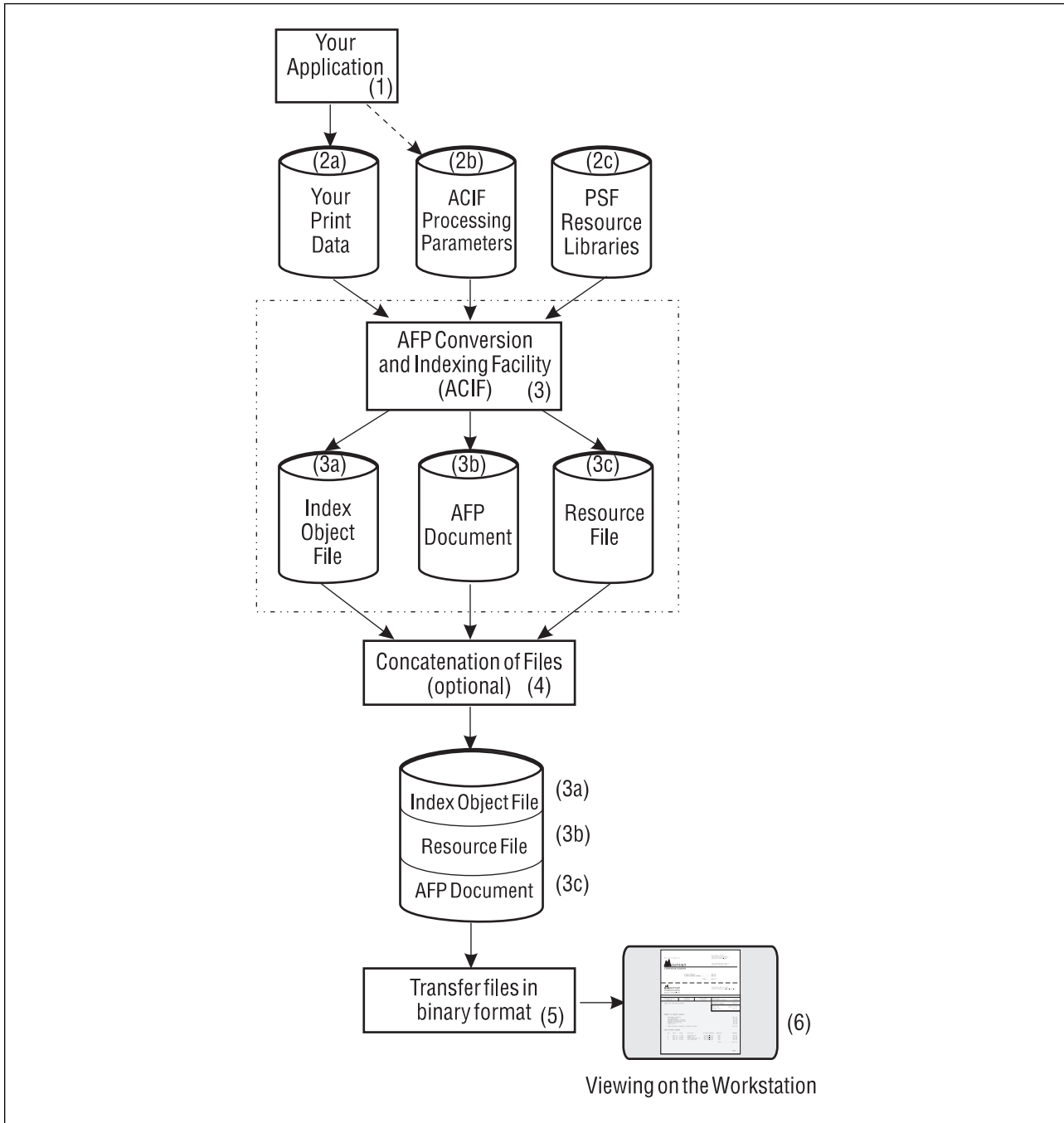


Figure 7. Using ACIF to Prepare Files for Viewing

Figure 7 shows the path your data takes when you are preparing files for viewing with the Workbench Viewer.

1. The process begins with **your application** (1), which is the program that processes your print data.
2. Your application creates **your print data** (2a) and optionally creates **ACIF processing parameters** (2b). Resources are stored in the **PSF resource libraries** (2c).
3. You run **ACIF** (3), specifying that it create the **index object file** (3a), the **AFP document** (3b), and the **resource file** (3c).
4. For optimal performance in locating pages in a file, you **concatenate** (4) the index object file (3a) to the AFP document (3b). If the resources used by the document are not present on the workstation where the Workbench Viewer is installed, you concatenate the resource file (3c) to the AFP document file. The order of concatenation must be as shown in Figure 7 on page 6, with the document file concatenated last.
5. **Transfer** the needed files in binary format to the workstation.
6. Using the Workbench Viewer, **view** your indexed document. You can also print the document from the Workbench Viewer.

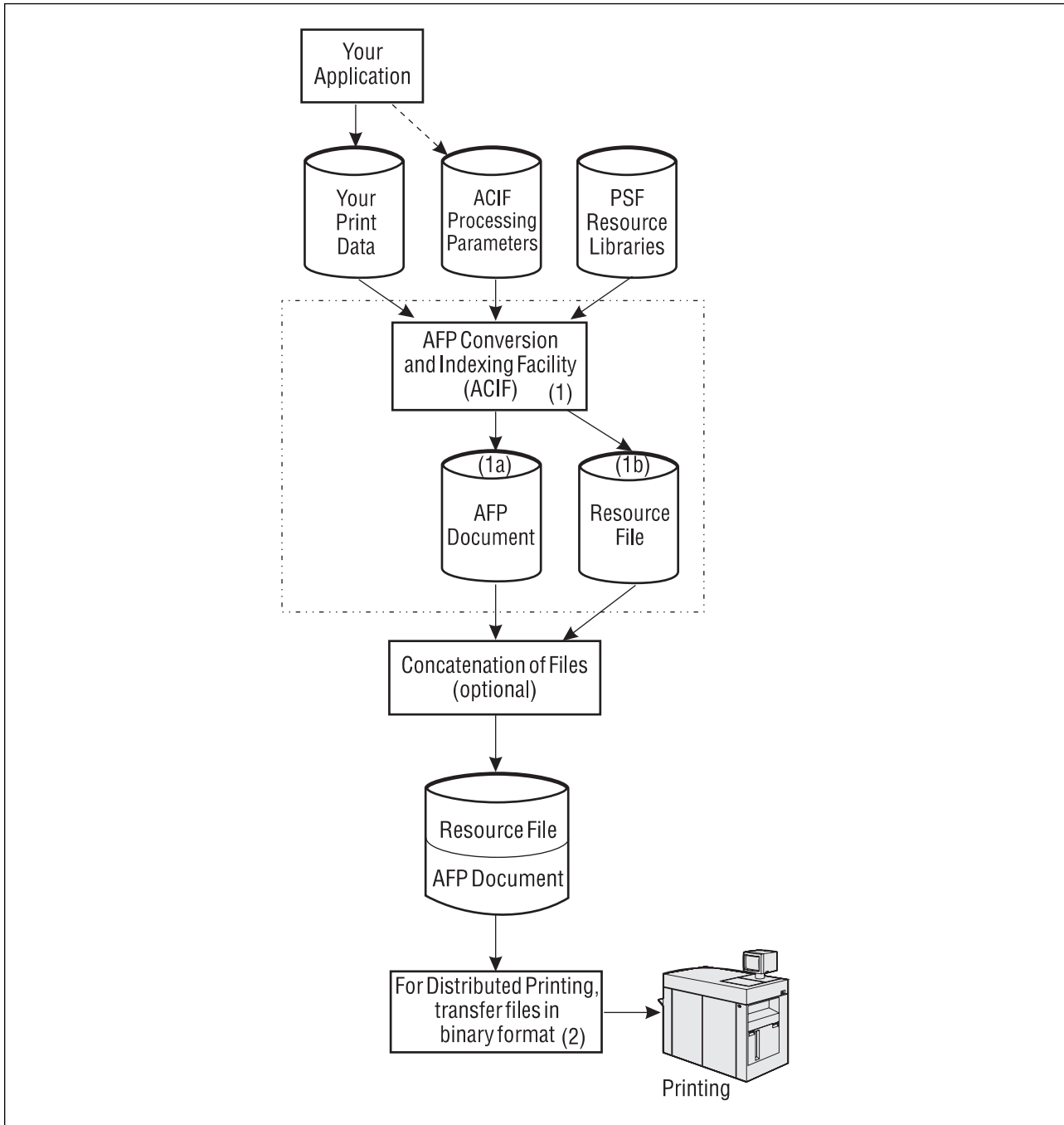


Figure 8. Using ACIF to Prepare Files for Distributed Printing

Figure 8 on page 8 shows the steps to perform to prepare your files for printing:

1. Run ACIF, specifying that the resource file (1b) be created along with the AFP document file (1a).

If you are using ACIF on AIX and your resources reside on another operating system, you can use the AIX Network File System (NFS) to mount them to the AIX system where you are running ACIF.

2. If the print driver program (PSF) that manages jobs for your target printer runs on a different operating system than the one you run ACIF on, transfer the files in binary format to the system where PSF runs.

If your resources are not present on the remote PSF platform, concatenate the AFP document file to the end of the resource file before submitting the file to PSF. If your resources are already present on the remote PSF platform, you do not have to concatenate or transmit them.

3. Submit your MO:DCA-P print job to PSF.

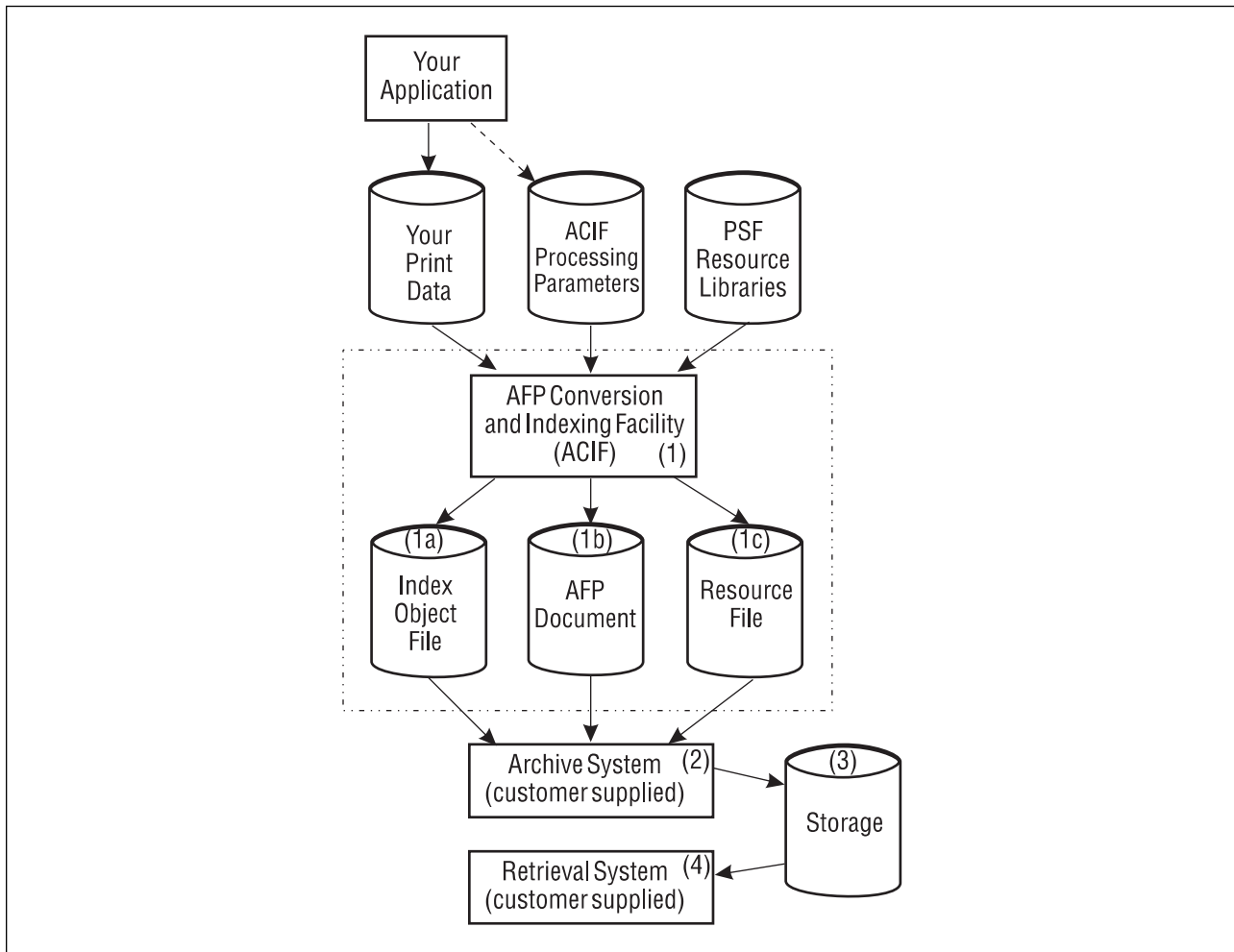


Figure 9. Using ACIF to Prepare Files for Archiving and Retrieving

Figure 9 shows numbered steps you can use to archive your files:

1. Run ACIF, specifying that both the index object file (1a) and the resource file (1c) be created.
2. Run your archival application to archive (3) all three files (1a, 1b, 1c), so that the document can later be retrieved (4) and viewed or printed with fidelity.

Tasks You Can Do with ACIF

This section describes the three tasks for which you can use ACIF. The tasks are listed in the order in which they are described, not in any order in which they should be done.

- Converting data streams
- Indexing documents
- Retrieving resources

Converting Data Streams

ACIF processes the following input data streams to create a MO:DCA-P document.

- AFP data
- MO:DCA-P data
- S/370 line-mode data
- Line-mode data
- Mixed-mode data
- Unformatted ASCII (AIX only)

The following sections describe each type of data and refer you to additional publications, if you need to better understand them:

AFP Data: The AFP data stream is a superset of the MO:DCA-P data stream and supports the following objects:

- Graphics (GOCA)
- Presentation text (PTOCA)
- Image (IOCA and IM)
- Bar-code (BCOCA)

The AFP data stream also supports print resources such as fonts, overlays, page segments, form definitions, and page definitions. For more information on this data stream format, refer to *Mixed Object Document Content Architecture Reference*, which points to publications describing the other types of data objects.

Mixed Object Document Content Architecture Data: ACIF supports MO:DCA-P data as a valid input data stream, with the following restrictions:

- Every structured field must appear in one record and cannot span multiple records.
- Each record (structured field) must contain a X'5A' character before the first byte of the structured field introducer.

ACIF does not change the MO:DCA-P structured fields it processes, because they are already in the correct format. However, although the MO:DCA-P input data stream may contain multiple Begin Document (BDT) and End Document (EDT) structured fields, the ACIF output contains only one BDT/EDT structured-field pair.

For more information about this data stream, refer to *Mixed Object Document Content Architecture Reference*.

S/370 Line-Mode Data: S/370 line-mode data is characterized by records of data that may begin with a single carriage control character, which may be followed by a single table reference character (TRC). After these characters, zero or more bytes of EBCDIC data may follow. ACIF formats S/370 line-mode data into pages by using a page-definition (PAGEDEF) resource, in the same way as does PSF. For more information about line data, refer to *Advanced Function Presentation: Programming Guide and Line Data Reference*.

Mixed-Mode Data: Mixed-mode data is a mixture of line-mode data, with the inclusion of some AFP structured fields, composed-text pages, and resource objects such as image, graphics, bar code, and text. For more information about this data stream, refer to *Advanced Function Presentation: Programming Guide and Line Data Reference*.

Unformatted ASCII Data: Unformatted ASCII data is data that is generated in the workstation environment and has not been formatted for printing. Unformatted ASCII data is formatted by ACIF using a page definition resource. Unformatted ASCII data is contrasted with the type of ASCII data that contains control characters (or, escape sequences) for the IBM Proprinter and Quietwriter, and does not need to be formatted by ACIF before printing with PSF for AIX. Unformatted ASCII can also be submitted for printing with PSF for AIX without being converted by ACIF, but the output format is predetermined (using a Proprinter emulation font and 60 lines per page, for example).

A page definition can be created for use in an unformatted ASCII file to allow the use of AFP functions such as varied print directions, multiple-up printing, and different fonts in the output format. You can use IBM Page Printer Formatting Aid for AIX (PPFA for AIX) to create your own page definitions. PPFA for AIX is a separate feature of PSF for AIX that you can order. For information on how to create page definitions using PPFA for AIX, refer to *IBM Page Printer Formatting Aid for AIX: User's Guide*.

Indexing Documents

One of the principal tasks you can do with ACIF is indexing print files, which are also known as documents. When indexing with ACIF, you can divide (segment) a large print file into smaller, uniquely identifiable units, called *groups*, as defined by the MO:DCA-named group structured fields. Using ACIF, you can divide the large bank-statement application into the individual groups by inserting in the file structured fields that define the group boundaries. A group is a named collection of sequential pages, which, in this application, consists of the pages describing a single customer's account. For example, when you print a bank-statement application, you probably produce a large printout consisting of thousands of individual customer statements. You can think of each of these statements as a smaller, separate unit, each of which is uniquely identified by an account number or other attributes such as date and Social Security number.

Using ACIF, you can also create an index object file, which enables you to:

- Retrieve individual statements from storage, based on an account number or any other attribute
- More rapidly access the statements for viewing by, for example, the Workbench Viewer.
- Archive individual statements or the entire indexed print file for long-term storage and subsequent data management and reprinting, even years after its creation

In addition to building an index-information file containing structured fields (the *index object file*), ACIF also inserts strings of character data called *tags* in the print file in structured-field format. ACIF inserts these same structured fields in the index object file. (The tags are contained in Tagged Logical Element [TLE] structured fields, which are described in Appendix A, “Helpful Hints” and Appendix B, “Data Stream Information”) You can use the indexing-tag structured fields to identify a group of pages.¹ Figure 10 shows the relationship between the group-level tags and the entries in the index object file.

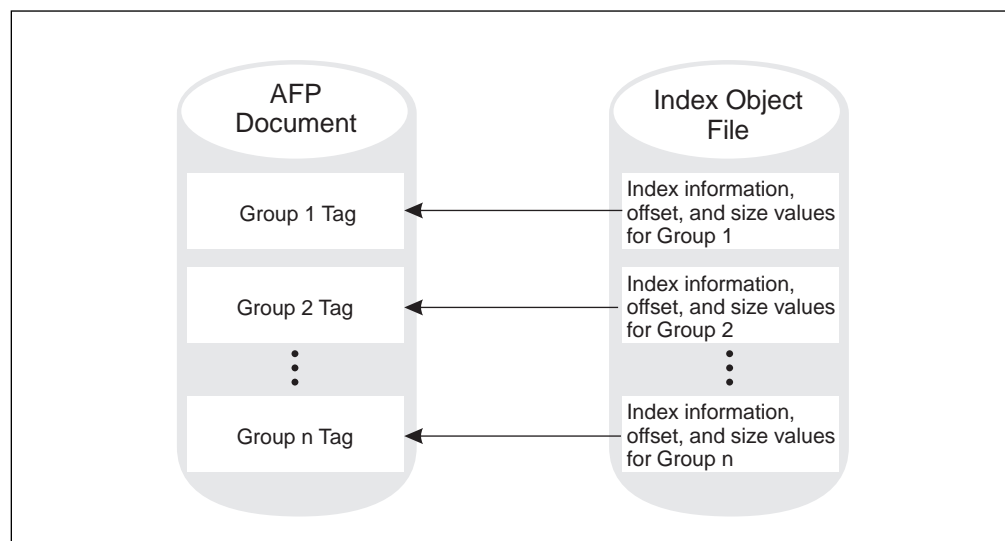


Figure 10. AFP Document with Index Tags and the Index Object File

ACIF can create an index object file for the following types of input files:

- S/370 line-mode data or mixed-mode data
 - Unformatted ASCII data
 - AFP data produced by the AFP Application Programming Interface, Document Composition Facility (DCF), or by AFP Toolbox, with or without indexing tags
- Note:** In this instance, you are producing an index object file from an input file that contains index tags. You are not adding new indexing tags to an existing file.
- AFP data produced by any other application

ACIF provides the following ways for you to generate the indexing tags placed in the print file:

- Using values present in the input data itself, when the data has been formatted so that ACIF can reliably locate the values. This kind of indexing is called *indexing with data values*.
- Using literal values that you specify to ACIF, which is useful when the values you want to use in the indexing tags are not consistently present in the data. This kind of indexing is called *indexing with literal values*.

¹ With ACIF, you can generate group-level tags; with Document Composition Facility (DCF), AFP Toolbox, and AFP Application Programming Interface (AFP API), you can generate both group-level tags and page-level tags. These products will be described later in this chapter.

Indexing with Data Values: Some applications such as payroll or accounting statements contain data that might be appropriate to use for indexing tags. In the bank statement example, the account number is a type of data value that you may want to tag. You can then archive a single customer's account statement using the account number, and you can retrieve and view the same statement using the account number. If the data value you want to use in an indexing tag is consistently located in the same place for each statement, you can specify ACIF parameters that create a separate group of pages for each statement. The ACIF parameters that you use in this case are the **TRIGGERn**, **FIELDn**, and **INDEXn** parameters.

Indexing with Literal Values: Some print files such as technical documents and memos cannot be divided easily into groups of pages using values in the data, because no data value is consistently present in the same location. Likewise, the output of an application may not contain the data you would like to use for an indexing tag. In these cases, you can specify one or more literal values for ACIF to use in the indexing tags for a single group of pages. The ACIF parameter that you use in this case is the **FIELDn** parameter.

Notes:

1. If you are using ACIF to add indexing tags to a file, and the input file already contains indexing tags, ACIF issues an error message and stops processing. If the input file already contains indexing tags, you can create the index object file by running `acif` *without* specifying any indexing parameters.
2. ACIF includes the name of the output document in the index object file and includes the name of the index object file in the output document, which provides a method of correlating the index object file with the appropriate output document.

An Indexing Example Using Data Values: This example shows how to use the ACIF parameters described in Chapter 2, "Using ACIF Parameters in AIX" on page 25 and Chapter 7, "Using ACIF Parameters in MVS, VM, and VSE" on page 87. Figure 11 shows the print file for a typical bank statement.

```
1ACCOUNT NUMBER: 445-66-3821-5          PAGE 1
  CUSTOMER NAME: HENRY WALES
  DATE: 09/30/91
  CHECK# 001 - 455.00
  CHECK# 002 - 337.85
  ...
1ACCOUNT NUMBER: 333-56-4378-5          PAGE 1
  CUSTOMER NAME: KATHERINE CHARLES
  DATE: 09/30/91
  CHECK# 221 - 5.00
  CHECK# 222 - 1567.35
  ...
```

Figure 11. Example Bank Statement Input File

In Figure 11 on page 14, the print file contains bank statements dated September 30, 1991 (09/30/91). Each statement has the same general format, although statements may vary in size or number of pages. Assume you want to index the bank statements using the account number and the date. Although the account number identifies each customer's account, the date is important to differentiate one month's statement from another. For ACIF to extract the account number and date, it must first locate the records that contain the required information.

Because ACIF can process different data streams with various file formats (carriage control characters, no carriage control characters, table-reference characters, and so on), it requires *triggers* to determine an *anchor point* from which it can locate the necessary index values. You may require multiple triggers to uniquely identify the start of a new statement. To index the bank statements using the account number and the date, first define the trigger values and the fields as shown in Figure 12:

```
trigger1=*,1,'1'  
trigger2=0,39,'PAGE 1'  
field1=0,18,3  
field2=0,22,2  
field3=0,25,4  
field4=0,30,1  
field5=2,8,2  
field6=2,11,2  
field7=2,14,2  
index1='Account Number',field1,field2,field3,field4  
index2='Date',field5,field6,field7
```

Figure 12. ACIF Processing Parameters to Index the Bank Statement

The information in Figure 12 defines two trigger values:

- The first trigger instructs ACIF to examine the first byte of every input record until it finds the occurrence of an ANSI skip-to-channel 1 carriage control character ('1'). Because each page created by this particular application may contain this carriage control character, this trigger alone does not identify the start of a new bank statement.
- The second trigger accomplishes this task. When ACIF locates a record containing a '1' in the first byte, it looks for the string 'PAGE 1' in that same record, starting at byte (column) 39. If this condition is found, a new statement exists, and ACIF uses the record containing **trigger1** as the anchor point. The **field** definitions are relative to this anchor point.

In the example (Figure 12), the account number has four fields. These fields can be defined as one field if the dashes are included as part of the index information. The date has three fields to remove the forward slashes. After ACIF has extracted all of the necessary indexing information for this statement, it begins looking for **trigger1** again. This process is repeated until the entire print file is processed.

In summary, when ACIF indexes an input file, it first scans the input file to find matches for its parameters. When ACIF finds matches in the input file, it inserts structured fields immediately prior to the corresponding pages of the output file. Also, ACIF places structured fields in the index object file that point to matches in the output file. ACIF inserts structured fields before the corresponding pages of the output file where it finds the matches in the input file. ACIF also places structured fields that point to these matches in the index object file.

Indexing Limitations: For a line-mode application that does not contain the appropriate data values in the application output and for which literal values are not suitable, the application program cannot insert tagging structured fields in the print data, because tagging structured fields are not allowed in mixed-mode data. In the case where the application data does not contain the necessary appropriate data values for indexing, the application could add the index triggers. One possible location is the record containing the new-page carriage control character (for example, a skip-to-channel 1). The application would need to add the indexing trigger and attribute value to this record at a specified location on each statement in the print file. This allows ACIF to retrieve this information at processing time. (For information about different types of carriage control characters, see the description of the **cctype** parameter on page 30 for AIX, or page 90 for MVS, VM, and VSE.)

Retrieving Resources

ACIF can determine the list of required AFP resources needed to view or print the document and can retrieve these resources from the specified libraries. Then, you can view or print the document with fidelity. This ACIF function is especially valuable if the resources are not present on the designated platform in a distributed print environment.

When you archive a document, ACIF also allows you to archive the retrieved resources (fonts, page segments, and so on) in the form in which they existed when the file was printed. By archiving the original resources, you can reproduce the document with fidelity at a later date, even if the resources have changed since that time. For example, suppose that a page segment contains a company officer's signature and is included in the print data. When someone else replaces the officer, current print files must reference the new officer's signature, but archived files must reference the former officer's signature.

The type of resources ACIF retrieves from specified libraries is based on the value of the **restype** parameter. When ACIF processes a print file, it:

- Identifies the resources requested by the print file

While ACIF converts the input file into an AFP document, it builds a list of all the resources necessary to successfully print the document, including all the resources referenced inside other resources. For example, a page can include an overlay, and an overlay can reference other resources such as fonts and page segments.

- Creates a resource file

ACIF creates a logical resource library in the form of an AFP resource group and stores this resource group in a resource file. If you specify **restype=fdef,font,pseg,ovly,objcon** or **restype=all**, this resource file contains all the resources necessary to view or print the document with fidelity. Each time ACIF processes a print file, it can create a resource file in one of two different formats:

- A partitioned data set (PDS). The PDS format is supported only on MVS and allows the resource file to be referenced as a user library (USERLIB) when printing with PSF/MVS.
 - An AFP data-stream resource group. The AFP resource-group format is useful when you are routing print output to remote AFP platforms (for example, PSF/2 on OS/2) or when you are viewing the output with the Viewer application of AFP Workbench.
- Calls the specified resource exit for each resource it retrieves
- Before ACIF retrieves a resource from a library, it first calls the resource exit program as specified in the **resexit** parameter. You can write an exit program to filter out any resources you do not want included in the resource file. For example, the exit program can specify that all referenced fonts except for a specific typeface be included in the resource file. The only way to accomplish this is by using the resource exit.
- Includes the name of the output document in the resource file and the name of the resource file in the output document, which provides a method of correlating resource files with the appropriate output document

An example of specifying ACIF processing parameters for resource retrieval can be found in Chapter 8, “Example: ACIF Application in MVS, VM, or VSE” on page 111 and Chapter 3, “Example of an ACIF Application in AIX” on page 53.

What Other IBM Products are Related to ACIF?

Although ACIF is a stand-alone utility, it has been designed for use with other programs. These programs are described in this section:

- The Workbench Viewer
- AFP Application Programming Interface (AFP API)
- AFP Toolbox
- Document Composition Facility (DCF)

The Workbench Viewer

Figure 13 shows how Workbench Viewer can display documents on a workstation running Microsoft Windows or OS/2. These documents can contain an index object file and a resource group.

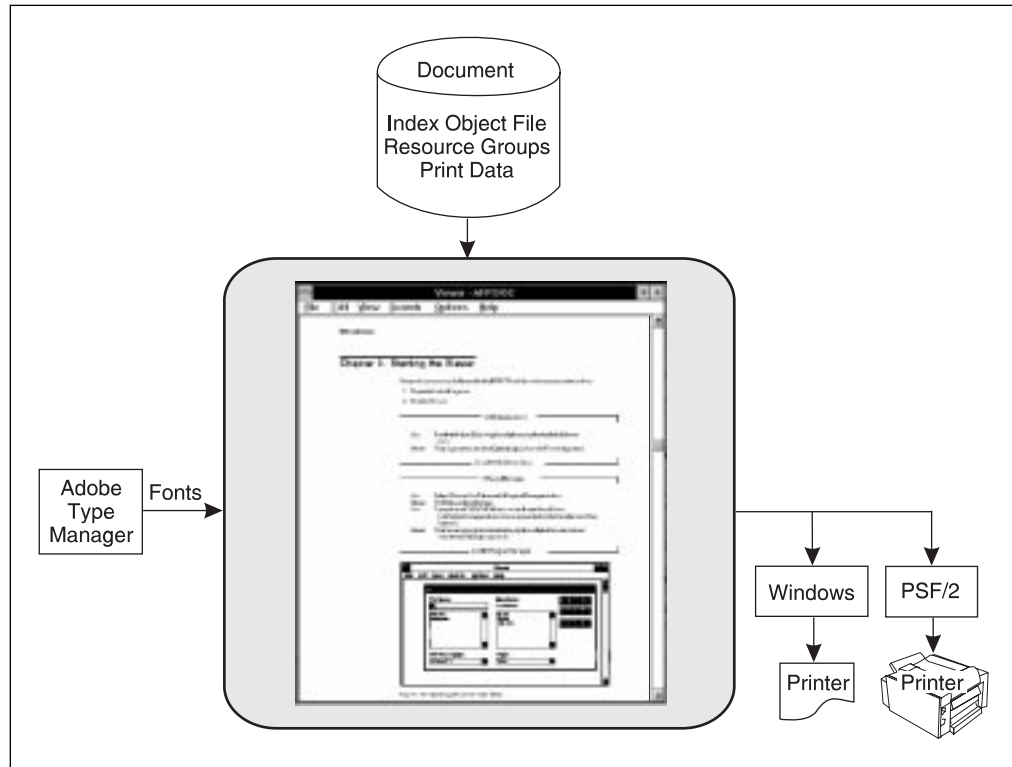


Figure 13. Workbench Viewer

Workbench Viewer uses Adobe Type 1 outline fonts when displaying documents. If the document references a font for which no Type 1 font is available at the workstation, Workbench Viewer can substitute an outline font for the requested font. Workbench Viewer matches the requested point size and attempts to match the typeface as closely as possible. Font definition files are available with Workbench Viewer to allow you to define which Type 1 fonts are to be substituted for your AFP fonts. Because Workbench Viewer does not use AFP fonts, you do not need to specify the **restype=font** ACIF parameter when preparing a document to use with Workbench Viewer.

When using ACIF to index a file for viewing, specify **indexobj=all**. This setting provides Workbench Viewer with the most complete indexing information for accessing groups of pages in a file. Also, concatenate the index object file to the document for optimal performance of Workbench Viewer. (It is important that the document file comes last, at the end of the resulting concatenated file; otherwise, an error will occur.)

Workbench Viewer limits the size of the attribute names in indexing information to 64 bytes. When indexing data for viewing, make your attribute names unique within the first 64 bytes. (ACIF allows up to 250 bytes for attribute names.)

Workbench Viewer supports a subset of MO:DCA-P data and may not display everything that PSF can print. For example, OS/2 Workbench Viewer recognizes

but ignores all graphics in GOCA format and bar code (BCOCA) objects, whereas PSF can print these objects.

Note: The Windows version of Workbench Viewer does not recognize GOCA. For more information about Workbench Viewer, refer to *AFP Workbench for Windows and OS/2: Using the Viewer Application*.

AFP Application Programming Interface

AFP Application Programming Interface (AFP API) is a related product you can use with ACIF to index your data, among other functions. Using AFP API, a programmer who knows COBOL or PL/1 can format complex output without knowing the syntax and semantics of MO:DCA-P. Using AFP API, you can index (tag) your document with both group-level and page-level indexing tags, allowing more specific indexing information to be included in the file, whereas, with ACIF, you can add only group-level indexing tags. You can then process the AFP API output file with ACIF to create a separate index object file for increased performance when viewing the output file. Only ACIF generates the index object file. AFP API and ACIF compliment each other while sharing similar, but not identical, indexing capabilities.

AFP API is shipped with PSF/MVS Version 2.1.1, with PSF/VM Version 2.1.1, and with PSF/VSE Version 2.2.1. For more information, see *AFP Application Programming Interface: Programming Guide and Reference*.

The IBM AFP Toolbox for Multiple Operating Systems

Similar to programmers who use AFP API, an OS/2 or AIX programmer who knows C or C++ languages can use the IBM AFP Toolbox for Multiple Operating Systems program to index data and to format complex output without knowing the syntax and semantics of MO:DCA-P. You can then process the output from the AFP Toolbox with ACIF to create the external index file. For more information about AFP Toolbox, see the *AFP Toolbox for Multiple Operating Systems User's Guide*. For a detailed description of the MO:DCA data stream, see *Mixed Object Document Content Architecture Reference*.

Document Composition Facility (DCF)

Document Composition Facility (DCF) is a program used primarily to prepare and format documents for printing. It is another product that can be used with ACIF to index your data in the MVS, VM, or VSE environments. Along with its many other features, DCF provides the ability to add both group-level and page-level indexing tags; whereas, with ACIF, you can add only group-level indexing tags. Only ACIF generates the index object file.

In DCF, the indexing function is known as “navigation.” DCF also provides a very different function already called “indexing.” In DCF terminology, you “navigate” through a document with the viewing application, and its indexing function is used to build an alphabetical listing of page references (a “back-of-the-book index”).

Support for navigation (indexing) is provided with DCF Version 4.0. APAR PN36437 is required to enable the support.

For further information about DCF, refer to *Document Composition Facility Script/VS Language Reference*, (SH35-0070). Note that DCF is not applicable to the AIX environment.

What are the Systems Considerations for ACIF?

You must consider the following when using ACIF:

- System limitations
- System prerequisites

System Limitations

If you are using ACIF to build applications for PSF/MVS, PSF/VM, or PSF/VSE, you need to take the following limitations into consideration:

PSF/MVS Limitations

To print an ACIF output file that contains indexing information, you must have either PSF/MVS 2.1.1 or PSF/MVS 2.1.0 with PTF number UY96049 installed.

PSF/VM Limitations

To print an ACIF output file that contains indexing information, you must have PSF/VM 2.1.1 or PSF/VM 2.1.0 with PTF number UN37799 installed.

PSF/VSE Limitations

PSF/VSE does not support inclusion of fonts, page segments, and overlays in the print file. If you use ACIF to retrieve these resources, do not concatenate the ACIF resource file to the print file. If you want to print an ACIF output document using PSF/VSE, ensure that the resources are present on that platform.

To print an ACIF output file that contains indexing information, you must have either PSF/VSE 2.2.1 or PSF/VSE 2.2.0 with APAR DY42845 installed.

System Prerequisites

The following section describes system prerequisites necessary to use ACIF in the AIX, MVS, VM, and VSE environments.

AIX Prerequisites

The following AIX software products are required to use ACIF:

- IBM AIX for RISC System/6000 Version 3.2.5 (program number 5756-030) or higher.
- PSF for AIX Version 2.1 (program number 5765-505) with, minimally, the **psf.acif** option installed.

Notes:

1. Installation of *only* **psf.acif** will provide you with the ability to run the **acif** command.

The **formdef** parameter is required for running ACIF. Though you may use form definitions from other sources, the **psf.acif** install option provides a form definition named F1ACIF. (See the description for form definition F1A10110 in *IBM Print Services Facility for AIX: Print Submission* for details regarding form definition F1ACIF.)

The **pagedef** parameter is required for running ACIF if the input file contains S/370 line data, mixed-mode data, or unformatted ASCII data. Though you may use page definitions from other sources, several page definitions are provided with PSF for AIX. The PSF-supplied page definitions are installed into the **/usr/lpp/psf/reslib** directory with the **psf.acif** option. These page definitions are listed and described in *IBM Print Services Facility for AIX: Print Submission*.

2. In order to print and to use the other resources that are available for ACIF, you must *also* install **psf.base**.

For more information about PSF for AIX requirements, refer to *Licensed Program Specifications for IBM Print Services Facility for AIX*.

MVS Prerequisites

The following MVS software products are required to use ACIF:

- MVS/SP Version 2.2.0 or above
- MVS/ESA Version 4.1.0 or above

PSF/MVS 2.1.0 (with PTF# UY96049 for printing files that contain indexing tags) or PSF/MVS 2.1.1

VM Prerequisites

The following VM software products are required to use ACIF:

- VM/SP 5 or above
- VM/SP HPO 5 or above
- VM/XA 1.2.1 or above
- VM/ESA 1.1.0 or above

PSF/VM 2.1.0 (with PTF# UN37799 for printing files that contain indexing tags) or PSF/VM 2.1.1

VSE Prerequisites

The following VSE software products are required to use ACIF:

- VSE/SP 4.1.2 or above
- VSE/ESA 1.1.0 or above

PSF/VSE 2.2.0 (with APAR DY42845 for printing files that contain indexing tags) or PSF/VSE 2.2.1 or above

Note: You can use later versions or releases of these products. Each of the above products may require additional software products. Refer to their respective publications for the current list of system requirements.

Part 2. Using ACIF in the AIX Environment

Chapter 2. Using ACIF Parameters in AIX

This section describes the **acif** command for AIX, including syntax rules, parameters, and values.

Purpose

Transforms S/370 line data and unformatted ASCII files into MO:DCA-P files for printing, viewing, archiving, and retrieving.

Syntax

```
acif [cc=value] [cctype=value] [chars=fontnames] [comsetup=name]
[cpgid=value] [dcfpagenames=value] [fdeflib=pathlist]
[fieldn={record,column,length} | {literal value | X'literal value'}] [fileformat=value]
[fontlib=pathlist] [formdef=fdefname] [groupname=value] [imageout=value]
[indexn={'attribute name' | X'attribute name'},field definition] [indexdd=filename]
[indexobj=value] [indexstartby=value] [indexexit=programname]
[inpexit=programname] [inputdd=filename] [msgdd=filename] [objconlib=pathlist]
[outexit=programname] [outputdd=filename] [ovlylib=pathlist] [pagedef=pdefname]
[parmdd=filename] [pdeflib=pathlist] [prmode=value] [pseglib=pathlist]
[resexit=programname] [reslib=pathlist] [resobjdd=filename] [restype=value]
[trc=value] [triggern={record | *},{column | *},{'trigger value' | X'trigger value'}]
[uniquebngs=value] [userlib=pathlist]
```

The syntax shown here is what the **acif** command expects to receive. For example, **acif** expects to receive literal single quote characters for the **field**, **index**, and **trigger** parameters. In order for ACIF to receive these single quote characters, you must “escape” the quote characters so that your shell will not parse them. The way you “escape” quote characters depends on the shell you are using. If you need guidance in passing the **acif** command parameter syntax through the shell, refer to the documentation for the shell you are using in *AIX RISC System/6000 Commands Reference*.

The following **acif** command parameters specify resource names or resource directories, and are equivalent to the parameters specified to PSF for AIX when you print a job: **objconlib**, **fontlib**, **fdeflib**, **pseglib**, **reslib**, and **ovlylib**. For these parameters, you should specify the same value with the **acif** command as you specify to PSF for AIX with an AIX print command. In this way, the search paths and resources used at transform time are identical to the search paths and resources used at print time.

Note: You may need to consult your system support group for information on resource directories and other printing defaults contained in the PSF for AIX printer profiles used in your installation.

In addition to the notational conventions shown in “Why are some words highlighted, italicized, or underscored?” on page xiv, the **acif** command follows these additional rules. These rules apply to parameters you type at the command line. See also the **parmdd** parameter on page 43 for the syntax of parameters contained in the **acif** parameter file.

- When the **acif** command processes any unrecognized or unsupported parameter, **acif** issues a message, ignores the parameter, and continues processing any remaining parameters. The **acif** command then stops processing.

- Though the parameters themselves are not case-sensitive, associated values, such as file names, attribute names, and directory names *are* case-sensitive. For example,

```
formdef=F1MINE
```

is *not* the same as

```
formdef=f1mine
```

Be sure to specify these values in the case in which they exist in the file system (for external resources) or in the print file (for inline resources).

- If the same parameter is specified more than one time, the **acif** command uses the last value specified. For example, if you specify the following:

```
pagedef=P1MINE  
pagedef=P1YOURS
```

the **acif** command uses page definition P1YOURS only.

Description

The **acif** command transforms S/370 line data, mixed-mode data, and unformatted ASCII files into the Mixed Object: Document Content Architecture for Presentation (MO:DCA-P) data stream. With this data stream, you can do the following:

- Print the file on a printer defined to PSF for AIX, or to other PSF products
- View the file using a viewer product such as AFP Workbench
- Archive and retrieve the file using your own archival management system

The **acif** command provides indexing and resource retrieval capabilities, which enable you to perform the viewing, archiving, and retrieving tasks with the appropriate system and software.

Three types of files can be generated, depending on what output options you specify:

- An AFP document file
- An AFP resource file
- An index object file

When you use the **acif** command to convert S/370 line data or unformatted ASCII data, you must specify a page definition (**pagedef** parameter). If the page definition names some fonts, the **acif** command uses those fonts. If the page definition does not name any fonts (like the sample page definitions supplied with PSF for AIX), and if you want the file to print with more than one font, then the input file must contain table reference characters, and you must:

- Specify **trc=yes**.
- Use **chars** to indicate the fonts to be associated with each Table Reference Character (TRC). *fontname1* is associated with TRC 0, *fontname2* is associated with TRC 1, and so on.

Note: If the page definition does not specify fonts, and you have not specified any TRCs, your job will print, although the output may not be formatted correctly.

If the page definition does not name any fonts, and you want the whole file to print with only one font, then you must:

- Specify **trc=no**.
- Use **chars** to indicate the single font in which the file should be printed.

The **acif** command searches for resources in the following order:

1. Paths specified by the **userlib** parameter
2. Paths specified by the **fdeflib**, **fontlib**, **pdeflib**, **pseglib**, **objconlib**, and **ovlylib** parameters for specific types of resources
3. Paths specified by the **reslib** parameter
4. Paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**
6. The directory **/usr/lpp/afpfonts**
7. The directory **/usr/lpp/psf/fontlib**

When the **acif** command finds more than one resource with the same name in the same directory, it selects the resource to be used depending on the file extension. Figure 14 shows the order in which resources with the same name but different file extensions are used by **acif**.

Note: If a file name includes a period (.), the file extension is that part of the file name that follows the period. For example, the file extension of the file name ARTWORK.PSEG3820 is PSEG3820.

<i>Figure 14. File Extensions for Resources</i>	
Type of Resource	File Extensions Searched (see note)
Form definitions	<ol style="list-style-type: none"> 1. No file extension 2. FDEF3820 3. FDEF38PP 4. FDE
Page definitions	<ol style="list-style-type: none"> 1. No file extension 2. PDEF3820 3. PDEF38PP 4. PDE
Fonts, 240-pel resolution	<ol style="list-style-type: none"> 1. No file extension 2. 240 3. FONT3820 4. FONT38PP
Fonts, 300-pel resolution	<ol style="list-style-type: none"> 1. No file extension 2. 300 2. FONT300
Page segments	<ol style="list-style-type: none"> 1. No file extension 2. PSEG3820 3. PSEG38PP 4. PSG
Overlays	<ol style="list-style-type: none"> 1. No file extension 2. OVLY3820 3. OVLY38PP 4. OVL
Coded fonts	<ol style="list-style-type: none"> 1. No file extension 2. FONT3820 3. FONT38PP 4. CFT
Setup data	<ol style="list-style-type: none"> 1. No file extension 2. SETUP 3. SET 4. COMSETUP
Note: All file extensions must be in uppercase format.	

You can use ACIF to prepare S/370 line data or unformatted ASCII files. At print submission time, to *automatically* invoke the **acif** command for the purpose of preparing S/370 line data or unformatted ASCII files for printing with PSF for AIX, use the **-odatatype=line** flag and keyword-value pair with one of the AIX print commands (**enq**, **lp**, or **qprt**), or use the **psfin** command to specify a job script with a setting of **-JsFiletype=line**.

Note: The **line2afp** command of PSF for AIX is a subset of the **acif** command and uses the **acif** command parameters for conversion to produce output for printing. As does ACIF, the **line2afp** command uses a page definition to define how the data is to be formatted on the printed page. If you use the **line2afp** command, you will be able to transform and print files, but you will not be able to take advantage of the indexing and resource retrieval functions for viewing. The **line2afp** command is described in *IBM Print Services Facility for AIX: Print Submission*.

The following parameters are used for just the *conversion* function of ACIF, with either the **acif** command or the **line2afp** command: **cc**, **cctype**, **chars**, **fdeflib**, **fileformat**, **fontlib**, **formdef**, **imageout**, **inpexit**, **inputdd**, **msgdd**, **outexit**, **outputdd**, **ovlylib**, **pagedef**, **parmdd**, **pdeflib**, **prmode**, **pseglib**, **resexit**, **reslib**, **trc**, and **userlib**.

The following parameters are used for *resource retrieval* with the **acif** command: **comsetup**, **fdeflib**, **fontlib**, **formdef**, **objconlib**, **ovlylib**, **pseglib**, **resexit**, **resobjdd**, and **restype**.

The **acif** command uses the following parameters for its *indexing* functions: **cpgid**, **fieldn**, **groupname**, **indexn**, **indexdd**, **indexobj**, **indexstartby**, **indxexit**, and **triggern**.

Flags and Values

cc={yes | no}

Specifies whether the input file has carriage-control characters. Values are:

yes The file contains carriage-control characters.

no The file does not contain carriage-control characters.

Carriage-control characters, if present, are located in the first byte (column) of each line in a document. They are used to control how the line will be formatted (single space, double space, triple space, and so forth). In addition, other carriage-controls can be used to position the line anywhere on the page. If there are no carriage-controls, single spacing is assumed.

cctype={z | a | m}

Specifies the type of carriage-control characters in the input file. The **acif** command supports ANSI carriage-control characters in either ASCII or EBCDIC encoding, as well as machine carriage-control characters. The **acif** command does not allow a mixture of ANSI and machine carriage-control characters within a file. Values are:

z The file contains ANSI carriage-control characters that are encoded in ASCII.

The carriage-control characters are the ASCII hexadecimal values that directly relate to ANSI carriage-controls, which cause the action of the carriage-control character to occur *before* the line is printed. For example, if the carriage-control character is zero (X'30'), which represents double spacing, double spacing will occur *before* the line is printed.

- a The file contains ANSI carriage-control characters that are encoded in EBCDIC.

The use of ANSI carriage-control characters cause the action of the carriage-control character to occur *before* the line of data is printed. For example, if the carriage-control character is a zero (X'F0'), which represents double spacing, the double spacing will occur *before* the line is printed.

- m The file contains machine code carriage-control characters that are encoded in hexadecimal format.

The use of machine code carriage-control characters cause the action of the carriage-control character to occur *after* the line of data is printed. For example, if the carriage-control character is a X'11', which represents double spacing, the line will be printed and the double spacing will occur *after* the line is printed. In addition, machine code carriage-control has a set of carriage-control characters that perform the action, but do not print the associated line. For example, if the carriage-control character is a X'13', which also represents double spacing, the print position will be moved down two lines but the line that contains the X'13' carriage-control character will not be printed. The next line in the data will be printed at the current print position and the action for the associated carriage-control character will be performed *after* the line is printed.

If you specify **cc=yes** but you do not specify **cctype**, the **acif** command assumes that the file contains ANSI carriage-control characters encoded in ASCII.

If you are not sure which type of carriage-control characters are in your input file, consult your system support group.

chars=fontname1, fontname2, fontname3, fontname4

Specifies the file name of from one to four coded fonts to be used in processing the print file. A coded font specifies a character set and code page pair. The value is:

fontname The name of the desired coded font. The font name is limited to four alphanumeric or national characters, and should not include the two-character prefix of the coded-font name (X0 through XG). The font name is case-sensitive.

If you use the ASCII fonts that are supplied with PSF for AIX, use the four-character names as shown in Chapter 5, "IBM AFP Fonts for ASCII Data" on page 77. If you use your own coded font that has a file name with more than six characters (including the Xn prefix), then do one of the following:

- Rename the font file to a shorter name. For example,
mv X0423002 X04202
- Copy the font file to a file that has a shorter name. For example,
cp X0423002 X04202
- Link the original font file to a shorter name. For example,
ln -s X0423002 X04202

When you use the **acif** command to convert S/370 line-mode data or unformatted ASCII data, you must specify a page definition (**pagedef** parameter). If the page definition names some fonts, the **acif** command uses those fonts, and ignores the **chars** parameter. If the page definition does not name any fonts (like the sample page definitions supplied with PSF for AIX), and if you want the file to print with more than one font, then the input file must contain table reference characters, and you must:

- Specify **trc=yes**.
- Use **chars** to indicate the fonts to be associated with each Table Reference Character (TRC). *fontname1* is associated with TRC 0, *fontname2* is associated with TRC 1, and so on.

If the page definition does not name any fonts, and you want the whole file to print with only one font, then you must:

- Specify **trc=no**.
- Use **chars** to indicate the single font in which the file should be printed.

Consider the following when specifying fonts with the **chars** parameter:

- If the input file is unformatted ASCII, you can do one of the following:
 - Specify a font that has the appropriate ASCII code points. To specify a font search path, either use the **fontlib** parameter to specify it explicitly, or set the **PSFPATH** environment variable to search the appropriate directories. See Chapter 5, “IBM AFP Fonts for ASCII Data” on page 77 for a list of suggested ASCII fonts.
 - Use the **apka2e** or **asciinpe** input record exit programs to convert the ASCII code points in the input file into EBCDIC, and use EBCDIC fonts. To do this, use the **inpexit** parameter, specifying:

```
inpexit=/usr/lpp/psf/bin/apka2e
```

or

```
inpexit=/usr/lpp/psf/bin/asciinpe
```

See the **inpexit** parameter on page 41 for a description of **apka2e** and **asciinpe** functions.

- You can specify fonts with the **chars** parameter only if you want the entire file printed in a single printing direction. The **acif** command uses the fonts that have 0° character rotation for the specified direction. When a file requires fonts with more than one print direction or character rotation, you must specify the fonts in the page definition.
- You can specify from one to four fonts with the **chars** parameter. If you specify more than one font with the **chars** parameter, then the input file must contain table reference characters, and you must specify **trc=yes**.

- If you use **chars** to specify fonts, but you also use the **pagedef** parameter to specify a page definition that names fonts, the **chars** parameter is ignored. Therefore, if your page definition names fonts, you should not use **chars**.
- If you do not specify a **chars** parameter, and if no fonts are contained in the page definition you specified, the **acif** command uses the default font that is set in the printer's hardware.

comsetup=*name*

Specifies the name of a COM setup file. A COM setup file is an AFP resource that contains instructions required when printing on a microfilm device. The value is:

name Any valid COM setup file name. The *name* can be one to eight alphanumeric or national characters, including the two-character prefix, if there is one. The *name* is case-sensitive.

Note: If the file name of the COM setup file includes a file extension, do not use the file extension when specifying the setup file. For example, to use a setup file named **mysetup.SET**, specify **setup=mysetup**.

The COM setup file you use may be located:

- In an AIX directory
- Inline in the file (that is, within the file itself)

If the COM setup file is in an AIX directory, use the **userlib** or **objconlib** parameter to specify the path to the file. For example:

```
comsetup=mysetup userlib=/usr/afp/resources
```

or

```
comsetup=mysetup objconlib=/usr/lib/setups
```

If the COM setup file is an inline resource, you must do the following:

- Specify **comsetup=*name***, where *name* is the name of the inline COM setup file; or specify **comsetup=dummy**.

If you specify **comsetup=dummy** but the file does not include an inline COM setup file, the **acif** command looks for the COM setup file named **dummy**.

If the name specified for the **comsetup** parameter does not match the name of an inline COM setup file, the **acif** command looks for the COM setup file in the **comsetup** search path.

An input file can contain multiple COM setup files, but only one COM setup file can be used for printing. If a file contains more than one COM setup file, and you specify **comsetup=*name***, ACIF uses the first inline COM setup file named *name*. If a file contains more than one inline COM setup file, and you specify **comsetup=dummy**, ACIF uses the first inline COM setup file in the input file.

cpgid={ 850 | *code page identifier*}

Specifies the code page used when the index values and attribute names were specified on the **index***n* and **field***n* parameters. Values are:

850 IBM code page 850

code page identifier

Any valid code page, which is a three-character decimal value (for example, 395) that defines an IBM-registered code page

ACIF uses this code page identifier value when it creates a Coded Graphic Character Set Global Identifier Triplet X'01' in the Begin Document (BDT) structured field for the output file. For more information on this triplet, refer to *Mixed Object Document Content Architecture Reference*.

The code page identifier value is used by programs, such as Workbench Viewer, that must display indexing information. These programs use this value with code page translation tables to represent the index attribute and value data. If this parameter is not specified, ACIF uses code page 850 as the default. For code-page numbers less than 100, add leading zeros (for example, 037). If a non-decimal value is specified, ACIF reports an error condition and terminates processing.

dcfpagenames={**yes** | **no**}

Specifies whether ACIF generates page names using an 8-byte counter or uses structured field tokens found in the input data stream. If the input data contains BPGs with FQNs, ACIF does not generate page names.

yes ACIF uses structured field tokens in the input data stream to generate page names.

no The default, ACIF generates page names using an 8-byte counter.

fdeflib=*pathlist*

Specifies the directories in which form definitions are stored. The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths. The **acif** command searches the paths in the order in which they are specified.

The **acif** command searches for the form definition in the following order:

1. The paths you specified with **userlib**, if any
2. The paths you specified with **fdeflib**, if any
3. The paths you specified with **reslib**, if any
4. The paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**

For information on how PSF for AIX selects resources, refer to *IBM Print Services Facility for AIX: Print Administration*.

field*n*={ *record,column,length*} | {' *literal value*' | X' *literal value*' }

Specifies the data fields to be used to construct the indexing information. Either these data fields can be specified as literal values (constants), or ACIF can retrieve the data from the input records of the file. A maximum of 16 fields can be defined (**field1** through **field16**). The definition of a field includes:

record Specifies the relative record number from the indexing anchor record. When ACIF is indexing the file, it uses the information specified in the **trigger***n* parameters to determine a page group boundary. When all of the specified **trigger***n* values are true, ACIF defines the indexing anchor record as the record where **trigger1** is located. **trigger1** becomes the reference point from which all indexing information is located. The supported range of values for *record* are ±0 to 255.

column Specifies the byte offset from the beginning of the record. A value of 1 refers to the first byte in the record. For files containing carriage-control characters, column 1 refers to the carriage-control. For those applications that use a specific carriage-control character to define page boundaries (for example, skip to channel 1), consider defining the value of the carriage-control character as one of the **trigger***n* parameters. The supported range of values for *column* are 1 to 32756. If the specified value exceeds the physical length of the record, ACIF reports an error condition and terminates processing.

length Specifies the number of contiguous bytes (characters) starting at *column* that compose this field. The supported range of values for *length* are 1 to 250.

The field can extend outside the record length, as long as the column where it begins lies within the record length. In this case, ACIF adds padding blanks to fill out the record. If the field begins outside the maximum length of the record, ACIF reports an error condition and terminates processing.

literal value | X' *literal value*'

Specifies the literal (constant) value of the **field***n* parameter. If the input data file contains unformatted ASCII data, this value can be specified either as character data or hexadecimal data. If the input data file is *anything other than* unformatted ASCII, the value must be specified as hexadecimal data (otherwise, the comparisons between the input file and what is coded in the **field***n* parameter will not yield a match). The literal value can be 1 to 250 bytes in length. ACIF does not perform any validity checking on the actual content of the supplied data.

For example, to specify five fields in your print job, you would enter:

```
field1=0,2,20  
field2=5,5,10
```

```
field3=-15,30,5
field4='444663821'
field5=X'0001'
```

In the preceding example, the fields have the following values:

- The first field in the example is located in the indexing anchor record (**trigger1**). The field is 20 bytes in length starting at the second byte of the record.
- The second field is located five records down from the indexing anchor record. The field is 10 bytes in length starting at the fifth byte of the record.
- The third field is located 15 records before the indexing anchor record. It is 5 bytes in length starting at byte 30.
- The fourth and fifth fields are literal (constant) values. The fourth field is specified in ASCII, while the fifth field is specified as hexadecimal (for example, EBCDIC) data.

For more information about using data values or literal values for indexing, refer to “Indexing with Data Values” on page 14, and “Indexing with Literal Values” on page 14.

ACIF allows fields to be defined but never referenced as part of an index. Because ACIF requires either a field or **trigger** to appear on the first page of a group (for example, a bank statement), you can satisfy this requirement by defining a “dummy” field. This dummy field allows ACIF to determine the beginning page of a group, but it is not used as part of an index.

fileformat={**record** | **record,n** | **stream** | **stream,(newline=X'nn')**}

Specifies the format of the input file. If you do not specify **fileformat**, the **acif** command uses **stream** as the default.

The **fileformat** parameter does not apply to input files that are resources. Resource files are in MO:DCA-P or AFP data stream format, and the **acif** command automatically determines that the file is a resource. Values are:

- record** The input file is formatted in S/370 record format, where the first two bytes of each line specify the length of the line. Files with **record** format typically are MVS or VM files that have a variable record format, and that are NFS-mounted to AIX.
- record,n** The input file is formatted in such a way that each record (including AFP data stream and MO:DCA-P records) is a fixed length, *n* bytes long. The value of *n* is a number from 1 to 32767. The encapsulated size of the AFP structured field must be less than the size of *n*. Files with **record,n** format typically come with fixed-length file attributes from a S/370 host system like MVS or VM.
- stream** The input file has no length information; it is a stream of data separated by a newline character. The AFP portion of the input file has its length information encapsulated in the structured field. Files with **stream** format typically

come from a workstation operating system like AIX, OS/2, or DOS.

The **acif** command examines the first six bytes of the first line data record of the input file, to determine whether the input file is ASCII or EBCDIC. If ACIF determines that the input file is ASCII, ACIF looks for the ASCII newline character (X'0A') to delimit the end of a record. If ACIF determines that the input file is EBCDIC, ACIF looks for the EBCDIC newline character (X'25') to delimit the end of a record. If the input record is MO:DCA-P, no newline character is required. The **acif** command does not include newline characters in the MO:DCA-P data stream that ACIF produces.

stream,(*newline=X'nn'*)

newline is an optional subparameter of **fileformat** that is used only with the **stream** parameter. You use **newline** to specify a hexadecimal value for the newline character in the input data file.

You can use **newline** when ACIF's algorithm cannot determine the correct newline character (if blanks are at the beginning of the file, for instance). Or you can use **newline** if you want to specify a newline character that is not the standard default. For example, you could use **newline** as follows:

```
fileformat=stream,(newline=X'0D')
```

If **newline** is not specified, ACIF uses the algorithm specified under **fileformat=stream**.

fontlib=*pathlist*

Specifies the directories in which fonts are stored. The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths. The **acif** command searches the paths in the order in which they are specified.

The **acif** command searches for the fonts in the following order:

1. The paths you specified with **userlib**, if any
2. The paths you specified with **fontlib**, if any
3. The paths you specified with **reslib**, if any
4. The paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**
6. The directory **/usr/lpp/afpfonts**
7. The directory **/usr/lpp/psf/fontlib**

For information on how PSF for AIX selects resources, refer to *IBM Print Services Facility for AIX: Print Administration*.

formdef=*fdefname*

Specifies the file name of the form definition. A form definition defines how a page of data is placed on a form, the number of copies of a page, any modifications to that group of copies, the paper source, and duplexing. The form definition is actually used at print time, not at transform time. The value is:

fdefname Any valid form definition file name. The *fdefname* can be one to eight alphanumeric or national characters, including the two-character prefix, if there is one. The *fdefname* is case-sensitive.

Note: If the file name of the form definition includes a file extension, do not use the file extension when specifying the form definition. For example, to use a form definition named **memo.FDEF38PP**, specify `formdef=memo`.

The **acif** command requires a form definition in order to process the input file (even though the form definition actually gets used at print time). If you do not specify **formdef=**, or if you specify **formdef=** without a form definition file name, the **acif** command will not work.

The form definition you use may be located:

- In an AIX directory
- Inline in the file (that is, within the file itself)

If the form definition file is in an AIX directory, use the **userlib** parameter or **fdeflib** parameter to specify the path to the file. For example:

```
formdef=memo userlib=/usr/afp/resources
```

or

```
formdef=memo fdeflib=/usr/lib/formdefns
```

If the form definition is an inline resource, you must do the following:

- Specify **cc=yes** to indicate that the file contains carriage-control characters.
- Specify **formdef=fdefname**, where *fdefname* is the name of the inline form definition; or specify **formdef=dummy**.

If you specify **formdef=dummy** but the file does not include an inline form definition, the **acif** command looks for the form definition named **dummy**.

If the name specified for the **formdef** parameter does not match the name of an inline form definition, the **acif** command looks for the form definition in the **formdef** search path.

An input file can contain multiple form definitions, but only one form definition can be used for printing. If a file contains more than one inline form definition, and you specify **formdef=fdefname**, ACIF uses the first inline form definition named *fdefname*. If a file contains more than one inline form definition, and you specify **formdef=dummy**, ACIF uses the first inline form definition in the input file.

groupname={index1 | indexn}

Specifies which of the eight possible **index** values should be used as the group name for each index group. Using a unique index value for the group name is recommended. The intent is to have a unique group name for every group ACIF produces in the output file. The value includes the **field** definitions from the **index** parameter but not the attribute name. ACIF uses this parameter only when the file is

indexed. Workbench Viewer displays this value along with the attribute name and index value. You can use the group name to select a group of pages to be viewed. Values are:

index1 ACIF uses the value of **index1**.

index n ACIF uses the value of the specified **index** (**index1**, **index2**, **index3**,...**index8**).

imageout={asis | ioca}

Specifies the format of the image data produced by the **acif** command in the output document. Values are:

asis The **acif** command produces all image data in the same format that it was in the input file.

ioca The **acif** command produces all image data in the Image Object Content Architecture uncompressed format.

index n ={ 'attribute name' | X'attribute name' },field n [,field n ...]

Specifies the content of the indexing tags for the entire file. A maximum of eight indexes can be defined (**index1**, **index2**,... **index8**), and each index can be made up of one or more **field** definitions.

If literal values are specified for every index, ACIF treats the entire file as one page group and uses this information to index the document. ACIF reports an error condition and terminates processing if literal values are specified for all **index n** parameters and if any **trigger n** parameters are also specified.

For **field** parameters that specify data values within the file, ACIF determines the actual location of the indexing information based on the indexing anchor record, set by the **trigger** parameters.

A valid set of index parameters comprises either of the following:

- **field** definitions containing only constant data (literal values)
- **field** definitions containing both constant data and application data (data fields in the print file)

You can also specify the same **field** parameters in more than one **index** parameter.

Note: If one or more **trigger n** parameters are specified (that is, ACIF will index the file), at least one **index n** parameter must be specified, and that index must comprise at least one **field n** parameter value that is not a literal. ACIF reports an error condition and terminates processing if this rule is not satisfied.

Valid components of the **index n** parameter are:

'attribute name' | X'attribute name'

Specifies a user-defined attribute name to be associated with the actual index value. For example, assume **index1** is a person's bank account number. The string 'Account Number' would be a meaningful attribute name. The value of **index1** would be the account number (for example, 1234567). Think of the attribute name as a label for the actual index value. This value can be specified either as character data or hexadecimal

data. (If the input file is *anything other than* ASCII, then the value *must* be specified as hexadecimal data.) The attribute name is a string from 1 to 250 bytes in length. ACIF does not perform any validity checking on the contents of the attribute name.

field*n*[,field*n*...]

Specifies one or more **field*n*** parameters that compose the index value. A maximum of 16 **field*n*** parameters can be specified. If more than one **field*n*** parameter is specified, ACIF concatenates them into one physical string of data. No delimiters are used between the concatenated fields. Because an index value has a maximum length of 250 bytes, the total of all specified **field*n*** parameters for a single index cannot exceed this length. ACIF reports an error condition and terminates processing if this occurs.

For example, if you want to specify the following fields and indexes, you enter:

```
field1='1234567'  
field2=0,10,20  
field3=0,25,20  
index1='Patent Number',field1  
index2='Employee Name',field2,field3
```

The example above specifies that the first index tag is made up of the literal character string '1234567', while the other two index tags are made up of fields within the file records.

```
field1='123456'  
field2='444556677'  
index1='Account Number',field1  
index2='Social Security Number',field2
```

This example specifies both index tags as literal values. The entire file will be indexed using these two values. The resulting index object file contains only one record in this case.

Note: The preceding examples are based on ASCII input data. If the input data were *not* ASCII, then in ACIF for AIX the literal values used in these examples would be expressed in hexadecimal strings. For an example using hexadecimal strings, see “Specifying ACIF Processing Parameters for EBCDIC Input Data” on page 58.

indexdd={ INDEX | *filename*}

Specifies the name or the full path name for the index object file. When ACIF is indexing the file, it writes indexing information in the file with this name. If you specify the file name without a path, ACIF puts the index object file into your current directory. Values are:

INDEX ACIF uses **INDEX** as the name for the index object file.

filename A character string containing only those alphanumeric characters supported in AIX file names.

indexobj={ group | all | none}

Specifies the type of information ACIF puts in the index object file. Values are:

- group** Places only group-level entries into the index object file, which saves space.
- all** Places both page-level and group-level entries into the index object file. Select **all** if you are indexing a file for use with the Workbench Viewer.
- none** Suppresses the collection of all index-level information. Choose **none** if you do not require an external index file. Choosing **none** will also reduce ACIF storage requirements.

indexstartby={ 1 | nn}

Specifies the output page number by which ACIF must find an indexing field, if ACIF is indexing the file. Values are:

- 1** ACIF must find an index on the first page.
- nn* Specifies the output page number by which ACIF must find the index criteria specified.

This parameter is helpful if, for example, your file contains header pages. If your file contains two header pages, you can specify a page number one greater than the number of header pages (**indexstartby=3**).

If ACIF does not find an indexing field before the page number specified in the **indexstartby** parameter, it issues a message and stops processing.

indexit=programname

Specifies the name or the full path name of the index record exit program. This is the program ACIF calls for every record (structured field) it writes in the index object file (**indexdd**). If you specify the file name without a path, ACIF searches for the exit program in the paths specified by the **PATH** environment variable. If this parameter is not specified, ACIF will not use an index record exit program. The value is:

programname

Any valid index record exit program name. The exit program name is case-sensitive.

inpexit=programname

Specifies the name or the full path name of the input record exit program. The **acif** command calls this program for every record (every line) it reads from the input file (**inputdd**). If you specify the file name without a path, the **acif** command searches for the exit program in the paths specified by the **PATH** environment variable. If you do not specify this parameter, the **acif** command will not use an input record exit program. The value is:

programname

Any valid input record exit program name. The exit program name is case-sensitive.

If the input file is unformatted ASCII, but the fonts you are using contain EBCDIC, not ASCII, code points (for example, you specify **chars=GT15**), you can specify one of the following exit programs supplied with PSF for AIX:

/usr/lpp/psf/bin/apka2e

Converts ASCII stream data to EBCDIC stream data.

/usr/lpp/psf/bin/asciinp

Converts unformatted ASCII data that contains carriage returns and form feeds into a record format that contains an American National Standards Institute (ANSI) carriage control character. This exit encodes the ANSI carriage control character in byte 0 of every record.

/usr/lpp/psf/bin/asciinpe

Converts unformatted ASCII data into a record format in the same way as **asciinp**, and then converts the ASCII stream data to EBCDIC stream data.

If your input file uses fonts that have ASCII code points (for example, you specify **chars=H292**, or any of the fonts listed in Chapter 5, “IBM AFP Fonts for ASCII Data” on page 77) you should *not* use the **apka2e** or **asciinpe** exit programs. However, if your unformatted ASCII file contains carriage returns and form feeds, you may want to specify the following exit program supplied with PSF for AIX:

inputdd=filename

Specifies the full path name of the input file that the **acif** command will process. If you do not specify **inputdd**, the **acif** command uses standard input.

msgdd=filename

Specifies the name or the full path name of the file where the **acif** command writes error messages. If you specify the file name without a path, the **acif** command puts the error file into your current directory. If you do not specify **msgdd**, the **acif** command uses standard error for its message output.

objconlib=pathlist

Specifies the directories in which setup files are stored. A COM setup file consists of a MO:DCA structure called an object container. For further information about the COM setup file see page 32.

The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths. The **acif** command searches the paths in the order in which they are specified.

The **acif** command searches for a setup file in the following order:

1. The paths you specified with **userlib**, if any
2. The paths you specified with **objconlib**, if any
3. The paths specified in **reslib**, if any
4. The paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**

outexit=programname

Specifies the name or the full path name of the output record exit program. The **acif** command calls this program for every output record (every line) it writes to the output document file (**outputdd**). If you specify the file name without a path, the **acif** command searches for the file name in the paths specified by the **PATH** environment variable. If you do not specify this parameter, the **acif** command will not use an output record exit program. The value is:

programname

Any valid output record exit program name. The exit program name is case-sensitive.

outputdd=filename

Specifies the name or the full path name of the output document file. If you specify the file name without a path, the **acif** command puts the output file into your current directory. If you do not specify **outputdd**, the **acif** command writes the output to standard output.

ovlylib=pathlist

Specifies the directories in which overlays are stored. The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths. The **acif** command searches the paths in the order in which they are specified.

The **acif** command searches for an overlay in the following order:

1. The paths you specified with **userlib**, if any
2. The paths you specified with **ovlylib**, if any
3. The paths specified in **reslib**, if any
4. The paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**

You should specify the same value for the **ovlylib** parameter to the **acif** command as specified to PSF for AIX. In this way, the search paths and resources used at transform time are identical to the search paths and resources used at print time. For information on how PSF for AIX selects resources, refer to *IBM Print Services Facility for AIX: Print Administration*.

pagedef=pdefname

Specifies the file name of the page definition. A page definition defines the page format that the **acif** command uses to compose the input file into pages. The page definition is actually used at transform time, not at print time. The value is:

pdefname Any valid page definition file name. The *pdefname* can be one to eight alphanumeric or national characters, including the two-character prefix, if there is one. The *pdefname* is case-sensitive.

Note: If the file name of the page definition includes a file extension, do not use the file extension when specifying the page definition. For example, to use a page definition named **memo.PDEF38PP**, specify **pagedef=memo**.

The **acif** command does not require a page definition when indexing an AFP data stream file. However, ACIF does require a page definition to transform an input file that contains S/370 line data, mixed-mode data, or unformatted ASCII data into MO:DCA-P. If you are transforming such an input file, and you do not specify **pagedef=**, or if you specify **pagedef=** without a page definition file name, the **acif** command will not work.

The page definition you use may be located:

- In an AIX directory
- Inline in the file (that is, within the file itself)

If the page definition file is in an AIX directory, use the **userlib** parameter or **pdeflib** parameter to specify the path to the file. For example:

```
pagedef=memo userlib=/usr/afp/resources
```

or

```
pagedef=memo pdeflib=/usr/lib/pagedefns
```

If the page definition is an inline resource, you must do the following:

- Specify **cc=yes** to indicate that the file contains carriage-control characters.
- Specify **pagedef=pdefname**, where *pdefname* is the name of the inline page definition; or specify **pagedef=dummy**.

If you specify **pagedef=dummy** but the file does not include an inline page definition, the **acif** command looks for the page definition named **dummy**.

If you use **pagedef** to specify an inline page definition that is different than the actual page definition used inline, the **acif** command looks for the page definition in the page definition search path instead of the inline page definition.

An input file can contain multiple page definitions, but only one page definition can be used by the **acif** command. If a file contains more than one inline page definition, and you specify **pagedef=pdefname**, ACIF uses the first inline page definition named *pdefname*. If a file contains more than one inline page definition, and you specify **pagedef=dummy**, the **acif** command uses the first inline page definition in the input file.

parmdd=filename

Specifies the name or the full path name of a file that contains the **acif** parameters and their values. If you specify the file name without a path, the **acif** command searches for the file name in your current directory.

You may find it convenient to put the **acif** parameters and values into a file, so that you do not have to type all of them on the command line whenever you use ACIF. Values are:

filename The name of the file containing **acif** command parameters and values.

Notes:

1. The beginning delimiter for comments is “/*.” For example:

```
formdef=F1TEMP /* Temporary formdef
```

Comments can appear anywhere, but the **acif** command ignores all information in the line following the “/*” character string. Also, although the ending delimiter (“*/”) is shown in some examples, it is not a required entry for ACIF.

2. Each parameter must be on a separate line. For example:

```
chars=GT10 cctype=a /* This is not allowed.
```

pdeflib=*pathlist*

Specifies the directories in which page definitions are stored. The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths. The **acif** command searches the paths in the order in which they are specified.

The **acif** command searches for a page definition in the following order:

1. The paths you specified with **userlib**, if any
2. The paths you specified with **pdeflib**, if any
3. The paths specified in **reslib**, if any
4. The paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**

prmode={SOSI1 | SOSI2 | aaaaaaa}

Specifies the type of data in the input file and whether the **acif** command must perform optional processing of that data. Values are:

SOSI1 Specifies that each shift-out, shift-in code be converted to a blank and a Set Coded Font Local text control. The **SOSI1** data conversion is the same as the **SOSI1** data conversion performed by PSF/MVS, PSF/VM, and PSF/VSE.

SOSI2 Specifies that each shift-out, shift-in code be converted to a Set Coded Font Local text control. The **SOSI2** data conversion is the same as the **SOSI2** data conversion performed by PSF/MVS, PSF/VM, and PSF/VSE.

aaaaaaa Any eight-byte alphanumeric string. This value is supplied to all of the ACIF user exits.

For either **SOSI** process to work correctly, the first font specified in the **chars** parameter (or in a font list in a page definition) must be a single-byte font, and the second font must be a double-byte font.

pseglib=*pathlist*

Specifies the directories in which page segments are stored. The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths. The **acif** command searches the paths in the order in which they are specified.

The **acif** command searches for page segments in the following order:

1. The paths you specified with **userlib**, if any
2. The paths you specified with **pseglib**, if any
3. The paths specified in **reslib**, if any
4. The paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**

You should specify the same value for the **pseglib** parameter to the **acif** command as specified to PSF for AIX. In this way, the search paths and resources used at transform time are identical to the search paths and resources used at print time. For information on how PSF for AIX selects resources, refer to *IBM Print Services Facility for AIX: Print Submission*.

resexit=programname

Specifies the name or the full path name of the resource exit program. This is the program the **acif** command calls each time it attempts to retrieve a requested resource from a directory. If you specify the file name without a path, the **acif** command searches for the file name in the paths specified by the **PATH** environment variable. If you do not specify this parameter, the **acif** command does not use a resource exit program. The exit program name is case-sensitive. The value is:

programname

Any valid resource exit program name.

reslib=pathlist

Specifies the paths for the system resource directories. System resource directories typically contain resources that are shared by many users. The directories can contain any AFP resources (fonts, page segments, overlays, page definitions, or form definitions).

In most cases, you will want the **acif** command to find the same resources that PSF for AIX uses when it prints the file. If so, the **reslib** paths should be the same as the paths specified with the **respath** parameter to PSF for AIX.

The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths.

The **acif** command searches for resources in the following order:

1. Paths specified by the **userlib** parameter
2. Paths specified by the **fdeflib**, **fontlib**, **pdeflib**, **pseglib**, **objconlib**, and **ovlylib** parameters for specific types of resources
3. Paths specified by the **reslib** parameter
4. Paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**
6. The directory **/usr/lpp/afpfonts**
7. The directory **/usr/lpp/psf/fontlib**

resobjdd={ RESOBJ | *filename*}

Specifies the name or the full path name for the resource file produced by ACIF. If you specify the file name without a path, ACIF puts the resource file into your current directory. When ACIF processes a print file, it can optionally create a file containing all or some of the resources required to print or view the file. ACIF writes the resource data in a file with this name. Values are:

RESOBJ ACIF writes the resource data in a file with this name.

filename A character string containing only those alphanumeric characters supported AIX file names.

restype={ none | [fdef][pseg][ovly][font] [objcon] | all}

Specifies the type of AFP print resources ACIF should retrieve from the resource directories for inclusion in the resource file (**resobjdd**). Values are:

none Specifies that no resource file is to be created.

fdef Specifies that the form definition (**formdef**) used in processing the file will be included in the resource file.

pseg Specifies that all page segments required to print or view the output document file will be included in the resource file.

ovly Specifies that all overlays required to print or view the output document file will be included in the resource file.

font Specifies that all font character sets and code pages required to print or view the output file will be included in the resource file.

Note: Coded fonts are never included in the resource file.

objcon Specifies that all object container files requested by the input data stream (including the one specified by the **comsetup** parameter) be included in the resource file.

all Specifies that all resources required to print or view the output document file (**outputdd**) will be included in the resource file (**resobjdd**).

(See the **reslib** parameter for a description of how ACIF searches for resources.)

ACIF supports the specification of **fdef**, **font**, **objcon**, **ovly**, and **pseg** in any combination. For example, if you want to specify form definitions, page segments, and overlays as the resource types, you would enter:

```
restype=fdef,pseg,ovly
```

Because Workbench Viewer does not use AFP raster fonts when presenting the data on the screen, you may want to specify **restype=fdef,ovly,pseg** to prevent fonts from being included in the resource file. This reduces the number of bytes transmitted when the file is transferred to the workstation.

Note: Keep in mind that if you have a resource type that you want saved in a resource file, and it is included in another resource type, you must specify both resource types. For example, if you request that just page segments be saved in a resource file, and the page segments are included in overlays, the page segments will not be saved in the resource file, because the overlays will not be searched. In this case, you would have to request that both page segments and overlays be saved.

trc={yes | no}

Specifies whether the input file contains Table Reference Characters (TRCs). Some applications may produce output that uses different fonts on different lines of a file by specifying TRCs at the beginning of each line after the carriage-control character, if one is present. Values are:

yes The input file contains table reference characters.

no The input file does not contain table reference characters.

Consider the following when you use TRCs:

- The order in which the fonts are specified in the **chars** parameter establishes which number is assigned to each associated TRC. For example, the first font specified is assigned 0, the second font 1, and so on.
- If you specify **trc=yes** but TRCs are not contained in the file, the **acif** command interprets the first character (or second, if carriage-control characters are used) of each line as the font identifier. Consequently, the font used to process each line of the file may not be the one you expect, and one byte of data will be lost from each line.
- If you specify **trc=no** or you do not specify **trc** at all, but your data contains a TRC as the first character (or second if carriage-control characters are used) of each line, the **acif** command interprets the TRC as a text character in the processed output, rather than using it as a font identifier.

triggern={record | *},{column | *},{'trigger value' | X'trigger value'}

Specifies the locations and values of data fields within the input file that are to be used to define indexing groups in the file. These data fields are referred to as “triggers” because their presence in the file triggers a processing action. A maximum of four **trigger** parameters can be specified. The number of **triggern** parameters required to uniquely identify the beginning of a group of pages within the file is a function of the complexity of the application output. **trigger1** is special and each record in the file containing this value is referred to as an indexing anchor record. The presence of a **trigger** parameter causes ACIF to index the input file. Each **triggern** parameter comprises three values:

*record | ** Specifies the relative record number from the indexing anchor record (that is, **trigger1**). A value of ****** can only be specified in **trigger1**; this value indicates that every record should be checked for the presence of the **trigger1** value. After the **trigger1** value has been found,

all other **trigger n** parameter values are specified as a relative offset from **trigger1**. ACIF reports an error condition and terminates processing if an '*' is specified with any **trigger n** parameter other than **trigger1**. The supported range of values for *record* is 0 to 255.

column | * Specifies the byte offset from the beginning of the record where the trigger value is located. This value can be specified in absolute terms (for example, 10) or as '*', which results in ACIF scanning the record from left to right looking for the trigger value. A value of 1 refers to the first byte in the record. For files containing carriage-control characters, column 1 refers to the carriage-control. The supported range of values for *column* are 1 to 32756. ACIF compares the trigger value to the input data. If the specified value exceeds the physical length of the record, ACIF considers the comparison "false" and continues processing.

'trigger value' | X'trigger value'

Specifies the actual alphanumeric (case-sensitive) or hexadecimal value of the trigger. ACIF does not perform any validity checking on this value, but uses it in performing a byte-for-byte comparison with the records in the file. The trigger value can be 1–253 bytes in length. If the combined values of *column* and the trigger length exceed the physical length of the record, ACIF considers the comparison "false" and continues processing.

For example, to use a carriage-control character as a trigger, you enter:

```
trigger1=*,1,'1'           /* Look for Skip-to-Channel 1
trigger2=0,50,'ACCOUNT:'   /* Find account number
trigger3=3,75,'PAGE 1'     /* Find page 1
```

In this example, **trigger1** instructs ACIF to scan every record, looking for the occurrence of '1' in the first byte. After ACIF locates a record containing the '1', it looks in the same record, starting at byte 50, for the occurrence of 'ACCOUNT:'. If 'ACCOUNT:' is found, ACIF looks at the third record down for a value of 'PAGE 1', starting at byte 75. If 'PAGE 1' is found, ACIF defines the record containing **trigger1** as the indexing anchor record, and all indexing information is specified as relative locations relative from this point.

If ACIF finds either 'ACCOUNT:' or 'PAGE 1', ACIF begins scanning the first record after the farthest field specified. If either 'ACCOUNT:' or 'PAGE 1' are not found at their specified locations relative to **trigger1**, ACIF begins looking for **trigger1** again, starting with the next record (that is, the current record containing **trigger1** + 1).

Notes:

1. ACIF requires that at least one **trigger n** or **field n** value appear within the page range specified by the **indexstartby** parameter. If no **trigger n** or **field n** parameter is satisfied within the **indexstartby** page range, ACIF stops processing.
2. At least one **trigger n** or **field n** must exist on the first page of every unique page group. ACIF cannot detect an error condition if **trigger n** or **field n** is missing, but the output may be incorrectly indexed.
3. **trigger1** must be specified when ACIF is requested to index the file.
4. An error condition will occur if you specify any **trigger** parameters when the input file contains indexing tags.

uniquebngs={yes | no}

Specifies whether ACIF creates a unique group name by generating an 8-character numeric string and appending the string to the group name. The group name defined in the Begin Named Page Group (BNG) structured field is comprised of an index value and a sequence number.

yes Specifies that ACIF generate an 8-character numeric string and append the string to the group name.

no ACIF does not generate the string. No is the default if you specify **dcfpagenames=yes**. Specify **no** if you use the AFP API to generate your own group names.

userlib=pathlist

Specifies the names of user directories containing AFP resources for processing the input file. The directories can contain any AFP resources (fonts, page segments, overlays, page definitions, form definitions, or COM setup files).

By convention, these resources are typically used by one user, as opposed to the system resources (specified with the **reslib** parameter) that are shared by many users. Therefore, you should use the **userlib** parameter to specify resources that are not retrieved with the **fdeflib**, **fontlib**, **objconlib**, **ovlylib**, **pdeflib**, or **pseglib** parameters. The value is:

pathlist Any valid search path. You must use a colon (:) to separate multiple paths.

The **acif** command searches for resources in the following order:

1. Paths specified by the **userlib** parameter
2. Paths specified by the **fdeflib**, **fontlib**, **objconlib**, **ovlylib**, **pdeflib**, **pseglib** parameters for specific types of resources
3. Paths specified by the **reslib** parameter
4. Paths specified by the **PSFPATH** environment variable
5. The directory **/usr/lpp/psf/reslib**
6. The directory **/usr/lpp/afpfonts**
7. The directory **/usr/lpp/psf/fontlib**

Examples

The examples contained in this section show how to use ACIF processing parameters for conversion, resource retrieval, specifying fonts, and identifying the location of resource directories. For indexing examples, see Chapter 3, “Example of an ACIF Application in AIX” on page 53. For examples of how to use ACIF or the **line2afp** command to transform S/370 line data and unformatted ASCII files for printing with PSF for AIX, refer to “Using the line2afp Command” in *IBM Print Services Facility for AIX: Print Submission*.

Note: In the following examples, ACIF is run by entering the **acif** command, parameters, and values on the command line. When all of the parameters will not fit on a single line across the screen, the backslash, \, tells AIX to continue reading the command from the next line.

1. You have an EBCDIC S/370 line data file named 0LDFILE.1403 that you want to transform into a MO:DCA-P document named NEWFILE.afp. To do this, enter:

```
acif inputdd=0LDFILE.1403 outputdd=NEWFILE.afp cctype=a \  
fileformat=record pagedef=P1A06462 formdef=F1A10110
```

ACIF converts the S/370 line data file, specified with the **inputdd** parameter, into a document file with the name specified by the **outputdd** parameter.

You specified **cctype=a** to indicate that the file contains EBCDIC ANSI carriage control characters. This particular input file is in S/370 variable length record format, so you indicated this by specifying **fileformat=record**. The **pagedef** and **formdef** parameters are required with your line data input file, so you specified the file names of the page definition and form definition you want ACIF to use in processing this file.

2. You have an AFP file (MYFILE) that contains page segments and overlays. You would like to retrieve the page segments and overlays from the file and create both a data file and a resource file. To do this, enter:

```
acif inputdd=MYFILE outputdd=MYDATA resobjdd=MYRES \  
restype=pseg,ovly,fdef formdef=F1H10110
```

From this job, ACIF will produce an AFP document file and a resource file. The AFP document file (MYDATA) will contain the AFP data from MYFILE. The resource file (MYRES) will contain the resource data from MYFILE.

You specified **restype=pseg,ovly,fdef** so that the page segments and overlays would be included in the resource file, along with the form definition (specified with the **formdef** parameter) that you want ACIF to use when processing the file.

For more information about when and why you would want to use ACIF's resource retrieval functions, refer to “Retrieving Resources” on page 16.

3. You have an input file (MYFILE.asc) that contains unformatted ASCII data, and you want three coded fonts to be used in processing the file: Helvetica 12-point, and Times New Roman 10-point and 9-point. You are using a page definition supplied with PSF for AIX (P1A6462), and the page definition does not name any fonts. To use the three fonts, specify the following:

```
acif inputdd=MYFILE.asc outputdd=MYFILE.afp chars=H2B2,N202,N292 \  
trc=yes pagedef=P1A06462 formdef=F1A10110
```

You specified the font names with the **chars** parameter. Because you need to use fonts with the appropriate ASCII code points for your unformatted ASCII input, you referred to Chapter 5, “IBM AFP Fonts for ASCII Data” on page 77. There you found that the IBM Core Interchange Font name is X0H230B2 for Helvetica 12-point, X0N23002 for Times New Roman 10-point, and X0N23092 for Times New Roman 9-point. Because the **chars** parameter limits the specification of a font name to four characters, you used the corresponding short name from the table for each of the three fonts, without eliminating the 2-character coded font prefix (X0). Because table reference characters are required when you want the file to print with more than one font, you specified **trc=yes**.

Your input and output file names are specified with the **inputdd** and **outputdd** parameters. The page definition and form definition you want ACIF to use when processing the file are specified with the **pagedef** and **formdef** parameters.

4. You have an input file and you want to use specific resources during processing. You want to use a form definition (FORMD1A) and an overlay that are stored in the general resource directory at your location (/usr/site/resdir). To be sure that ACIF finds the resources you want to use, specify the following:

```
acif inputdd=INFILE outputdd=OUTFILE \  
  pagedef=PAGED6B formdef=FORMD1A \  
  userlib=/usr/mystuff/art1:/usr/mystuff/art2 \  
  pdeflib=/usr/dept/pdefdir3 reslib=/usr/site/resdir
```

The page definition you want to use (PAGED6B) is stored in one of the several page definition directories used by your department (/usr/dept/pdefdir3). The page definition is a copy of one with the same file name that is stored in the site’s general resource directory, with some modifications made for use by your department. Your page segments are stored in two other directories that you have set up for your own use (/usr/mystuff/art1 and /usr/mystuff/art2). Because ACIF always searches the path specified by the **userlib** parameter first, your page segments will be found in your personal directories. ACIF next searches the paths specified by the parameters for specific resource libraries (**pdeflib**, **fdeflib**, and so forth), so ACIF will then find the page definition you want to use from the department’s directory. ACIF will then search the path specified with the **reslib** parameter, finding your form definition and your overlay. ACIF will *not* use the page definition named PAGED6B that is stored in the /usr/site/resdir directory, because it will already have found the modified PAGED6B in the department directory specified with the **pdeflib** parameter.

Implementation Specifics

The **acif** command is part of PSF for AIX, and is installed with the **psf.acif** option.

Files

/usr/lpp/psf/bin/acif

The executable program (the **acif** command)

/usr/lpp/psf/acif/apkinp.c, apkind.c, apkres.c, apkout.c, apka2e.c, asciinp.c, asciinpe.c

Sample ACIF user exits

/usr/lpp/psf/bin/apka2e, apkinp, apkind, apkres, apkout, apka2e, asciinp, asciinpe

Sample user exit executables

/usr/lpp/psf/bin/Makefile

The build rules for the ACIF user exits, **apkinp**, **apkind**, **apkres**, **apkout**, **apka2e**, **asciinp**, and **asciinpe**

/usr/lpp/psf/acif/apkexits.h

C language header file for the ACIF user exits

NLS Messages

ACIF messages on the AIX platform may be written in any one of the following languages: Simplified Chinese, Traditional Chinese, English, French, French-Canadian, German, or Japanese. The message files can be found in:

```
/usr/lib/nls/msg/<country of choice>
```

Consult the description of the NLSPATH and LANG environment variables for information on setting these variables in an appropriate manner. The default message catalog is

```
/usr/lib/nls/msg/enUS/ACIF.cat
```

Suggested Reading

“Transforming Line Data for Printing with PSF” in *IBM Print Services Facility for AIX: Print Submission*.

“Form Definitions Supplied with PSF” and “Page Definitions Supplied with PSF” in *IBM Print Services Facility for AIX: Print Submission*.

IBM Page Printer Formatting Aid/6000: User's Guide Version 2.1 for information on how to create your own form definitions and page definitions.

Chapter 3. Example of an ACIF Application in AIX

The line-data application used as the example in this section is shown in Figure 15 on page 54. The application generates telephone bills. The objective is to make the billing application output available on customer service representatives' workstations. Then, when a customer calls with a billing inquiry, the representative can view the bill in the same format as the customer's printed copy.

To do this, you must convert the output from your application into a document format that can be used with the Viewer application of AFP Workbench (Workbench Viewer). You also must index the file to facilitate searching the file with Workbench Viewer. To ensure that all resources used in the bills are available at the workstation, you must use the resource retrieval function of ACIF.

Your tasks include:

1. Examining the input file to determine how to tag it for viewing
2. Specifying ACIF parameters for indexing and resource retrieval for either ASCII or EBCDIC input data
3. Identifying the locations of the resources used when the bills are printed
4. Determining the form definition and page definition used to print the bills
5. Running the ACIF job
6. Concatenating the index object file, the resource file, and the document file
7. Making the document file available to a workstation running Microsoft Windows for viewing with Workbench Viewer

Note: This example is hypothetical; an input file is *not* actually provided. The example is intended only to help you understand how ACIF may be used for an actual application, and to assist you when you use ACIF for your own application. For practical use, you must provide your own input file, and specify paths, directories, and so forth, as they apply to your particular installation and application.

The following topics are covered in this section:

- An example of an input file
- Specifying ACIF processing parameters for ASCII input data
- Specifying ACIF processing parameters for EBCDIC input data
- Identifying the locations of the specified resources
- Determining the form definition and page definition resources needed to format and print the ACIF job
- How to run the ACIF job
- The output files created by running the ACIF job
- Concatenating the output files
- Accessing the document file from a workstation for viewing

For an explanation of any parameter not specifically described in this section, see the description for that parameter in Chapter 2, "Using ACIF Parameters in AIX" on page 25.



<p>Make check payable to</p> 		<p><i>Return this portion with your payment.</i></p> <p>WILLIAM R. SMITH 5280 SUNSHINE CANYON DR BOULDER CO 80000-0000</p>																																											
		<p>TOTAL AMOUNT DUE: \$56.97 DATE DUE: JAN 29, 1993</p>																																											
<p>1 BASIC SERVICE \$30.56 2 LONG DISTANCE CHARGES \$26.41</p> <p style="text-align: right;">TOTAL \$56.97</p>																																													
		<p>BILL DATE: JAN 11, 1993 ACCOUNT NUMBER: 303-222-3456-6B</p>																																											
PREVIOUS BILL \$66.79	PAYMENT \$66.79	ADJUSTMENTS \$0.00	PAST DUE DISREGARD IF PAID \$0.00																																										
THANK YOU FOR YOUR PAYMENT			CURRENT CHARGES \$56.97																																										
			DATE DUE JAN 29, 1993																																										
			AMOUNT DUE \$56.97																																										
<p>SUMMARY OF CURRENT CHARGES</p> <p>RESIDENCE SERVICE \$25.07 911 SURCHARGE \$0.50 CUSTOMER ACCESS SERVICE \$3.50 WIRING MAINTENANCE PLAN \$0.50 FEDERAL EXCISE TAX \$0.50 STATE TAX \$0.49</p> <p>LONG DISTANCE CHARGES (ITEMIZED BELOW) \$26.41</p>																																													
<p>LONG DISTANCE CHARGES</p> <table border="1"> <thead> <tr> <th>NO.</th> <th>DATE</th> <th>TIME</th> <th>TO PLACE</th> <th>TO AREA NUMBER</th> <th>MINUTES</th> <th>AMOUNT</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>DEC 11</td> <td>7:15P</td> <td>LOVELAND CO</td> <td>303 666-7777</td> <td>006</td> <td>\$0.82</td> </tr> <tr> <td>2</td> <td>DEC 15</td> <td>9:16A</td> <td>NIWOT CO</td> <td>303 555-6666</td> <td>012</td> <td>\$1.56</td> </tr> <tr> <td>3</td> <td>DEC 24</td> <td>9:32P</td> <td>SANTA BARBARA CA</td> <td>805 999-2222</td> <td>032</td> <td>\$15.80</td> </tr> <tr> <td>4</td> <td>DEC 25</td> <td>2:18P</td> <td>LAS VEGAS NV</td> <td>702 888-7654</td> <td>015</td> <td>\$8.23</td> </tr> <tr> <td colspan="6" style="text-align: right;">TOTAL</td> <td>\$26.41</td> </tr> </tbody> </table>				NO.	DATE	TIME	TO PLACE	TO AREA NUMBER	MINUTES	AMOUNT	1	DEC 11	7:15P	LOVELAND CO	303 666-7777	006	\$0.82	2	DEC 15	9:16A	NIWOT CO	303 555-6666	012	\$1.56	3	DEC 24	9:32P	SANTA BARBARA CA	805 999-2222	032	\$15.80	4	DEC 25	2:18P	LAS VEGAS NV	702 888-7654	015	\$8.23	TOTAL						\$26.41
NO.	DATE	TIME	TO PLACE	TO AREA NUMBER	MINUTES	AMOUNT																																							
1	DEC 11	7:15P	LOVELAND CO	303 666-7777	006	\$0.82																																							
2	DEC 15	9:16A	NIWOT CO	303 555-6666	012	\$1.56																																							
3	DEC 24	9:32P	SANTA BARBARA CA	805 999-2222	032	\$15.80																																							
4	DEC 25	2:18P	LAS VEGAS NV	702 888-7654	015	\$8.23																																							
TOTAL						\$26.41																																							
PAGE 1																																													

Figure 15. Example of a Customer's Telephone Bill

The Input File

Figure 16 shows the line data file currently used to print the telephone bills.

Line	1	2	3	4	5	6	7	8	
0	1								
						WILLIAM R. SMITH			
						5280 SUNSHINE CANYON DR			
						BOULDER CO 80000-0000			
						TOTAL AMOUNT DUE: \$56.97			
						DATE DUE: JAN 29, 1993			
5	-								
	0	1 BASIC SERVICE.				\$30.56			
		2 LONG DISTANCE CHARGES							
					\$26.41			
10	0			TOTAL		\$56.97			
	0						BILL DATE: JAN 11, 1993		
							ACCOUNT NUMBER: 303-222-3456-6B		
15	-	\$66.79	\$66.79	\$0.00			\$0.00		
							\$56.97		
							JAN 29, 1993		
							\$56.97		
20	0	SUMMARY OF CURRENT CHARGES							
	0	RESIDENCE SERVICE					\$25.07		
		911 SURCHARGE					\$0.50		
		CUSTOMER ACCESS SERVICE					\$3.50		
		WIRING MAINTENANCE PLAN					\$0.50		
25		FEDERAL EXCISE TAX					\$0.50		
		STATE TAX					\$0.49		
		LONG DISTANCE CHARGES (ITEMIZED BELOW)					\$30.56		
	0	LONG DISTANCE CHARGES							
	0	NO.	DATE	TIME	TO PLACE	TO AREA NUMBER	MINUTES	AMOUNT	
30	0	1	DEC 11	7:15P	LOVELAND CO	303 666-7777	006	\$0.82	
		2	DEC 15	9:16A	NIWOT CO	303 555-6666	012	\$1.56	
		3	DEC 24	9:32P	SANTA BARBARA CA	805 999-6666	032	\$15.80	
		4	DEC 25	2:18P	LAS VEGAS NV	702 888-7654	015	\$8.23	
								TOTAL \$26.41	
35	-								
	0							PAGE 1	

Figure 16. Line-Data Telephone Bill

Specifying ACIF Processing Parameters for ASCII Input Data

You can process the ACIF parameters that are needed to produce the telephone bill for this example application by using one of the following methods:

- Create and specify a parameter file
- Enter the **acif** command, parameters, and values on the command line or in a shell script

This section describes the creation of a parameter file that can be used to process ACIF parameters when the input file is in ASCII.

Using a Parameter File with ASCII Input Data

A parameter file created for use with ASCII input data is shown in Figure 17. To use a parameter file, you specify the parameter file name with the **acif** command **parmdd** parameter. For example, to use a parameter file named PARMFILE, specify:

```
acif parmdd=PARMFILE
```

```
cc=yes
cctype=z
chars=42B2
cpgid=850
fdeflib=/usr/res/fdeflib1:/usr/res/fdeflib2
field1=13,66,15
field2=0,50,30
field3=1,50,30
field4=2,50,30
field5=4,60,12
fontlib=/usr/res/fontlib1:/usr/res/fontlib2
formdef=F1A10110
index1='Account Number',field1
index2='Name',field2
index3='Address',field3
index4='City, State, Zip',field4
index5='Date Due',field5
indexobj=all
indexdd=INDXOBJ
inputdd=/usr/data/INFILE
outputdd=OUTDOC
msgdd=acif.msg
ovlylib=/usr/res/ovlylib1:/usr/res/ovlylib2
pagedef=P1A08682
pdeflib=/usr/res/pdeflib1:/usr/res/pdeflib2
pseglib=/usr/res/pseglib1:/usr/res/pseglib2
resobjdd=RESDATA
restype=fdef,pseg,ovly
trigger1=*,1,'1'
trigger2=13,50,'ACCOUNT NUMBER'
```

```
/* example phone bill */
/* carriage control used */
/* ASCII ANSI carriage controls */
/* coded font */
/* code page identifier */
/* formdef directories */
/* Account Number data field */
/* Name data field */
/* Address data field */
/* City, State, Zip data field */
/* Date Due data field */
/* font directories */
/* formdef name */
/* 1st index attribute */
/* 2nd index attribute */
/* 3rd index attribute */
/* 4th index attribute */
/* 5th index attribute */
/* index object file entries */
/* index file name */
/* input path & file name */
/* output file name */
/* error message file name */
/* overlay directories */
/* pagedef name */
/* pagedef directories */
/* pseg directories */
/* resource file name */
/* resource type selection */
/* 1st trigger */
/* 2nd trigger */
```

Figure 17. Example of a Parameter File for ASCII Input Data

The example uses the following data values as the indexing attributes:

- Account Number
- Name
- Address
- City, State, Zip
- Date Due

The task is to specify the ACIF indexing parameters so that the first page of each bill includes group-level indexing tags containing the values of all five of these attributes.

To generate these indexing attributes, specify the **trigger1** parameter first, because ACIF always scans for the data specified in **trigger1** first. Because the data contains carriage control characters, including a carriage control character of 1 to indicate a new page, request that ACIF locate the start of a page by searching every record in the file for a trigger value of '1' in column 1 of the data. To do this, specify:

```
trigger1 = *,1,'1'
```

When ACIF finds a record that contains a '1' in column 1, that record becomes the indexing anchor record.

Subsequent **trigger** parameters are defined relative to the indexing anchor record. In this example, you want to ensure that the page being indexed is the first page of the bill, which is the only page in the bill that has the text 'ACCOUNT NUMBER' starting at byte 50 in the 13th record following the anchor record. To specify this additional trigger for locating the correct page to index, enter:

```
trigger2 = 13,50,'ACCOUNT NUMBER'
```

ACIF uses both trigger values to locate a place in the file to begin searching for the data supplied in the **index** parameters.

Next, specify the attribute name of the first indexing parameter as 'Account Number', and define the location of the attribute value in the data relative to the index anchor record set by **trigger1**. Because the data value for the Account Number attribute is located in the 13th record from the index anchor record starting in byte 66 and extending for 15 bytes, specify:

```
field1=13,66,15  
index1='Account Number',field1
```

To create the indexing tag for the Name attribute, define 'Name' as the indexing attribute. Locate the value for 'Name' in the anchor record in the data starting at byte 50 and extending for 30 bytes. The ACIF parameters to specify this are:

```
field2=0,50,30  
index2='Name',field2
```

Repeat this process to specify the other three indexing tags, so that the index attributes and values are defined as follows:

- index1='Account Number',field1
 - 'Account Number' is the 1st index attribute
 - field1 maps to the **field1** index value, which is:
 - 13 lines down from the indexing anchor record, 66 columns across, 15 bytes in length
- index2='Name',field2
 - 'Name' is the 2nd index attribute
 - field2 maps to the **field2** index value, which is:
 - 0 lines down (in the indexing anchor record), 50 columns across, 30 bytes in length
- index3='Address',field3
 - 'Address' is the 3rd index attribute
 - field3 maps to the **field3** index value, which is:
 - 1 line down from the indexing anchor record, 50 columns across, 30 bytes in length
- index4='City, State, Zip',field4
 - 'City, State, Zip' is the 4th index attribute
 - field4 maps to the **field4** index value, which is:
 - 2 lines down from the indexing anchor record, 50 columns across, 30 bytes in length

- `index5='Date Due',field5`
 - 'Date Due' is the 5th index attribute
 - `field5` maps to the **field5** index value, which is:
 - 4 lines down from the indexing anchor record, 60 columns across, 12 bytes in length

The result of using these indexing parameters is that the first page of each bill in the ACIF output file will contain indexing tags for each of the five indexing attributes. Using Workbench Viewer, customer service representatives can locate a single customer bill in the ACIF document using any combination of the indexing attributes.

Specifying ACIF Processing Parameters for EBCDIC Input Data

You can process the ACIF parameters that are needed to produce the telephone bill for this example application by using one of the following methods:

- Create and specify a parameter file
- Enter the **acif** command, parameters, and values on the command line or in a shell script

This section describes the creation of a parameter file that can be used to process ACIF parameters when the input file is in EBCDIC.

For the sake of this example, assume that the data was generated on a S/370, and is accessed via NFS. The disk where the input data resides is mounted as binary, which retains the data as EBCDIC. The input data is in variable length record format on the S/370 host, so a **fileformat** parameter, specifying `fileformat=record`, is required. (See the description of the **fileformat** parameter, on page 35.)

Using a Parameter File with EBCDIC Input Data

A parameter file created for use with EBCDIC input data is shown in Figure 18 on page 59. To use a parameter file, you specify the parameter file name with the **acif** command **parmdd** parameter. For example, to use a parameter file named `PARMFILE`, specify:

```
parmdd=PARMFILE
```

Note: Literal values used in the **field**, **index**, and **trigger** parameters must be expressed in hexadecimal strings when the input data is anything other than ASCII. In Figure 18, because the input data is EBCDIC, hexadecimal strings are entered which represent the literal values. For example, 'Name', which is the 2nd index attribute, for the 2nd data field, is represented as follows:

```
index2=X'D5819485',field2
```

```

cc=yes /* example phone bill */
cctype=a /* carriage control used */
chars=GT15 /* EBCDIC ANSI carriage controls */
cpgid=037 /* coded font */
fdeflib=/usr/res/fdeflib1:/usr/res/fdeflib2 /* code page identifier */
field1=13,66,15 /* formdef directories */
field2=0,50,30 /* Account Number data field */
field3=1,50,30 /* Name data field */
field4=2,50,30 /* Address data field */
field5=4,60,12 /* City, State, Zip data field */
fileformat=record /* Date Due data field */
fontlib=/usr/res/fontlib1:/usr/res/fontlib2 /* input file format */
formdef=F1A10110 /* font directories */
index1=X'C1838396A495A340D5A494828599',field1 /* formdef name */
index2=X'D5819485',field2 /* 1st index attr (Account Number) */
index3=X'C184849985A2A2',field3 /* 2nd index attr (Name) */
index4=X'C389A3A86B40E2A381A3856B40E98997',field4 /* 3rd index attr (Address) */
index5=X'C481A38540C4A485',field5 /* 4th index attr (City, State, Zip) */
indexobj=all /* 5th index attr (Date Due) */
indexdd=INDXOBJ /* index object file entries */
inputdd=/usr/data/INFILE /* index file name */
outputdd=OUTDOC /* input path & file name */
msgdd=acif.msg /* output file name */
ovlylib=/usr/res/ovlylib1:/usr/res/ovlylib2 /* error message file name */
pagedef=P1A08682 /* overlay directories */
pdeflib=/usr/res/pdeflib1:/usr/res/pdeflib2 /* pagedef name */
pseglib=/usr/res/pseglib1:/usr/res/pseglib2 /* pagedef directories */
resobjdd=RESDATA /* pseg directories */
restype=fdef,pseg,ovly /* resource file name */
trigger1=*,1,X'F1' /* resource type selection */
trigger2=13,50,X'C1C3C3D6E4D5E340D5E4D4C2C5D9' /* 1st trigger (1) */
/* 2nd trigger (ACCOUNT NUMBER) */

```

Figure 18. Example of a Parameter File for EBCDIC Input Data

The example uses the following data values as the indexing attributes:

- Account Number
- Name
- Address
- City, State, Zip
- Date Due

The task is to specify the ACIF indexing parameters so that the first page of each bill includes group-level indexing tags containing the values of all five of these attributes.

To generate these indexing attributes, specify the **trigger1** parameter first, because ACIF always scans for the data specified in **trigger1** first. Because the data contains carriage control characters, including a carriage control character of 1 to indicate a new page, request that ACIF locate the start of a page by searching every record in the file for a trigger value of the hexadecimal string for '1' in column 1 of the data. To do this, specify:

```
trigger1 = *,1,X'F1'
```

When ACIF finds a record that contains a '1' in column 1, that record becomes the indexing anchor record.

Subsequent **triggern** parameters are defined relative to the indexing anchor record. In this example, you want to ensure that the page being indexed is the first page of the bill, which is the only page in the bill that has the hexadecimal string for the text 'ACCOUNT NUMBER' starting at byte 50 in the 13th record following the anchor

record. To specify this additional trigger for locating the correct page to index, enter:

```
trigger2=13,50,X'C1C3C3D6E4D5E340D5E4D4C2C5D9'
```

ACIF uses both trigger values to locate a place in the file to begin searching for the data supplied in the **index** parameters.

Next, specify the attribute name of the first indexing parameter as the hexadecimal string for 'Account Number', and define the location of the attribute value in the data relative to the index anchor record set by **trigger1**. Because the data value for the Account Number attribute is located in the 13th record from the index anchor record starting in byte 66 and extending for 15 bytes, specify:

```
field1=13,66,15
```

```
index1=X'C1838396A495A340D5A494828599',field1
```

To create the indexing tag for the Name attribute, define the hexadecimal string for 'Name' as the indexing attribute. Locate the value for 'Name' in the anchor record in the data starting at byte 50 and extending for 30 bytes. The ACIF parameters to specify this are:

```
field2=0,50,30
```

```
index2=X'D5819485',field2
```

Repeat this process to specify the other three indexing tags, so that the index attributes and values are defined as follows:

- index1=X'C1838396A495A340D5A494828599',field1
 - X'C1838396A495A340D5A494828599' is 'Account Number', the 1st index attribute
 - field1 maps to the **field1** index value, which is:
 - 13 lines down from the indexing anchor record, 66 columns across, 15 bytes in length
- index2=X'D5819485',field2
 - X'D5819485' is 'Name', the 2nd index attribute
 - field2 maps to the **field2** index value, which is:
 - 0 lines down (in the indexing anchor record), 50 columns across, 30 bytes in length
- index3=X'C184849985A2A2',field3
 - X'C184849985A2A2' is 'Address', the 3rd index attribute
 - field3 maps to the **field3** index value, which is:
 - 1 line down from the indexing anchor record, 50 columns across, 30 bytes in length
- index4=X'C389A3A86B40E2A381A3856B40E98997',field4
 - X'C389A3A86B40E2A381A3856B40E98997' is 'City, State, Zip', the 4th index attribute
 - field4 maps to the **field4** index value, which is:
 - 2 lines down from the indexing anchor record, 50 columns across, 30 bytes in length
- index5=X'C481A38540C4A485',field5
 - X'C481A38540C4A485' is 'Date Due', the 5th index attribute
 - field5 maps to the **field5** index value, which is:

- 4 lines down from the indexing anchor record, 60 columns across, 12 bytes in length

The result of using these indexing parameters is that the first page of each bill in the ACIF output file will contain indexing tags for each of the five indexing attributes. Using Workbench Viewer, customer service representatives can locate a single customer bill in the ACIF document using any combination of the indexing attributes.

Using the Shell with EBCDIC Literal Values

Literal values used in the **field**, **index**, and **trigger** parameters must be expressed in hexadecimal strings when the input data is anything other than ASCII. Because the input data for this example is EBCDIC, hexadecimal strings are required, and *must* be entered if you specify your parameters within a parameter file. If the parameters are *not* specified in a parameter file, you can use AIX commands (such as **axeb** or **iconv**) to convert ASCII literal values into EBCDIC literal values. For example, to convert the ASCII literal 'Name', for the 2nd index attribute (**index2**), do the following:

1. Create a shell environment variable to hold the EBCDIC literal

- To do this using the AIX **axeb** command, enter:
attr2=\$(echo -n "Name" | axeb)
- To do this using the AIX **iconv** command, enter:
attr2=\$(echo -n "Name" | iconv -fIBM-850 -tIBM-037)

2. Then, on the command line or in a shell script, specify the 2nd index attribute by entering:

```
index2="'$attr2'",field2
```

Note: This example is for use with the Korn Shell (ksh). If you are using a different shell, refer to the documentation for the shell you are using in *AIX for RISC System/6000 Commands Reference*.

By using this method to convert the ASCII literals to the EBCDIC literals, no mistakes are made when converting the literals to a hexadecimal string.

Identifying the Locations of the Resources

To build the resource file, ACIF must know where to find the resources specified in the job. In the example, the following directories are defined:

fdeflib	Form definition directories, /usr/res/fdeflib1:/usr/res/fdeflib2
fontlib	Font directories, /usr/res/fontlib1:/usr/res/fontlib2
ovlylib	Overlay directories, /usr/res/ovlylib1:/usr/res/ovlylib2
pdeflib	Page definition directories, /usr/res/pdeflib1:/usr/res/pdeflib2
pseglib	Page segment directories, /usr/res/pseglib1:/usr/res/pseglib2

Determining the Form Definition and the Page Definition

To format and print the job, you need to specify page definition and form definition resources. In the example, the following resources are used:

formdef	F1A10110, a standard form definition, provided with PSF for AIX
pagedef	P1A08682, a standard page definition, provided with PSF for AIX

Running the ACIF Job

You can run the ACIF job using one of the following methods:

- Use a parameter file that contains the parameters and values needed for the application. To run the **acif** command with a parameter file named PARMFILE, you would enter:

```
acif parmdd=PARMFILE
```

- Enter the **acif** command, parameters and values on the command line or in a shell script. For the example application you would enter:

```
acif cc=yes cctype=z chars=42B2 cpgid=850... and so forth (continuing by entering all of the remaining parameters and values)
```

See “Examples” on page 50 for examples of running ACIF from the command line. For information on creating and running shell scripts, refer to *IBM Print Services Facility for AIX: AIX for Users of Print Services Facility*.

The **acif** command then processes the parameters that you have specified on the command line, in the parameter file, or in the shell script.

ACIF Output

With the example, the ACIF job creates the following output files in the current directory:

- OUTDOC** The document file, including indexing structured fields
- INDXOBJ** The index object file
- RESDATA** The resource file
- acif.msg** The message file listing, including the ACIF parameters used, the resources used, and the return code

To view the document file on a workstation using Workbench Viewer, you must first concatenate the index object file, the resource file, and the document file.

Concatenating ACIF Output Files

The following are examples of shell commands you can use to perform the concatenation of the index object file, the resource file, and the document file.

Note: To view the concatenated file with Workbench Viewer, it is important that the index object file is first, followed by the resource file, and then the document file. The document file must be the last data in the concatenated file.

- In the following example, the index object file, the resource file, and the document file are combined to create a new file that contains all three files:

```
cat INDXOBJ RESDATA OUTDOC > NEWFILE
```

- In the following example, the resource file and the document file are added on to the end of the existing index object file:

```
cat RESDATA OUTDOC >> INDXOBJ
```

You may use whichever method you prefer to concatenate the files. You do *not* need to do both.

Accessing the Document File from the Workstation

To view the concatenated document file, it must be accessed from a workstation running Microsoft Windows for viewing with Workbench Viewer. You can access the file from the workstation by either of the following methods:

- *Transfer* the document file, in binary format, to the workstation where Workbench Viewer is installed
- *Mount* your AIX directory on the workstation where Workbench Viewer is installed

Each of these methods is described in the following sections.

Notes:

1. Whether you *transfer* the file to the workstation, or you *mount* your AIX directory on the workstation system, you must have TCP/IP installed on both the AIX system and on the workstation system where Workbench Viewer is installed.
2. To *mount* your AIX directory on the workstation where Workbench Viewer is installed, you must have TCP/IP with Network File System (NFS) installed on both the AIX system and on the workstation system where Workbench Viewer is installed.

For additional information about TCP/IP and NFS, please refer to your TCP/IP documentation.

Transferring the Document File to the Workstation

You can *transfer* the concatenated document file to the workstation where Microsoft Windows and Workbench Viewer are installed with the File Transfer Protocol (FTP) program, using the following procedures:

- From the drive and directory of the workstation where you want the document file to reside, enter the FTP command and the name of your AIX system:

```
ftp AIXsystemname
```

- You will then be prompted for your AIX user name.

```
Enter your AIX user name
```

- You will then be prompted for the password for your AIX user name.

```
Enter the password for your AIX user name
```

- Access the AIX directory where the concatenated document file currently resides. Enter:

```
cd AIXdirectoryname
```

- The file must be transferred in binary format, so you must now enter:

```
bin
```

- To transfer a concatenated document file named NEWFILE, enter:

```
get NEWFILE
```

- The file will now be copied to the workstation, where you may open it for viewing with Workbench Viewer.

Mounting the AIX Directory on the Workstation

You can *mount* your AIX directory on the workstation where Microsoft Windows and Workbench Viewer are installed by using the NFS **mount** command and the following procedures.

- Before using the NFS **mount** command, the following are required on the AIX system:
 - You must have “root” authority, or have a system administrator perform the following.
 - The **nfsd** daemon program must be started on the “server” where the directory to be mounted resides.
 - Your **/etc/exports** file must contain an entry specifying the directory you want to export (the name of the AIX directory where the concatenated document file resides), and where you want to export it to (the name of the workstation where Microsoft Windows and Workbench Viewer are installed). Edit the **/etc/exports** file to add the following entry:

```
AIXdirectoryname -access=workstationname
```

- Reexport the **/etc/exports** file, as root, by entering:


```
/usr/sbin/exportfs -a
```
 - Your concatenated document file *must* be located in the exported directory, or in a subdirectory of the exported directory.
- To *mount* the AIX directory on the workstation, enter the following from the workstation:

```
mount -lAIXusername x: AIXsystemname:AIXdirectoryname
```

(Where **x:** is the workstation drive where you want to mount the AIX directory.)

When prompted, enter the password for your AIX user name.

Once the system has issued the message indicating that the procedure was successful, you may open the document file for viewing with Workbench Viewer.

When you open the example file for viewing with Workbench Viewer, you can select groups labeled:

- Account Number
- Name
- Address
- City, State, Zip
- Date Due

Chapter 4. User Exits and Attributes of the Input Print File in AIX

A user exit is a point during ACIF processing that enables you to run a user-written program and return control of processing to ACIF after your user-written program ends. ACIF provides data at each exit that can serve as input to the user-written program.

This section describes the following topics:

- User programming exits
- Non-zero return codes
- Attributes of the input print file

User Programming Exits

ACIF provides several sample programming exits to assist you in customizing the product. Use of the programming exits is optional. You specify the names of the exit programs with the **inpexit**, **indxexit**, **outexit**, and **resexit** parameters. Each of these parameters is described in Chapter 2, "Using ACIF Parameters in AIX."

ACIF provides the following sample exits:

/usr/lpp/psf/acif/apkinp.c	Input record exit
/usr/lpp/psf/acif/apkind.c	Index record exit
/usr/lpp/psf/acif/apkout.c	Output record exit
/usr/lpp/psf/acif/apkres.c	Resource exit

In addition, ACIF provides the following user input record exits to translate input data streams:

/usr/lpp/psf/acif/apka2e.c

Converts ASCII stream data to EBCDIC stream data.

/usr/lpp/psf/acif/asciinp.c

Converts unformatted ASCII data that contains carriage returns and form feeds into a record format that contains an American National Standards Institute (ANSI) carriage control character. This exit encodes the ANSI carriage control character in byte 0 of every record.

/usr/lpp/psf/acif/asciinpe.c

Converts unformatted ASCII data into a record format as does **asciinp.c**, and then converts the ASCII stream data to EBCDIC stream data.

The C language header file for all ACIF exit programs is also provided:

/usr/lpp/psf/acif/apkexits.h

along with the build rules for the ACIF user exits:

/usr/lpp/psf/acif/Makefile

For more information about compiling user exit programs, refer to *IBM Print Services Facility for AIX: Print Administration*.

Input Record Exit

ACIF provides an exit that enables you to add, delete, or modify records in the input file. You can also use the exit to insert indexing information. The program invoked at this exit is defined in the ACIF **inpexit** parameter.

This exit is called after each record is read from the input file. The exit can request that the record be discarded, processed, or processed and control returned to the exit for the next input record. The largest record that can be processed is 32756 bytes. This exit is not called when ACIF is processing resources from directories.

In a MO:DCA-P document, indexing information can be passed in the form of a Tag Logical Element (TLE) structured field. For more information about the TLE structured field, see Appendix B, “Data Stream Information.” The exit program can create these structured fields while ACIF is processing the print file. This is an alternative to modifying the application in cases where the indexing information is not consistently present in the application output.

Note: TLEs are not supported in line-mode or mixed-mode data.

Figure 19 contains a sample C language header that describes the control block that is passed to the exit program.

```
typedef struct _INPEXIT_PARMS /* Parameters for the input record exit */
{
    char          *work;        /* Address of 16-byte static work area */
    PFATTR        *pfattr;     /* Address of print file attribute information */
    char          *record;     /* Address of the input record */
    void          *reserved1;  /* Reserved for future use */
    unsigned short recordln;   /* Length of the input record */
    unsigned short reserved2;  /* Reserved for future use */
    char          request;     /* Add, delete, or process the record */
    char          eof;         /* EOF indicator */
} INPEXIT_PARMS;
```

Figure 19. Sample Input Record Exit C Language Header

The address of the control block containing the following parameters is passed to the input record exit:

work (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 75 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)

A pointer to the first byte of the input record including the carriage control character. The record resides in a buffer that resides in storage allocated by ACIF, but the exit program is allowed to modify the input record.

reserved1 (Bytes 13–16)

These bytes are reserved for future use.

recordIn (Bytes 17–18)

Specifies the number of bytes (length) of the input record. If the input record is modified, this parameter must also be updated to reflect the actual length of the record.

reserved2 (Bytes 19–20)

These bytes are reserved for future use.

request (Byte 21)

Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00', X'01', or X'02', where:

X'00' Specifies that the record be processed by ACIF.

X'01' Specifies that the record not be processed by ACIF.

X'02' Specifies that the record be processed by ACIF and control returned to the exit program to allow it to insert the next record. The exit program can set this value to save the current record, insert a record, and then supply the saved record at the next call. After the exit inserts the last record, the exit program must reset the **request** byte to X'00'.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the **request** byte value to X'01'. If you want the record to be processed, and you want to insert an additional record, change the **request** byte value to X'02'. Any value greater than X'02' is interpreted as X'00', and the exit processes the record.

Note: Only one record can reside in the buffer at any time.

eof (Byte 22)

An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that specifies whether an **eof** condition has been encountered. When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the input exit, and the input file has been closed. The pointer **record** is no longer valid. Records may not be inserted when **eof** is signaled. The following are the only valid values for this parameter:

Y Specifies that **eof** has been encountered.

N Specifies that **eof** has not been encountered.

This end-of-file indicator allows the exit program to perform some additional processing at the end of the print file. The exit program cannot change this parameter.

Using the ACIF User Input Record Exits

The **apka2e** input record exit program translates data that is encoded in ASCII (code set IBM-850) into EBCDIC (code set IBM-037) encoded data. You should use this exit when your print job requires fonts such as GT12, which has only EBCDIC code points defined.

To execute the **apka2e** input record exit program, set the following parameters as follows in your ACIF parameter file:

```
inpexit=apka2e
cc=yes
cctype=z
```

Also, ensure that the directory where the **apka2e** input record exit program resides is included in the **PATH** environment variable.

The **asciinp** input record exit program transforms an ASCII data stream into a record format that contains a carriage control character in byte 0 of every record. If byte 0 of the input record is an ASCII carriage return (X'0D'), byte 0 is transformed into an ASCII space (X'20') that causes a data stream to return and advance one line; no character is inserted. If byte 0 of the input record is an ASCII form feed character (X'0C'), byte 0 is transformed into an ANSI skip to channel 1 command (X'31') that serves as a form feed in the carriage control byte.

To execute the **asciinp** input record exit program, set the following parameters as follows in your ACIF parameter file:

```
inpexit=asciinp
cc=yes
cctype=z
```

Also, ensure that the directory where the **asciinp** input record exit program resides is included in the **PATH** environment variable.

The **asciinpe** input record exit program combines both user input record exits described above. To execute, specify `inpexit=asciinpe` and follow the directions specified for both **apka2e** and **asciinp**. Also, ensure that the directory where the **asciinpe** input record exit program resides is included in the **PATH** environment variable.

While the **asciinp** and **asciinpe** input record exits do not recognize other ASCII printer commands, you can modify these exits to account for the following:

- backspacing (X'08')
- horizontal tabs (X'09')
- vertical tabs (X'0B')

For more information on using and modifying these programs, refer to the prolog of the **asciinp.c** source file that is provided with PSF for AIX in the **/usr/lpp/psf/acif** directory.

Index Record Exit

ACIF provides an exit that allows you to modify or ignore the records that ACIF writes in the index object file. The program invoked at this exit is defined by the ACIF **indxexit** parameter.

This exit receives control before a record (structured field) is written to the index object file. The exit program can request that the record be ignored or processed. The largest record that can be processed is 32752 bytes (this does not include the record descriptor word).

Figure 20 contains a sample C language header that describes the control block that is passed to the exit program.

```
typedef struct _INDEXIT_PARMS /* Parameters for the index record exit */
{
    char          *work;      /* Address of 16-byte static work area */
    PFATTR        *pfattr;    /* Address of print file attribute information */
    char          *record;    /* Address of the record to be written */
    unsigned short recordln; /* Length of the output index record */
    char          request;    /* Delete or process the record */
    char          eof;       /* Last call indicator to ACIF */
} INDEXIT_PARMS;
```

Figure 20. Sample Index Record Exit C Language Header

The address of the control block containing the following parameters is passed to the index record exit:

work (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 75 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)

A pointer to the first byte of the index record including the carriage control character. The record resides in a 32KB (where KB equals 1024 bytes) buffer. The buffer resides in storage allocated by ACIF, but the exit program is allowed to modify the index record.

recordln (Bytes 13–14)

Specifies the length, in bytes, of the index record. If the index record is modified, this parameter must also be updated to reflect the actual length of the record.

request (Byte 15)

Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01' where:

X'00' Specifies that the record be processed by ACIF.

X'01' Specifies that the record not be processed by ACIF.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the **request** byte value to X'01'. Any value greater than X'01' is interpreted as X'00'; the record is processed.

Note: Only one record can reside in the buffer at any time.

eof (Byte 16)

An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that signals when ACIF has finished processing the index object file.

When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the index exit. The pointer **record** is no longer valid. Records may not be inserted when **eof** is signaled. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

This end-of-file flag, used as a last call indicator, allows the exit program to return control to ACIF. The exit program cannot change this parameter.

Output Record Exit

Using the output record exit, you can modify or ignore the records ACIF writes into the output document file. The program invoked at this exit is defined by the ACIF **outexit** parameter.

The exit receives control before a record (structured field) is written to the output document file. The exit can request that the record be ignored or processed. The largest record that the exit can process is 32752 bytes, not including the record descriptor word. The exit is not called when ACIF is processing resources.

Figure 21 contains a sample C language header that describes the control block passed to the exit program.

```
typedef struct _OUTEXIT_PARMS /* Parameters for the output record exit */
{
    char          *work;      /* Address of 16-byte static work area */
    PFATTR        *pfattr;    /* Address of print file attribute information */
    char          *record;    /* Address of the record to be written */
    unsigned short recordln; /* Length of the output record */
    char          request;    /* Delete or process the record */
    char          eof;        /* Last call indicator */
} OUTEXIT_PARMS;
```

Figure 21. Sample Output Record Exit C Language Header

The address of the control block containing the following parameters is passed to the output record exit:

work (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 75 for more information on the format of this data structure and the information contained in it.

record (Bytes 9–12)

A pointer to the first byte of the output record. The record resides in a 32KB (where KB equals 1024 bytes) buffer. The buffer resides in storage allocated by ACIF, but the exit program is allowed to modify the output record.

recordln (Bytes 13–14)

Specifies the length, in bytes, of the output record. If the output record is modified, this parameter must also be updated to reflect the actual length of the record.

request (Byte 15)

Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01', where:

X'00' Specifies that the record be processed by ACIF.

X'01' Specifies that the record be ignored by ACIF.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the **request** byte value to X'01'. Any value greater than X'01' is interpreted as X'00'; the exit processes the record.

Note: Only one record can reside in the buffer at any time.

eof (Byte 16)

An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that signals when ACIF has finished writing the output file.

When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the output exit. The pointer **record** is no longer valid. Records may not be inserted when **eof** is signaled. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

This end-of-file flag, used as a last-call indicator, allows the exit program to return to ACIF. The exit program cannot change this parameter.

Resource Exit

ACIF provides an exit that enables you to “filter” resources from being included in the resource file. If you want to exclude a specific type of resource (for example, an overlay), you can control this with the **restype** parameter. This exit is useful in controlling resources at the file name level. For example, assume you were going to send the output of ACIF to PSF for AIX and you only wanted to send those fonts that were not shipped with the PSF for AIX product. You could code this exit program to contain a table of all fonts shipped with PSF for AIX and filter those from the resource file. Security is another consideration for using this exit because you could prevent certain named resources from being included. The program invoked at this exit is defined by the ACIF **resexit** parameter.

This exit receives control before a resource is read from a directory. The exit program can request that the resource be processed or ignored (skipped), but it cannot substitute another resource name in place of the requested one. If the exit requests any overlay to be ignored, ACIF will automatically ignore any resources the overlay may have referenced (that is, fonts and page segments).

Figure 22 contains a sample C language header that describes the control block that is passed to the exit program.

```
typedef struct _RESEXIT_PARMS /* Parameters for the resource record exit */
{
    char          *work;      /* Address of 16-byte static work area */
    PFATTR        *pfattr;   /* Address of print file attribute information */
    char          resname[8]; /* Name of requested resource */
    char          restype;   /* Type of resource */
    char          request;   /* Ignore or process the resource */
    char          eof;       /* Last call indicator */
} RESEXIT_PARMS;
```

Figure 22. Sample Resource Exit C Language Header

The address of the control block containing the following parameters is passed to the resource record exit:

work (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 75 for more information on the format of this data structure and the information presented.

resname (Bytes 9–16)

Specifies the name of the requested resource. This value cannot be modified (changed) by the exit program.

restype (Byte 17)

Specifies the type of resource the name refers to. This is a 1-byte hexadecimal value where:

X'40' Specifies a font character set.

- X'41'** Specifies a code page.
- X'FB'** Specifies a page segment.
- X'FC'** Specifies an overlay.

ACIF does **not** call this exit for the following resource types:

- Page definition

The page definition (**pagedef**) is a required resource for processing line-mode application output. The page definition is never included in the resource file.

- Form definition

The form definition (**formdef**) is a required resource for processing print files. If you do not want the form definition included in the resource file, specify **restype=none** or explicitly exclude it from the **restype** list.

- Coded fonts

ACIF does not include any referenced coded fonts in the resource file; therefore, resource filtering is not applicable. ACIF must process coded fonts to determine the names of the code pages and font character sets they reference. This is necessary in creating Map Coded Font-2 (MCF-2) structured fields.

- COM setup files

A COM setup file (**setup**) is a required resource for processing microfilm files. If you do not want a setup file included in the resource file, specify **restype=none** or explicitly exclude it from the **restype** list.

request (Byte 18)

Specifies how the resource is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01' where:

- X'00'** Specifies that the resource be processed by ACIF.
- X'01'** Specifies that the resource not be processed by ACIF.

A value of X'00' on entry to the exit program specifies that the resource be processed. If you want to ignore the resource, change the **request** byte value to X'01'. Any value greater than X'01' is interpreted as X'00' the resource is processed.

eof (Byte 19)

An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that signals when ACIF has finished writing the resource file.

When **eof** is signaled (**eof** value = 'Y'), the last record has already been presented to the resource exit. The pointer **record** is no longer valid. Records may not be inserted when **eof** is signaled. The following are the only valid values for this parameter:

- Y** Specifies that the last record has been written.
- N** Specifies that the last record has not been written.

This end-of-file flag, used as a last-call indicator, returns control to ACIF. The exit program cannot change this parameter.

Non-Zero Return Codes

If ACIF receives a non-zero return code from any exit program, ACIF issues message 0425-412 and terminates processing.

Attributes of the Input Print File

ACIF provides information about the attributes of the input print file in a data structure available to ACIF's user exits. Figure 23 shows the format of this data structure.

```
typedef struct _PFATTR      /* Print File Attributes          */
{
    char      cc[3];        /* Carriage controls? - "YES" or "NO "          */
    char      cctype[1];   /* Carriage control type - A(ANSI), M(Machine), Z(ASCII) */
    char      chars[20];   /* CHARS values, including commas (eg. GT12,GT15) */
    char      formdef[8];  /* Form Definition (FORMDEF)                    */
    char      pagedef[8];  /* Page Definition (PAGEDEF)                    */
    char      prmode[8];   /* Processing mode                               */
    char      trc[3];      /* Table Reference Characters - "YES" or "NO "   */
} PFATTR;
```

Figure 23. Sample Print File Attributes C Language Header

The address of the control block containing the following parameters is passed to the user exits:

cc (Bytes 1–3)

The value of the **cc** parameter as specified on the **acif** command. ACIF uses the default value if this parameter is not explicitly specified.

cctype (Byte 4)

The value of the **cctype** parameter as specified on the **acif** command. ACIF uses the default value if this parameter is not explicitly specified.

chars (Bytes 5–24)

The value of the **chars** parameter as specified on the **acif** command, including any commas that separate multiple font specifications. Because the **chars** parameter has no default value, this field contains blanks if no values are specified.

formdef (Bytes 25–32)

The value of the **formdef** parameter as specified on the **acif** command. Because the **formdef** parameter, has no default value, this field contains blanks if no value is specified.

pagedef (Bytes 33–40)

The value of the **pagedef** parameter as specified on the **acif** command. Because the **pagedef** parameter has no default value, this field contains blanks if no value is specified.

prmode (Bytes 41–48)

The value of the **prmode** parameter as specified on the **acif** command. Because the **prmode** parameter has no default value, this field contains blanks if no value is specified.

trc (Bytes 49–51)

The value of the **trc** parameter as specified on the **acif** command. ACIF uses the default value if this parameter is not explicitly specified.

Notes:

1. Each of the previous character values is left-justified; that is, padding blanks are added to the end of the string. For example, if **pagedef=P1TEST** is specified on the **acif** command, the page definition value in the above data structure is 'P1TEST'.
2. Exit programs cannot change the values supplied in this data structure. For example, if 'P1TEST' is the page definition value, and an exit program changes the value to 'P1PROD', ACIF still uses 'P1TEST'.
3. This data structure is provided for informational purposes only.

Chapter 5. IBM AFP Fonts for ASCII Data

When you specify a coded font name with the **chars** parameter of the **acif** command or the **line2afp** command, the font name is limited to four characters, excluding the two-character prefix.

Figure 24 provides a list of the IBM Core Interchange Fonts for use with unformatted ASCII input data. Because these fonts have eight-character names, the table also provides a list of six-character short names. These coded fonts are installed in the **/usr/lpp/psf/reslib** directory when the **psf.acif** installation option is installed. The installation program also creates the symbolic links of the eight-character names that correspond to the six-character names. You may use these short names, *without* the **X0** prefix, to satisfy the four-character limitation for specifying font names with the **chars** parameter.

Figure 24 (Page 1 of 2). Font Mapping Table for Use with the chars Parameter

Type Family	Point Size	Coded Font Name	Linked Short Name (for chars Parameter)
Courier	7	X0423072	X04272
Courier	8	X0423082	X04282
Courier	10	X0423002	X04202
Courier	12	X04230B2	X042B2
Courier	14	X04230D2	X042D2
Courier	20	X04230J2	X042J2
Helvetica	6	X0H23062	X0H262
Helvetica	7	X0H23072	X0H272
Helvetica	8	X0H23082	X0H282
Helvetica	9	X0H23092	X0H292
Helvetica	10	X0H23002	X0H202
Helvetica	11	X0H230A2	X0H2A2
Helvetica	12	X0H230B2	X0H2B2
Helvetica	14	X0H230D2	X0H2D2
Helvetica	16	X0H230F2	X0H2F2
Helvetica	18	X0H230H2	X0H2H2
Helvetica	20	X0H230J2	X0H2J2
Helvetica	24	X0H230N2	X0H2N2
Helvetica	30	X0H230T2	X0H2T2
Helvetica	36	X0H230Z2	X0H2Z2
Times New Roman	6	X0N23062	X0N262
Times New Roman	7	X0N23072	X0N272
Times New Roman	8	X0N23082	X0N282
Times New Roman	9	X0N23092	X0N292
Times New Roman	10	X0N23002	X0N202

Figure 24 (Page 2 of 2). Font Mapping Table for Use with the chars Parameter

Type Family	Point Size	Coded Font Name	Linked Short Name (for chars Parameter)
Times New Roman	11	X0N230A2	X0N2A2
Times New Roman	12	X0N230B2	X0N2B2
Times New Roman	14	X0N230D2	X0N2D2
Times New Roman	16	X0N230F2	X0N2F2
Times New Roman	18	X0N230H2	X0N2H2
Times New Roman	20	X0N230J2	X0N2J2
Times New Roman	24	X0N230N2	X0N2N2
Times New Roman	30	X0N230T2	X0N2T2
Times New Roman	36	X0N230Z2	X0N2Z2

Part 3. Using ACIF in the MVS, VM, and VSE Environments

Chapter 6. Using ACIF in MVS, VM, and VSE

This chapter describes how to invoke ACIF in the MVS, VM, and VSE environments.

Using ACIF in the MVS Environment

Figure 25 contains sample JCL to invoke ACIF to process print output from an application.

```
//USERAPPL EXEC PGM=user application
//PRINTOUT DD DSN=print file,DISP=(NEW,CATLG)
//*
//ACIF      EXEC=APKACIF,PARM=[[ 'PARMDD=ddname ][,MSGDD=ddname ']],REGION=3M
//INPUT     DD DSN=*.USERAPPL.PRINTOUT
//OUTPUT    DD DSN=output file,DISP=(NEW,CATLG),
//          DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBA,DSORG=PS),
//          SPACE=(32760,(nn,nn)),UNIT=SYSDA
//RESOBJ    DD DSN=resource file,DISP=(NEW,CATLG),
//          DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBA,DSORG=PS),
//          SPACE=(32760,(nn,nn)),UNIT=SYSDA
//INDEX     DD DSN=index file,DISP=(NEW,CATLG),
//          DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBA,DSORG=PS),
//          SPACE=(32760,(nn,nn)),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
ACIF parms go here
```

Figure 25. Sample MVS JCL to Invoke ACIF

Explaining the MVS JCL Statements

The JCL statements in Figure 25 are explained as follows. For more information about programming JCL, refer to *Print Services Facility/MVS: Application Programming Guide*.

USERAPPL

Represents the job step to run the application that produces the actual print output. *USERAPPL* or *user application* is the name of the program that produces the print data set.

PRINTOUT

The DD statement that defines the output data set produced from the application. The application output cannot be spooled to the Job Entry Subsystem (JES), because ACIF does not read data from the spool. The *print file* is the name of the print data set created by the *user application*.

ACIF

Represents the job step that invokes ACIF to process the print data set. You can specify two optional input parameters to ACIF:

PARMDD

Defines the DDname for the data set containing the ACIF processing parameters. If **PARMDD** is not specified, ACIF uses **SYSIN** as the default DDname and terminates processing if **SYSIN** is not defined.

MSGDD

Defines the DDname for the message data set. When ACIF processes a print data set, it can issue a variety of informational or error messages. If **MSGDD** is not specified as an invocation parameter, ACIF uses **SYSPRINT** as the default DDname and stops processing if **SYSPRINT** is not defined.

Although the sample shows a specified REGION size of 3MB, this value can vary, depending on the complexity of the input data and the conversion and indexing options requested.

INPUT

This DD statement defines the print data set to be processed by ACIF. In the sample in Figure 25 on page 81, this is the same data set as defined in the **PRINTOUT** DD statement.

OUTPUT

This DD statement defines the name of the print data set that ACIF creates as a result of processing the application's print data set. See Figure 25 on page 81 for the DCB requirements.

RESOBJ

This DD statement defines the name of the resource data set that ACIF creates as a result of processing the print data set. This statement is not required if **RESTYPE=NONE** is specified in the processing parameter data set. See page106 for more information about the **RESTYPE** parameter.

INDEX

This DD statement defines the name of the index object file that ACIF creates as a result of processing the application's print data set.

This parameter is not required unless indexing is requested or unless the input print data set contains indexing structured fields. If you are not sure whether the print data set contains indexing structured fields, and you do not want an index object file created, specify DD DUMMY; no index object file will be created.

SYSPRINT

If you are not writing messages to spool, the data set must have the following attributes: **LRECL=137,BLKSIZE=** multiple of LRECL + 4 **RECFM=VBA**.

SYSIN

This DD statement defines the data set containing the ACIF processing parameters. This is the default DDname if **PARMDD** is not specified as an invocation parameter.

Note: Files named by the **FDEFLIB**, **PDEFLIB**, **PSEGLIB**, and **OVLYLIB** parameters are allocated to system-generated DDnames.

Using ACIF in the VM Environment

Figure 26 contains sample VM/CMS commands to invoke ACIF to process print output from an application.

```
USERAPPL
FILEDEF INPUT DISK filename filetype filemode
FILEDEF OUTPUT DISK filename filetype filemode (LRECL 32756 BLKSIZE 32760)
FILEDEF RESOBJ DISK filename filetype filemode (LRECL 32756 BLKSIZE 32760)
FILEDEF INDEX DISK filename filetype filemode (LRECL 32756 BLKSIZE 32760)
FILEDEF SYSIN DISK filename filetype filemode
FILEDEF SYSPRINT DISK filename filetype filemode
FILEDEF TRACE DISK filename filetype filemode
APKACIF (PARMDD ddname MSGDD ddname)
```

Figure 26. Sample VM CMS Commands to Invoke ACIF

Note: The usage of DD in these statements reflects the same naming convention of ACIF used in MVS.

Explaining the VM CMS Commands

The CMS commands in Figure 26 are explained as follows. For more information about programming CMS commands, refer to *Print Services Facility/VM: Application Programming Guide*.

USERAPPL

Invokes the application that produces the actual print output.

INPUT

Defines the DDname for the print file to be processed by ACIF. In the sample in Figure 26, this is the same print file that is created by **USERAPPL**.

OUTPUT

Defines the DDname for the file that ACIF creates as a result of processing the application's print file.

RESOBJ

Defines the DDname for the resource file that ACIF creates as a result of processing the application's print file. This command is not required if **RESTYPE=NONE** is specified in the processing parameter file.

INDEX

Defines the DDname for the index object file that ACIF creates as a result of processing the application's print file.

This parameter is not required unless indexing is requested or unless the print file contains indexing structured fields. If you are not sure whether the print file contains indexing structured fields, and you do not want an index object file created, specify FILEDEF INDEXDD DUMMY; no index object file will be created.

TRACE

Defines the default DDname for the trace file. This file is not created unless **TRACE=YES** is specified in the processing parameter file.

APKACIF

Invokes the ACIF program to process the application's print file. You can specify two optional input parameters to ACIF: **PARMDD** and **MSGDD**.

PARMDD

Defines the DDname for the file containing the ACIF processing parameters. If **PARMDD** is not specified, ACIF uses **SYSIN** as the default DDname and terminates processing if **SYSIN** is not defined.

MSGDD

Defines the DDname type for the message file. When ACIF processes a print file, it can issue a variety of informational or error messages. If **MSGDD** is not specified as an invocation parameter, ACIF uses **SYSPRINT** as the default DDname and terminates processing if **SYSPRINT** is not defined. **MSGDD** requires a LRECL of 137 and a block size that is a multiple of 137 plus 4 (for example, $(137*10)+4 = 1374$).

ACIF requires about 3MB of virtual memory to convert and index files. The amount of memory can vary, depending on the complexity of the input data and the conversion and indexing options requested.

Using ACIF in the VSE Environment

Figure 27 contains sample JCL to invoke ACIF to process print output from an application.

```
// DLBL PRNTOUT,'user print file'  
// EXTENT ....  
// ASSGN ...  
// EXEC USERAPPL  
// DLBL PRD2,'VSE'PRD2.LIBRARY'  
// EXTENT ,volser  
// LIBDEF PHASE,SEARCH=(PRD2.AFP)  
// ASSGN SYSLST,X'FEE'  
// ASSGN SYS006,xxx  
// DLBL INPUT,'your input file',0,SD  
// EXTENT SYS006,volser...  
// ASSGN SYS007,xxx  
// DLBL OUTPUT,'your output file',0,SD  
// EXTENT SYS007,volser...  
// ASSGN SYS008,xxx  
// DLBL RESOBJ,'your resource output file',0,SD  
// EXTENT SYS008,volser...  
// ASSGN SYS009,xxx  
// DLBL INDEX'your index output file',0,SD  
// EXTENT SYS009,volser...  
// EXEC PGM=APKACIF  
  ACIF parms go here  
/*  
/&
```

Figure 27. Sample VSE JCL to Invoke ACIF

Explaining the VSE JCL Statements

The statements in Figure 27 are explained as follows. For more information about programming JCL for VSE, refer to *Print Services Facility/VSE: Application Programming Guide*.

PRNTOUT

Defines the output file produced from the application. The application output cannot be spooled to POWER, because ACIF does not read data from the spool. The *user print file* is the name of the print data set created by your application.

USERAPPL

Represents the job step to run the application that produces the actual print output. The *user application* refers to the program that produces the print file.

```
// DLBL PRD2,'VSE'PRD2.LIBRARY'  
// EXTENT ,volser  
// LIBDEF PHASE,SEARCH=(PRD2.AFP)
```


Defines the library or libraries to be searched for the ACIF program and for all the AFP resources (form definitions, page definition, fonts, overlays, and page segments).

```
// ASSGN SYSLST,...
```

Defines the control statement and error message listing file.

```
// ASSGN SYS006, ...
// DLBL INPUT, ...
// EXTENT SYS006, ...
```

Defines the file to be processed by ACIF. In the sample in Figure 27 on page 85, this is the same data set as defined by the **PRNTOUT** file.

```
// ASSGN SYS007, ...
// DLBL OUTPUT, ...
// EXTENT SYS007, ...
```

Defines the document file that ACIF creates as a result of processing the application's print file. See **OUTPUTDD** on page 100 for the characteristics of this file.

```
// ASSGN SYS008, ...
// DLBL RESOBJ, ...
// EXTENT SYS008, ...
```

Defines the optional file in which ACIF places print resources used in processing the application's print file. This file is not required if **RESTYPE=NONE** is specified in the processing parameter file.

```
// ASSGN SYS009, ...
// DLBL INDEX, ...
// EXTENT SYS009, ...
```

Defines the optional file in which ACIF places the index object file, if indexing is requested.

This statement is not required unless indexing is requested or unless the input print file contains indexing structured fields. If you are not sure whether the input print file contains indexing structured fields, and you do not want an index object file created, specify `// ASSGN SYS009,IGN;` no index object file will be created.

```
//EXEC PGM=APKACIF
ACIF parms go here.
```

Invokes the ACIF program. This statement must be followed immediately by ACIF processing parameters.

Chapter 7. Using ACIF Parameters in MVS, VM, and VSE

This chapter describes ACIF syntax rules and parameters for MVS, VM, and VSE.

Many of the parameters specified to ACIF are the same as the parameters specified to PSF when you print a job. For those parameters that are common to both PSF and ACIF, you should specify the same value to ACIF as specified to PSF.

Notes:

1. For MVS and VSE, you may need to consult your system programmer for information on resource library names and other printing defaults contained in the PSF startup procedures used in your installation.
2. For VM/CMS, you may need to link to the appropriate disks containing the resource files used to convert and print your job.

Syntax Rules for MVS, VM, and VSE Parameters

The following are general syntax rules for parameter files:

- Each parameter with its associated values can span multiple records, but the parameter and the first value must be specified in the same record. If additional values need to be specified in the following record, a comma (,) must be specified, following the last value in the previous record. The comma indicates that additional values are specified in one or more of the following records. Underscored values are the default and are used by ACIF if no other value is specified.

MVS

```
FDEFLIB=TEMP.USERLIB,PROD.LIBRARY,  
OLD.PROD.LIBRARY /* These are the FORMDEF libraries.
```

VM

```
FDEFLIB=FDEF38PP,  
TEMPFDEF /* These are the FORMDEF libraries.
```

VSE

```
INPUTDD=INPUT|filename(LRECL=nnnn,BLKSIZE=nnnn,RECFM=F|FB|V|VB,DEVT=TAPE|DISK)
```

- Blank characters inserted between parameters, values, and symbols are allowed and ignored. For example, specifying:

```
FORMDEF = F1TEMP  
PAGEDEF = P1PROD  
INDEX1 = FIELD1 , FIELD2 , FIELD3
```

Is equivalent to specifying:

```
FORMDEF=F1TEMP  
PAGEDEF=P1PROD  
INDEX1=FIELD1,FIELD2,FIELD3
```

- When ACIF processes any unrecognized or unsupported parameter, it issues a message, ignores the parameter, and continues processing any remaining parameters until the end of the file, at which time it terminates processing.

- If the same parameter is specified more than one time, ACIF uses the last value specified. For example, if the following is specified:

```
CPGID=037
CPGID=395
```

ACIF uses code page 395.

- Comments must be specified using “/*” as the beginning delimiter. For example:

```
FORMDEF=F1TEMP /* Temporary FORMDEF
FORMDEF=F1PROD /* Production-level FORMDEF
```

Comments can appear anywhere, but ACIF ignores all information in the record following the “/*” character string.

- Although ACIF supports parameter values spanning multiple records, it does not support multiple parameters in a single record. The following is an example of this:

```
CHARS=X0GT10 CCTYPE=A /* This is not allowed.
```

Figure 28 (Page 1 of 2). ACIF Parameters, Tasks, and Operating Systems

ACIF Parameters	Task Usage Key	Operating System
CC= <u>YES</u> NO	A	MVS, VM, VSE
CCTYPE= <u>A</u> M Z	C	MVS, VM, VSE
CHARS= <i>fontname1</i> [, <i>fontname2</i>] [, <i>fontname3</i>] [, <i>fontname4</i>]	C, R	MVS, VM, VSE
COMSETUP= <i>name</i>	R	MVS
CPGID= <u>500</u> <i>code page identifier</i>	I	MVS, VM, VSE
DCFPAGENAMES= <i>value</i>	I	MVS, VM, VSE
FDEFLIB= <i>data set name1</i> [, <i>data set name2</i>] [, <i>data set name...</i>]	A	MVS
FDEFLIB= <i>filetype1</i> [, <i>filetype2</i>] [, <i>filetype...</i>]	A	VM
FIELDn= <i>record, column, length</i> { ' <i>literal value1</i> ' <i>X</i> ' <i>literal value'</i> ' }	I	MVS, VM, VSE
FONTLIB= <i>data set name1</i> [, <i>data set name2</i>] [, <i>data set name...</i>]	C, R	MVS
FONTLIB= <i>filetype1</i> [, <i>filetype2</i>] [, <i>filetype...</i>]	C, R	VM
FORMDEF= <i>fdefname</i>	A	MVS, VM, VSE
GROUPNAME= <u>INDEX1</u> INDEXn	I	MVS, VM, VSE
IMAGEOUT=ASIS <u>IOCA</u>	C	MVS, VM, VSE
INDEXn= <i>'attribute name'</i> , FIELDn [, FIELDn...]	I	MVS, VM, VSE
INDEXDD= <u>INDEX</u> <i>ddname</i>	I	MVS, VM
INDEXDD= <u>INDEX</u> <i>filename</i> (DEV= <u>TAPE</u> <u>DISK</u>)	I	VSE
INDEXOBJ= <u>GROUP</u> ALL NONE	I	MVS, VM, VSE
INDEXSTARTBY= <u>1</u> <i>nn</i>	I	MVS, VM, VSE
INDEXEXIT= <i>module name</i>	I	MVS, VM, VSE
INPEXIT= <i>module name</i>	G	MVS, VM, VSE
INPUTDD= <u>INPUT</u> <i>ddname</i>	G	MVS, VM

Figure 28 (Page 2 of 2). ACIF Parameters, Tasks, and Operating Systems

ACIF Parameters	Task Usage Key	Operating System
INPUTDD=INPUT filename(LRECL=nnnn,BLKSIZE=nnnn, RECFM=F FB V VB,DEVT=TAPE DISK)	G	VSE
OBJCONLIB =data set name1[,data set name2][,data set name...]	R	MVS
OUTEXIT =module name	G	MVS, VM, VSE
OUTPUTDD=OUTPUT ddname	G	MVS, VM
OUTPUTDD=OUTPUT filename(DEVT=TAPE DISK)	G	VSE
OVLYLIB =data set name1[,data set name2][,data set name...]	R	MVS
OVLYLIB =filetype1[,filetype2][,filetype...]	R	VM
PAGEDEF =pdefname	C	MVS, VM, VSE
PDEFLIB =data set name1[,data set name2][,data set name...]	C	MVS
PDEFLIB =filetype1[,filetype2][,filetype...]	C	VM
PRMODE =SOS11 SOS12 aaaaaaaaa	C	MVS, VM, VSE
PSEGLIB =data set name1[,data set name2][,data set name...]	R	MVS
PSEGLIB =filetype1[,filetype2][,filetype...]	R	VM
RESEXIT =module name	R	MVS, VM, VSE
RESFILE =SEQ PDS	R	MVS
RESOBJDD=RESOBJ ddname	R	MVS, VM
RESOBJDD=RESOBJ filename(DEVT=TAPE DISK)	R	VSE
RESTYPE =NONE [FDEF][,PSEG][,OVLY][,FONT][,OBJCON] ALL	R	MVS, VM, VSE
TRACE =YES NO	G	MVS, VM, VSE
TRACEDD =ddname	G	VM
TRACEDD=TRACE filename(DEVT=TAPE DISK)	G	VSE
TRC =YES NO	C	MVS, VM, VSE
TRIGGERn ={record *},{column *},{'value' X' value'}	I	MVS, VM, VSE
UNIQUEBNGS = YES NO	I	MVS, VM, VSE
USERLIB =data set name1[,data set name2][,data set name...]	C, R	MVS
Notes.: A Parameters used in all ACIF tasks G General Parameters used in any ACIF task I Parameters used in ACIF indexing tasks R Parameters used in ACIF resource tasks C Parameters used in ACIF converting tasks		

The processing parameters are optional unless noted otherwise.

CC=YES | NO

Specifies whether the input file has carriage control characters. If this parameter is not specified, ACIF assumes that the file contains carriage control characters.

CCTYPE=A | M | Z

Specifies the type of carriage-control characters in the input file. ACIF supports ANSI carriage-control characters in either ASCII or EBCDIC encoding, as well as machine carriage-control characters. ACIF does not allow a mixture of ANSI and machine carriage-control characters within a file. Values are:

- Z** The file contains ANSI carriage-control characters that are encoded in ASCII.

The carriage-control characters are the ASCII hexadecimal values that directly relate to ANSI carriage-controls, which cause the action of the carriage-control character to occur *before* the line is printed. For example, if the carriage-control character is zero (X'30'), which represents double spacing, double spacing will occur *before* the line is printed.

- A** The file contains ANSI carriage-control characters that are encoded in EBCDIC.

The use of ANSI carriage-control characters cause the action of the carriage-control character to occur *before* the line of data is printed. For example, if the carriage-control character is a zero (X'F0'), which represents double spacing, the double spacing will occur *before* the line is printed.

- M** The file contains machine code carriage-control characters that are encoded in hexadecimal format.

The use of machine code carriage-control characters cause the action of the carriage-control character to occur *after* the line of data is printed. For example, if the carriage-control character is a X'11', which represents double spacing, the line will be printed and the double spacing will occur *after* the line is printed. In addition, machine code carriage-control has a set of carriage-control characters that perform the action, but do not print the associated line. For example, if the carriage-control character is a X'13', which also represents double spacing, the print position will be moved down two lines but the line that contains the X'13' carriage-control character will not be printed. The next line in the data will be printed at the current print position and the action for the associated carriage-control character will be performed *after* the line is printed.

If you specify **CC=YES** but you do not specify **cctype**, ACIF assumes that the file contains ANSI carriage-control characters encoded in EBCDIC.

If you are not sure which type of carriage-control characters are in your input file, consult your system support group.

CHARS=fontname1[,fontname2][,fontname3][,fontname4]

Specifies the file name (in MVS and VSE, the member name) of the coded font you want ACIF to use to process a file.

fontname

Specifies the file name of the coded font. The name does not include the 2-character prefix of the coded-font name (X0 through XG). The file name for a **CHARS** parameter is limited to 4 alphanumeric or national characters.

Use **CHARS** to specify coded fonts in a font library having names of 6 or fewer characters (including the prefix). You can rename any fonts having more than 6 characters or use a font utility program to create new coded fonts for use with the CHARS parameter.

In line-mode data, the fonts are specified either in a page definition or in the **CHARS** parameter, but not in both. You cannot mix fonts specified in a page definition with fonts specified with CHARS for a single file. Select fonts either with table-reference characters (TRCs), with AFP structured fields, or in a page definition.

EXAMPLE

In the following example, two fonts are specified: X0GT10 (Gothic 10 pitch) and X0GT12 (Gothic 12 pitch):

```
CHARS=GT10,GT12
```

Notes:

1. You can specify fonts in the **CHARS** parameter only if you want the entire file printed in a single printing direction. ACIF uses the fonts that have 0° character rotation for the specified direction. When a file requires fonts with more than one printing direction or character rotation, you must specify the fonts in the page definition.
2. In VM and MVS, fonts you specify must reside in a library specified with the **FONTLIB** parameter or, in MVS, reside in a user library specified with the **USERLIB** parameter. In VSE, you must specify fonts in the // LIBDEF PHASE, SEARCH=(...) JCL statement.
3. You can specify from 1 to 4 fonts with the **CHARS** parameter. If you specify more than 1 font with the **CHARS** parameter, you must use table reference characters (TRCs) to select the fonts.
4. If you specify a page definition using the **PAGEDEF** parameter that specifies fonts for your file, the **CHARS** parameter is ignored.
5. If you do not specify a **CHARS** parameter, and if no fonts are contained in the page definition you specified, ACIF uses the printer default font.

COMSETUP=*name*

Specifies the member name of the setup file. A COM setup file is an AFP resource that contains instructions required when printing on a microfilm device. The value is:

name Any valid COM setup member name. The *name* can be one to eight alphanumeric or national characters, including the two-character prefix, if there is one. The *name* is not case-sensitive.

The COM setup file you use may be located:

- In an MVS library
- Inline in the file (that is, within the file itself)

If the COM setup file is in a library, use the **USERLIB** or **OBJCONLIB** parameter to specify the data sets. For example:

```
COMSETUP=MYSETUP  
USERLIB=USER.RESOURCES
```

or

```
COMSETUP=MYSETUP  
OBJCONLIB=USER.SETUPS
```

If the COM setup file is an inline resource, you must do the following:

- Specify **COMSETUP=***name*, where *name* is the name of the inline COM setup file; or specify **COMSETUP=DUMMY**.

If you specify **COMSETUP=DUMMY** but the file does not include an inline COM setup file, ACIF looks for the COM setup file named **DUMMY**.

If the name specified in the **COMSETUP** parameter does not match the name of an inline COM setup file, ACIF looks for the COM setup file in the **COMSETUP** search path.

An input file can contain multiple COM setup files, but only one COM setup file can be used for printing. If a file contains more than one COM setup file, and you specify **COMSETUP=name**, ACIF uses the first inline COM setup file named *name*. If a file contains more than one inline COM setup file, and you specify **COMSETUP=DUMMY**, ACIF uses the first inline COM setup file in the input file.

CPGID=500 | *code page identifier*

Specifies the code page for the index values and attribute names produced by ACIF. The *code page identifier* is a 3-character decimal value (for example, 395) that defines an IBM-registered code page. ACIF uses this value when it creates a Coded Graphic Character Set Global Identifier Triplet X'01' in the Begin Document (BDT) structured field for the output file. For more information on this triplet, refer to *Mixed Object Document Content Architecture Reference*.

This value is used by programs that must display indexing information. These programs use this value in conjunction with code page translation tables to represent the information. If this parameter is not specified, ACIF uses code page 500 as the default. For code-page numbers less than 100, add leading zeros (for example, 037). If a non-decimal value is specified, ACIF reports an error condition and terminates processing. For more information on code pages, refer to *IBM AFP Fonts: Technical Reference for Code Pages*.

DCFPAGENAMES={YES | NO}

Specifies whether ACIF generates page names using an 8-byte counter or uses structured field tokens found in the input data stream. If the input data contains BPGs with FQNs, ACIF does not generate page names.

YES ACIF uses structured field tokens in the input data stream to generate page names.

NO The default, ACIF generates page names using an 8-byte counter.

FDEFLIB=data set name1[,data set name2][,data set name...] (MVS)

Specifies the data sets that compose the form definition library. A maximum of 8 data sets can be specified. The parameter also specifies the concatenation sequence when ACIF searches for a particular form definition. ACIF first looks for the resource in *data set name1*. If it cannot find the resource in *data set name1*, it continues the search with *data set name2*, and so on, until it locates the requested resource or exhausts the list of specified data sets.

EXAMPLE

```
FDEFLIB=SYS1.FDEFLIB,USER.FDEFLIB
```

If **USERLIB** is also specified, ACIF searches for the resource in the data sets specified in **USERLIB** before searching the data sets identified in **FDEFLIB**.

Notes:

1. Data sets must be specified as fully-qualified names without quotation marks.
2. If the libraries specified for **FORMDEF** are not specified in the same order used by the PSF startup procedure, the printed and converted results may differ. For information on how PSF selects resources, refer to *Print Services Facility/MVS: System Programming Guide*.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
4. This is a required parameter. If FDEFLIB is not specified, ACIF reports an error condition and terminates processing.

FDEFLIB=*filetype1*[,*filetype2*][,*filetype...*] (VM)

Specifies the file types that define the form definition libraries. A maximum of 8 file types can be specified. This parameter also specifies the search order in which ACIF searches for a particular form definition. ACIF first looks for the resource with a file type of *filetype1*. If it cannot find the resource with a file type of *filetype1*, it continues the search with *filetype2*, and so on, until it locates the requested resource or exhausts the list of specified file types.

EXAMPLE

```
FDEFLIB=FDEF38PP,TEMPFDEF
```

Notes:

1. File type values must conform to CMS naming conventions.
2. This is a required parameter. If FDEFLIB is not specified, ACIF reports an error condition and terminates processing.

FDEFLIB

This parameter is not used for **VSE**. Form-definition resources are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement. For information on how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

FIELDn=*record,column,length* [{*'literal value'* | X'*literal value'*}

Specifies the data fields to be used to construct the indexing information. These data fields can be specified either as literal values (constants), or ACIF can retrieve the data from the input records of the file. A maximum of 16 fields can be defined (**FIELD1** through **FIELD16**).

record

Specifies the relative record number from the indexing anchor record. When ACIF is indexing the file, it uses the information specified in the **TRIGGERn** parameters to determine a page-group boundary. When all the specified **TRIGGERn** values are true, ACIF defines the indexing anchor record as the record where **TRIGGER1** is located. **TRIGGER1** becomes the reference point from which all indexing information is located. The supported range of values for *record* are ±0–255.

column

Specifies the byte offset from the beginning of the record. A value of 1 refers to the first byte in the record. For files containing carriage control characters, column 1 refers to the carriage control character. For those

applications that use a specific carriage control character to define page boundaries (for example, skip to channel 1), consider defining the value of the carriage control character as one of the **TRIGGERn** parameters. The supported range of values for *column* are 1–32756. If the specified value exceeds the physical length of the record, ACIF reports an error condition and terminates processing.

length

Specifies the number of contiguous bytes (characters) starting at *column* that compose this field. The supported range of values for *length* is 1–250.

The field can extend outside the record length, as long as the column where it begins lies within record length. In this case, ACIF adds padding blanks (X'40') to fill out the record. If the field begins outside the maximum length of the record, ACIF reports an error condition and terminates processing.

{*'literal value'* | X'*literal value'*}

Specifies the literal (constant) value of the **FIELDn** parameter. This value can be specified either as character data or as hexadecimal data. The literal value can be 1–250 bytes in length. ACIF does not perform any validity checking on the actual content of the supplied data.

EXAMPLE

```
FIELD1=0,2,20
FIELD2=5,5,10
FIELD3=-15,30,5
FIELD4='444663821'
FIELD5=X'0001'
```

The first field in the example is located in the indexing anchor record (**TRIGGER1**). It is 20 bytes in length, starting at the second byte of the record. The second field is located five records down from the indexing anchor record. It is 10 bytes in length, starting at the fifth byte of the record. The third field is located 15 records before the indexing anchor record. It is 5 bytes in length, starting at byte 30. The fourth and fifth fields are literal (constant) values. One is specified as character data (for example, EBCDIC), and the other is hexadecimal data.

ACIF allows fields to be defined but never referenced as part of an index. Because ACIF requires either a field or TRIGGER to appear on the first page of a logical document, unless the **INDEXSTARTBY** parameter is used, you can satisfy this requirement by defining a “DUMMY” field. This DUMMY field allows ACIF to determine the beginning page of a logical document, but it is not used as part of an index. If you specify the **INDEXSTARTBY** parameter, start counting on the first page on which you have a valid field, not a DUMMY field.

FONTLIB=*data set name1*[,*data set name2*][,*data set name...*] (MVS)

Specifies the data sets that compose the font library. A maximum of 8 data sets can be specified. This parameter also specifies the concatenation sequence when ACIF searches for a particular font resource. ACIF first looks for the resource in *data set name1*. If it cannot find the resource in *data set name1*, it continues the search with *data set name2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

EXAMPLE

```
FONTLIB=SYS1.FONTLIB,USER.FONTLIB
```

If **USERLIB** is also specified, ACIF searches for the resource in the data sets specified in **USERLIB** before searching the data sets identified in **FONTLIB**.

Notes:

1. Data sets must be specified as fully-qualified names without quotation marks.
2. If the libraries specified for **FONTLIB** are not specified in the same order used by the PSF startup procedure, the printed and converted results may differ. For information on how PSF selects resources, refer to *Print Services Facility/MVS: System Programming Guide*.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
4. This is a required parameter if font retrieval is requested or if any coded fonts are referenced in the file or in an overlay. The **RESTYPE** value determines whether fonts are to be retrieved for inclusion in the resource data set. If this parameter is not specified, and font retrieval is requested or a coded font is referenced, ACIF reports an error condition and terminates processing.

FONTLIB=*filetype1*[,*filetype2*][,*filetype...*] (VM)

Specifies the file types that define the font libraries. A maximum of 8 file types can be specified. This parameter also specifies the search order when ACIF searches for a particular font resource. ACIF first looks for the resource in *filetype1*. If ACIF cannot find the resource with a file type of *filetype1*, it continues the search with *filetype2*, and so on, until it either locates the requested resource or exhausts the list of specified file types.

EXAMPLE

```
FONTLIB=FONT3820,TESTFONT
```

Notes:

1. File type values must conform to CMS naming conventions.
2. This is a required parameter if font retrieval is requested or if any coded fonts are referenced in the print file or in an overlay. The **RESTYPE** value determines whether fonts are to be retrieved for inclusion in the resource file. If this parameter is not specified, and font retrieval is requested or a coded font is referenced, ACIF reports an error condition and terminates processing.

FONTLIB

This parameter is not used for **VSE**. Font resources are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement. For information on how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

FORMDEF=*fdefname*

Specifies the complete file name (in MVS and VSE, the member name) of the form definition. The *fdefname* can be from 1 to 8 alphanumeric or national characters. Unlike PSF/MVS, PSF/VM, and PSF/VSE, ACIF does **not** require

the name to begin with a F1 prefix, but if the name does begin with F1, you cannot omit it.

A form definition defines how a page of data is placed on a form, the number of copies of a page, any modifications to that group of copies, paper source, and duplexing, as well as other functions. ACIF uses a form definition only to retrieve resources; it does not use a form definition to convert data streams.

EXAMPLE

This example specifies *F1USER10* as the form definition:

```
FORMDEF=F1USER10
```

The form definition you ask ACIF to retrieve may be located:

- Inline in the file
- In a user library referenced in the **USERLIB** parameter (**MVS** only)
- In a library referenced in the **FDEFLIB** parameter (**MVS** and **VM**)
- In a library referenced in the // LIBDEF PHASE,SEARCH=(...) DLBL JCL statement (**VSE**)

Using Form Definitions from an MVS User Library:

You can instruct ACIF to retrieve a form definition from your user library instead of from a library specified in the **FDEFLIB** parameter. To use a form definition from a user library, you must:

- Reference the user library containing the form definition in the **USERLIB** parameter
- Specify the name of the form definition in the **FORMDEF** parameter

Using Inline Form Definitions: To use an inline form definition, you must do the following:

- Include an inline form definition in the file.
- If you specify the **FORMDEF** parameter, the name of the inline form definition must match the name of the specified form definition, or you must specify **FORMDEF=DUMMY**.
- If a form definition resource is included inline with the data, the file must be identified as containing carriage control characters. If the length of the records in the form definition is less than or equal to the logical-record length defined for the file, you can specify fixed-length records for the record format. If the length of the records in the form definition is greater than the logical-record length defined for the file, you must:
 - In **MVS**, specify variable length records for the record format (variable blocked with ANSI carriage control characters [VBA] or variable blocked with machine carriage control characters [VBM]).
 - In **VM**, specify variable length records for the record format.
 - In **VSE**, specify variable length records for the record format (variable blocked with ANSI carriage control characters [VBA] or variable blocked with machine carriage control characters [VBM]).

You can include more than one inline form definition in an input file, and you can change the form definition name in the **FORMDEF** parameter on different printing jobs to test different form definitions.

Notes:

1. If the name specified for the **FORMDEF** parameter does not match the name of an inline form definition, ACIF looks for the form definition in the **FORMDEF** search path.
2. If you specify **FORMDEF=DUMMY**, and you do not include an inline form definition, ACIF reports an error condition and terminates processing.
3. Also, the **FORMDEF** parameter is required. If this parameter is not specified, ACIF reports an error condition and terminates processing.

GROUPNAME=INDEX1 | INDEXn

Specifies which of the eight possible **INDEX** values should be used as the group name for each index group. Using the most unique index value for the group name is recommended. The intent is to have a unique group name for every group ACIF produces in the output file. If this parameter is not specified, ACIF uses the value of **INDEX1** as the default. The value includes the **FIELD** definitions from the **INDEX** parameter but does not include the attribute name. ACIF uses this parameter only when the file is indexed. The Viewer application of AFP Workbench displays this value along with the attribute name and index value. You can use the group name to select a group of pages to be viewed.

IMAGEOUT=ASIS | IOCA

Specifies the format of the image data produced by ACIF in the output document. **ASIS** specifies that ACIF produce the same image format as in the input file. **IOCA** specifies that ACIF produce all image data in uncompressed IOCA format.

INDEXn='attribute name',FIELDn[,FIELDn...]

Specifies the content of the indexing tags for the entire file. A maximum of 8 indexes can be defined (**INDEX1**, **INDEX2**,... **INDEX8**), and each index can be made up of one or more **FIELD** definitions.

If literal values are specified for every index, ACIF treats the entire file as one page group and uses this information to index the document. ACIF reports an error condition and terminates processing if literal values are specified for all **INDEXn** parameters and if any **TRIGGERn** parameters are also specified.

For **FIELD** parameters that specify data values within the file, ACIF determines the actual location of the indexing information based on the indexing anchor record, set by the **TRIGGER** parameters.

A valid set of index parameters comprises:

- **FIELD** definitions containing only constant data (literal values), or
- **FIELD** definitions containing both constant data and application data (data fields in the print file)

You can also specify the same **FIELD** parameters in more than one **INDEX** parameter.

Note: If one or more **TRIGGERn** parameters is specified (that is, ACIF will index the file), at least one **INDEXn** parameter must be specified, and that index must be comprised of at least one **FIELDn** parameter value that is not a literal. ACIF reports an error condition and terminates processing if this rule is not satisfied.

'attribute name'

Specifies a user-defined attribute name to be associated with the actual index value. For example, assume **INDEX1** is a person's bank account number. The string 'account number' would be a meaningful attribute name. The value of **INDEX1** would be the account number (for example, 1234567). Think of the attribute name as a label for the actual index value. The attribute name is an EBCDIC character string from 1–250 bytes in length. ACIF does not perform any validity checking on the contents of the attribute name.

FIELDn[,FIELDn...]

Specifies one or more **FIELDn** parameters that compose the index value. A maximum of 16 **FIELDn** parameters can be specified. If more than one **FIELDn** parameter is specified, ACIF concatenates them into one physical string of data. No delimiters are used between the concatenated fields. Because an index value has a maximum length of 250 bytes, the total of all specified **FIELDn** parameters for a single index cannot exceed this length. ACIF reports an error condition and terminates processing if this occurs.

EXAMPLE

```
FIELD1='1234567'  
FIELD2=0,10,20  
FIELD3=0,25,20  
INDEX1='Patent Number',FIELD1  
INDEX2='Employee Name',FIELD2,FIELD3
```

This example specifies that the first index tag is made up of the literal character string '1234567', while the other two index tags are made up of fields within the file records.

```
FIELD1='123456'  
FIELD2='444556677'  
INDEX1='Account Number',FIELD1  
INDEX2='Social Security Number',FIELD2
```

This example specifies both index tags as literal values. The entire file will be indexed using these two values. The resulting index object file contains only one record.

INDEXDD=INDEX | ddname (MVS and VM)

Specifies the DDname for the index object file. The DDname is a 1–8 byte character string containing only those alphanumeric characters supported in the operating environment. When ACIF is indexing the file, it writes indexing information to this DDname. If **INDEXDD** is not specified, ACIF uses **INDEX** as the default DDname. The following are suggested DCB characteristics for the file:

- A block size of 32760
- A maximum record length of 32756
- Variable blocked format
- Physical sequential format

INDEXDD=INDEX | filename (DEVT=TAPE | DISK) (VSE)

Specifies the file name that appears on the DLBL or TLBL JCL statement, a 1–7 character string containing only those alphanumeric characters supported in the operating environment. The following are the characteristics for the file:

- A block size of 32760

- A maximum record length of 32 756
- Variable blocked format
- Assigned to programmer logical unit 009

INDEXOBJ=GROUP | ALL | NONE

Specifies the amount of information ACIF puts in the index object file. Selecting **GROUP** causes only group-level entries to be put in the index object file, which saves space. Selecting **ALL** causes both page-level and group-level entries to be put in the index object file. You should select **ALL** if you are indexing a file for use with the Viewer application of AFP Workbench. If this parameter is not specified, ACIF uses **GROUP** as the default. Choose **NONE** if you do not require an external index file. Choosing **NONE** will also reduce ACIF storage requirements.

INDEXSTARTBY=1|nn

Specifies the output page number by which ACIF must find an indexing field, if ACIF is indexing the file. If ACIF does not find an indexing field, it issues a message and stops processing.

This parameter is helpful if your file contains header pages. You can specify a page number 1 greater than the number of header pages, so that ACIF will continue to look for matches for the number of pages specified for this parameter.

INDEXIT=module name

Specifies the name of the index record exit program. This is a 1–8 byte character name of the load module ACIF loads during initialization and subsequently calls for every record (structured field) it writes to the index object file (**INDEXDD**). If this parameter is not specified, no index record exit is used. See “Index Record Exit” on page 69 for more detailed information.

INPEXIT=module name

Specifies the name of the input record exit program. This is a 1–8 byte character name of the load module ACIF loads during initialization and subsequently call for every input record it reads from the input file (**INPUTDD**). If this parameter is not specified, no input record exit is used. See “Input Record Exit” on page 121 for more detailed information.

INPUTDD=INPUT | DDname (MVS and VM)

Specifies the DDname for the file ACIF will process. *ddname* is a 1–8 byte character string containing only those alphanumeric characters supported in the operating environment. When ACIF processes a file, it reads from this DDname. If **INPUTDD** is not specified, ACIF uses INPUT as the default DDname.

INPUTDD=INPUT | filename (LRECL=nnnn,BLKSIZE=nnnn,RECFM=F|FB|V|VB, DEVT=TAPE|DISK) (VSE)

Specifies the file name that appears on the DLBL or TLBL JCL statement, a 1–7 character string containing only those alphanumeric characters supported in the operating environment. You must specify the characteristics of this file. The defaults are:

- A disk input file
- Fixed-length, unblocked records, 133 bytes in length
- Assigned to programmer logical unit 006

LRECL=nnnn

Specifies the record length of the input data set.

BLKSIZE=nnnn

Specifies the block size of the input data set.

RECFM=F|FB|V|VB

Specifies the record format of the input data set (VSE only).

F Fixed

FB Fixed Block

V Variable

VB Variable Block

DEVT=TAPE|DISK

Specifies the device type, either **TAPE** or **DISK**

OBJCONLIB=data set name1[,data set name2][,data set name...] (MVS)

Specifies the data sets that compose the setup file library. A maximum of 8 data sets can be specified. The parameter also specifies the concatenation sequence when ACIF searches for a particular setup file. ACIF first looks for a setup file in data set name1. If it cannot find the setup file in data set name1, it continues the search with data set name2, and so on, until it locates the requested setup file or exhausts the list of specified data sets.

If **USERLIB** is also specified, ACIF searches for the resource in the data sets specified in the **USERLIB** before searching the data sets identified in **OBJCONLIB**.

OUTEXIT=module name

Specifies the name of the output record exit program. This is a 1–8 byte character name of the load module ACIF loads during initialization and subsequently call for every output record it writes to the output document file (**OUTPUTDD**). If this parameter is not specified, no output record exit is used. See “Output Record Exit” on page 125 for more detailed information.

OUTPUTDD=OUTPUT | DDname (MVS and VM)

Specifies the DDname for the output document file ACIF produces when it processes a file. The *DDname* is a 1–8 byte character string containing only those alphanumeric characters supported in the operating environment. When ACIF processes a print file, it writes the resultant converted print data to this DDname. If **OUTPUTDD** is not specified, ACIF uses **OUTPUT** as the default DDname. Suggested DCB characteristics of the file are:

- Variable blocked format
- A maximum record length of 32 756
- A block size of 32 760
- Physical sequential format

OUTPUTDD=OUTPUT|filename(DEVT=TAPE |DISK) (VSE)

Specifies the file name that appears on the DLBL or TLBL JCL statement, a 1–7 character string containing only those alphanumeric characters supported in the operating environment. Characteristics of the file are:

- A block size of 32 760
- A maximum record length of 32 756

- Variable blocked format
- Assigned to programmer logical unit 007

OVLYLIB=*data set name1*[,*data set name2*][,*data set name...*] (MVS)

Specifies the data sets that compose the overlay library. A maximum of 8 data sets can be specified. The parameter also specifies the concatenation sequence when ACIF searches for a particular overlay resource. ACIF first looks for the resource in *data set name1*. If ACIF cannot find the resource in *data set name1*, it continues the search with *data set name2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

EXAMPLE

```
OVLYLIB=SYS1.OVLYLIB,USER.OVLYLIB
```

If **USERLIB** is also specified, ACIF searches for the resource in the data sets specified in **USERLIB** before searching the data sets identified in **OVLYLIB**.

Notes:

1. Data sets must be specified as fully-qualified names without quotation marks.
2. If the libraries specified for **OVLYLIB** are not specified in the same order used by the PSF startup procedure, the printed and converted results may differ. For information on how PSF selects resources, refer to *Print Services Facility/MVS: System Programming Guide*.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
4. This is a required parameter if overlay retrieval is requested. The **RESTYPE** value determines whether overlays are to be retrieved for inclusion in the resource data set. If this parameter is not specified, and overlay retrieval is requested, ACIF reports an error condition and terminates processing.

OVLYLIB=*filetype1*[,*filetype2*][,*filetype...*] (VM)

Specifies the file types that define the overlay libraries. A maximum of 8 file types can be specified. The parameter also specifies the search order when ACIF searches for a particular overlay resource. ACIF first looks for the resource with a file type of *filetype1*. If ACIF cannot find the resource with a file type of *filetype1*, it continues the search with *filetype2*, and so on, until it either locates the requested resource or exhausts the list of specified files.

EXAMPLE

```
OVLYLIB=OVLY38PP,TEMPOVLY
```

Notes:

1. File types must conform to CMS naming conventions.
2. This is a required parameter if overlay retrieval is requested. The **RESTYPE** value determines whether overlays are to be retrieved for inclusion in the resource file. If this parameter is not specified, and overlay retrieval is requested, ACIF reports an error condition and terminates processing.

OVLYLIB

This parameter is not used for **VSE**. Overlay resources are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement. For

information on how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

PAGEDEF=*pdefname*

Specifies the complete file name (in MVS and VSE, the member name) of the page definition, which defines the page format that ACIF uses to compose line data into pages. The *pdefname* can be from 1–8 alphanumeric or national characters. Unlike PSF/MVS, PSF/VM, and PSF/VSE, ACIF does **not** require the name to begin with a P1 prefix, but if the name does begin with P1, you cannot omit it.

EXAMPLE

The following example specifies *P1USER10* as the page definition:

```
PAGEDEF=P1USER10
```

ACIF does not support a parameter equivalent to the LINECT parameter on the /*JOBPARM, /*OUTPUT, and OUTPUT JCL statements. The maximum number of lines processed on a page is defined in the page definition. The page definition can be located:

- Inline in the file
- In a user library referenced in the **USERLIB** parameter (**MVS** only)
- In a library referenced in the **PDEFLIB** parameter
- In a library referenced in the // LIBDEF PHASE,SEARCH=(...) DLBL JCL statement (**VSE**)

Using Page Definitions from an MVS User Library: You can instruct ACIF to select a page definition from your user library instead of from a library referenced in the **PDEFLIB** parameter. To use a page definition from a user library, you must:

- Reference the user library containing the page definition in the **USERLIB** parameter.
- Specify the name of the page definition in the **PAGEDEF** parameter.

Using Inline Page Definitions: To use an inline page definition, ensure the following:

- Include an inline page definition in the input file.
- The name of the inline page definition must match the specified page definition name in the **PAGEDEF** parameter, or **PAGEDEF=DUMMY** must be specified.
- If a page definition resource is included inline with the data, the file must be identified as containing carriage control characters. If the length of the records in the page definition is less than or equal to the logical-record length defined for the file, you can specify fixed-length records for the record format. If the length of the records in the page definition is greater than the logical-record length defined for the file, you must specify:
 - In **MVS**, variable length records for the record format (variable blocked with ANSI [VBA] carriage control characters or variable blocked with machine [VBM] carriage control characters)
 - In **VM**, variable length records for the record format

- In **VSE**, variable length records for the record format (variable blocked with ANSI [VBA] carriage control characters or variable blocked with machine [VBM] carriage control characters)

You can include more than one inline page definition in an input file, and you can change the page definition name in the **PAGEDEF** parameter on different ACIF jobs to test different page definitions. If the name of an inline page definition does **not** match the **PAGEDEF** name specified, ACIF uses the page definition from the resource library that matches the name specified in the **PAGEDEF** parameter.

Notes:

1. If the **PAGEDEF** parameter is not specified, and the print file contains line-mode data, ACIF reports an error condition and terminates processing.
2. If you specify **PAGEDEF=DUMMY**, and you do not include an inline page definition, ACIF reports an error condition and terminates processing.
3. A page definition is required for processing line-mode data. ACIF reports an error condition and terminates processing if it encounters a line-mode record, but no page definition was specified in a **PAGEDEF** parameter.

PDEFLIB=*data set name1* [, *data set name2*] [, *data set name...*] (MVS)

Specifies the files that compose the page definition library. A maximum of 8 files can be specified. The parameter also specifies the concatenation sequence when ACIF searches for a particular page definition. ACIF first looks for the resource in *data set name1*. If ACIF cannot find the resource in *data set name1*, it continues the search with *data set name2*, and so on, until it either locates the requested resource or exhausts the list of specified files.

EXAMPLE

```
PDEFLIB=SYS1.PDEFLIB,USER.PDEFLIB
```

If **USERLIB** is also specified, ACIF searches for the resource in the files specified in **USERLIB** before searching the files identified in **PDEFLIB**.

Notes:

1. The files must be specified as fully-qualified names without quotation marks.
2. If the libraries specified for **PDEFLIB** are not specified in the same order used by the PSF startup procedure, the printed and converted results may differ. For information on how PSF selects resources, refer to *Print Services Facility/MVS: System Programming Guide*.
3. For systems before MVS/DFP Version 2.3, files must be concatenated with the largest block size first.
4. This is a required parameter if the file contains any line-mode data. If this parameter is not specified, and the file contains any line-mode data, ACIF reports an error condition and terminates processing.

PDEFLIB=*filetype1* [, *filetype2*] [, *filetype...*] (VM)

Specifies the file types that define the page- definition libraries. A maximum of 8 file types can be specified. The parameter also specifies the search order when ACIF searches for a particular PAGEDEF resource. ACIF first looks for the resource with a file type of *filetype1*. If ACIF cannot find the resource with

a file type of *filetype1*, it continues the search with *filetype2*, and so on, until it either locates the requested resource or exhausts the list of specified files.

EXAMPLE

```
PDEFLIB=PDEF38PP,TESTPDEF
```

Notes:

1. The file types must conform to CMS naming conventions.
2. This is a required parameter if the print file contains any line-mode data. If this parameter is not specified, and the print file contains any line-mode data, ACIF reports an error condition and terminates processing.

PDEFLIB

This parameter is not used for **VSE**. Page-definition resources are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement. For information on how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

PRMODE=SOSI1|SOSI2|aaaaaaaa

Specifies the type of data in the input file and whether ACIF must perform optional processing of that data.

SOSI1

Specifies that each shift out, shift in code be converted to a blank and a Set Coded Font Local text control. The SOSI1 conversion by ACIF is the same as that performed by PSF/MVS, PSF/VM, and PSF/VSE.

SOSI2

Specifies that each shift out, shift in code be converted to a Set Coded Font Local text control. The SOSI2 conversion by ACIF is the same as that performed by PSF/MVS, PSF/VM, and PSF/VSE.

aaaaaaaa

Specifies any 8-byte alphanumeric string. This value is supplied by all the ACIF user exits.

EXAMPLE

This example specifies that the SOSI1 process mode be set up for a file.

```
PRMODE=SOSI1
```

For the process to work correctly, the first font specified in the CHARS parameter (or in a font list in a page definition) must be a single-byte font, and the second font must be a double-byte font.

PSEGLIB=*data set name1*[,*data set name2*][,*data set name...*] (MVS)

Specifies the data sets that compose the page segment library. A maximum of 8 data sets can be specified. The parameter also specifies the concatenation sequence when ACIF searches for a particular page segment. ACIF first looks for the resource in *data set name1*. If it cannot find the resource in *data set name1*, it continues the search with *data set name2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

EXAMPLE

```
PSEGLIB=SYS1.PSEGLIB,USER.PSEGLIB
```

If **USERLIB** is also specified, ACIF searches for the resource in the files specified in **USERLIB** before searching the files identified in **PSEGLIB**.

Notes:

1. The data sets must be specified as fully-qualified names without quotation marks.
2. If the libraries specified for **PSEGLIB** are not specified in the same order used by the PSF startup procedure, the printed and converted results may differ. For information on how PSF selects resources, refer to *Print Services Facility/MVS: System Programming Guide*.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
4. This is a required parameter if page segment retrieval is requested. The **RESTYPE** value determines whether page segments are to be retrieved for inclusion in the resource data set. If this parameter is not specified, and page segment retrieval is requested, ACIF reports an error condition and terminates processing.

PSEGLIB=*filetype1*[,*filetype2*][,*filetype...*] (VM)

Specifies the file types that define the page segment libraries. A maximum of 8 file types can be specified. It also specifies the search order when ACIF searches for a particular page segment resource. ACIF first looks for the resource with a file type of *filetype1*. If it cannot find the resource with a file type of *filetype1*, it continues the search with *filetype2*, and so on, until it either locates the requested resource or exhausts the list of specified files.

EXAMPLE

```
PSEGLIB=PSEG38PP,PSEGTEST
```

Notes:

1. The file types must conform to CMS naming conventions.
2. This is a required parameter if page segment retrieval is requested. The **RESTYPE** value determines whether page segments are to be retrieved for inclusion in the resource file. If this parameter is not specified, and page segment retrieval is requested, ACIF reports an error condition and terminates processing.

PSEGLIB

This parameter is not used for **VSE**. Page-segment resources are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement. For information on how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

RESEXIT=*module name*

Specifies the name of the resource exit program. This is a 1–8 byte character name of the load module ACIF loads during initialization and subsequently calls each time it attempts to retrieve a requested resource from a library. If this parameter is not specified, no resource exit is used. See “Resource Exit” on page 126 for more detailed information.

RESFILE=SEQ | PDS

Specifies the format of the resource file (**MVS** only) created by ACIF. ACIF can create either a sequential data set (SEQ) or a partitioned data set (PDS) from the resources it retrieves from the PSF/MVS resource libraries. If this parameter is not specified, ACIF writes to the DDname specified in the **RESOBJDD** parameter, assuming a sequential format. See “Format of the

Resources File” on page 183 for more information on the contents of the resource data set.

Specifying **SEQ** creates a resource group that can be concatenated to the document file as inline resources. Specifying **PDS** creates a member that can be placed in a user library or in a system library for use by PSF. The file created by selecting **PDS** cannot be concatenated to the document file and used as inline resources.

RESOBJDD=RESOBJ | DDname (MVS and VM)

Specifies the DDname for the resource file. When ACIF processes a print file, it can optionally create a file containing all or some of the resources required to print or view the file. It writes the resource data to this DDname. If **RESOBJDD** is not specified, ACIF uses RESOBJ as the default DDname. *DDname* is a 1–8 byte character string containing only those alphanumeric characters supported in the operating environment. Suggested DCB characteristics for the file are:

- Variable blocked format
- A maximum record length of 32 756
- A block size of 32 760
- Physical, sequential format

RESOBJDD=RESOBJ | filename (DEV=TAPE|DISK) (VSE)

Specifies the file name that appears on the DLBL or TLBL JCL statement, a 1–7 character string containing only those alphanumeric characters supported in the operating environment. The characteristics of the file are:

- A variable blocked file
- A maximum record length of 32 756
- A block size of 32 760
- Assigned to programmer logical unit 008

RESTYPE=NONE | [FDEF[, PSEG][,OVLY][,FONT] | ALL

Specifies the type of AFP print resources ACIF should retrieve from the resource libraries for inclusion in the resource file (**RESOBJDD**). The following is list of the supported values.

ALL

Specifies that all resources required to print or view the output document file (**OUTPUTDD**) be included in the resource file (**RESOBJDD**).

FDEF

Specifies that the form definition (FORMDEF), used in processing the file, be included in the resource file.

PSEG

Specifies that all page segments required to print or view the output document file be included in the resource file.

OBJCON

Specifies that all object container files requested by the input data stream (including the one specified by the **COMSETUP** parameter) be included in the resource file.

OVLY

Specifies that all overlays required to print or view the output document file be included in the resource file.

FONT

Specifies that all font character sets and code pages required to print or view the output file be included in the resource file.

Note: Coded fonts are never included in the resource file.

NONE

Specifies that no resource file be created.

ACIF supports the specification of FDEF, FONT, OVLY, and PSEG in any combination. The following example illustrates this.

```
RESTYPE=FDEF,PSEG,OVLY          /* Don't want any fonts saved
```

Because the Viewer application of AFP Workbench does not use AFP raster fonts when presenting the data on the screen, you may want to specify **RESTYPE=FDEF,OVLY,PSEG** to prevent fonts from being included in the resource file. This reduces the number of bytes transmitted when the file is transferred from the host to the workstation.

TRACE=YES | NO

Specifies that ACIF should provide trace (diagnostic) information while processing the file. If **TRACE=YES** is specified, ACIF uses the facilities of the MVS Generalized Trace Facility (GTF) to produce diagnostic trace records. ACIF writes GTF trace records with a user event id of X'314'. To capture ACIF GTF records, GTF will need to be started with the option TRACE=USRP, and subsequently modified with USR=(314). Tracing increases processor overhead and should be turned off unless you need to do problem determination. To activate tracing, GTF must be started (contact your systems programmer for information on starting GTF), and **TRACE=YES** must be specified in the ACIF processing parameter file. To activate tracing, **TRACEDD** and **TRACE=YES** must be specified in the ACIF processing parameter file.

TRACEDD=TRACE|*ddname* (VM)

Specifies the DDname for the trace file. *ddname* is a 1–8 byte character string containing only those alphanumeric characters supported in the VM operating environment. If TRACE=YES is specified, ACIF writes diagnostic information to this DDname. If TRACEDD is not specified, ACIF uses TRACE as the default DDname.

TRACEDD=TRACE|*filename* (DEVT=TAPE|DISK) (VSE)

Specifies the file name that appears on the DLBL or TLBL JCL statement, a 1–7 character string containing only those alphanumeric characters supported in the operating environment. The characteristics of the file are:

- A variable blocked file
- A maximum record length of 32756
- A block size of 32760
- Assigned to programmer logical unit 014

TRC=YES | NO

Specifies whether the input file contains table reference characters (TRCs).

In line data, you can use different fonts on different lines of a file by specifying a TRC at the beginning of each line after the carriage control character, if one is present.

EXAMPLE

The following example specifies that TRCs are present:

```
TRC=YES
```

Notes:

1. The order in which the fonts are specified in the **CHARS** parameter establishes which number is assigned to each associated TRC. For example, the first font specified is assigned 0, the second font 1, and so on.
2. If **TRC=YES** is specified, but no TRCs are contained in the file, the first character (or second if carriage control characters are used) of each line is interpreted as the font identifier. Consequently, the font used to process each line of the file may not be the one you expect, and one byte of data will be lost from each record.
3. If you do not specify **TRC=YES** in the SYSIN parameter file, but your line data contains a TRC as the first character (or second if carriage control characters are used) of each line, the TRC is processed as a text character in the processed output rather than being used as a font identifier.

TRIGGERn={*record* | *},{*column* | *},{*'value'* | *X' value'*}

Specifies the locations and values of data fields within the input file that are to be used to define indexing groups in the file. These data fields are referred to as “eye catchers” or triggers, because their presence in the file triggers a processing action. A maximum of 4 **TRIGGER** parameters can be specified. The number of **TRIGGERn** parameters required to uniquely identify the beginning of a group of pages within the file is a function of the complexity of the application output. **TRIGGER1** is special, and each record in the file containing this value is referred to as an indexing anchor record. The presence of a **TRIGGER** parameter causes ACIF to index the input file. Each **TRIGGERn** parameter comprises three values:

record | *

Specifies the relative record number from the indexing anchor record (that is, **TRIGGER1**). A value of “*” can be specified only in **TRIGGER1**, and it indicates that every record should be checked for the presence of the **TRIGGER1** value. After the **TRIGGER1** value has been found, all other **TRIGGERn** parameter values are specified as a relative offset from **TRIGGER1**. ACIF reports an error condition and terminates processing if “*” is specified with any **TRIGGERn** parameter other than **TRIGGER1**. The supported range of values for *record* is 0–255.

column | *

Specifies the byte offset from the beginning of the record where the **TRIGGER** value is located. This value can be specified in absolute terms (for example, 10) or as “*,” which results in ACIF scanning the record from left to right, looking for the **TRIGGER** value. A value of 1 refers to the first byte in the record. For files containing carriage control characters, column 1 refers to the carriage control character. The supported range of values for *column* is 1–32756. ACIF compares the **TRIGGER** value to the input data. If the specified value exceeds the physical length of the record, ACIF considers the comparison “false” and continues processing.

'trigger value' | X' trigger value'

Specifies the actual alphanumeric (case-sensitive) or hexadecimal value of the **TRIGGER**. ACIF does not perform any validity checking on this value, but uses it in performing a byte-for-byte comparison with the records in the file. The **TRIGGER** value can be from 1–250 bytes in length. If the combined values of *column* and the **TRIGGER** length exceed the physical length of the record, ACIF considers the comparison “false” and continues processing.

EXAMPLE

The following example illustrates the use of a carriage control character as a trigger.

```
TRIGGER1=*,1,X'F1'           /* Look for Skip-to-Channel 1
TRIGGER2=0,50,'ACCOUNT:'     /* Find account number
TRIGGER3=3,75,'PAGE 1'      /* Find page 1
```

In this example, **TRIGGER1** instructs ACIF to scan every record, looking for the occurrence of X'F1' in the first byte. After ACIF locates a record containing the X'F1', it looks in the same record, starting at byte 50, for the occurrence of 'ACCOUNT:'. If 'ACCOUNT:' is found, ACIF looks at the third record down for a value of 'PAGE 1', starting at byte 75. If 'PAGE 1' is found, ACIF defines the record containing **TRIGGER1** as the indexing anchor record, and all indexing information is specified as relative locations from this point.

If ACIF finds either 'ACCOUNT:' or 'PAGE 1', it begins scanning the first record after the farthest field specified. If either 'ACCOUNT:' or 'PAGE 1' is not found at its specified location relative to **TRIGGER1**, ACIF begins looking for **TRIGGER1** again, starting with the next record (that is, the current record containing **TRIGGER1** + 1).

Notes:

1. ACIF requires that at least one **TRIGGERn** or **FIELDn** value appear within the page range specified by the **INDEXSTARTBY** parameter. If no **TRIGGERn** or **FIELDn** parameter is satisfied within the **INDEXSTARTBY** page range, ACIF stops processing.
2. At least one **TRIGGER** or **FIELDn** must exist on the first page of every unique group. ACIF cannot detect an error condition if **TRIGGERn** or **FIELDn** is missing, but the output may be incorrectly indexed.
3. **TRIGGER1** must be specified when ACIF is requested to index the file.
4. An error condition occurs if you specify any **TRIGGER** parameters, if the input file contains indexing tags.

UNIQUEBNGS={YES | NO}

Specifies whether ACIF creates a unique group name by generating an 8-character numeric string and appending the string to the group name. The group name defined in the Begin Named Page Group (BNG) structured field is comprised of an index value and a sequence number.

YES Specifies that ACIF generate an 8-character numeric string and append the string to the group name.

NO ACIF does not generate the string. No is the default if you specify **DCFPAGENAMES=YES**. Specify **NO** if you use the AFP API to generate your own group names.

USERLIB=*data set name1*[,*data set name2*][,*file name...*] (MVS)

Specifies the name of 1 to 8 catalogued data sets containing AFP resources for processing the input data set. ACIF dynamically allocates these data sets and searches for resources in them in the order specified in the **USERLIB** parameter. If a resource is not found, ACIF searches the appropriate resource libraries defined for that resource type (for example, **PDEFLIB** for page definitions). The libraries you specify can contain any AFP resources (fonts, page segments, overlays, page definitions, or form definitions). If Resource Access Control Facility (RACF) is installed on your system, RACF checks the authority of the USER ID requesting access to a user library (data set). If ACIF is not authorized to allocate the data set, it reports an error condition and terminates processing.

EXAMPLE

```
USERLIB=USER.IMAGES,USER.AFP.RESOURCES
```

Notes:

1. Because AFP resources (except page segments) have reserved prefixes, naming conflicts should not occur.
2. An inline resource overrides a resource of the same name contained in a **USERLIB** parameter.
3. The files must be specified as fully-qualified names without quotation marks.
4. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.

Chapter 8. Example: ACIF Application in MVS, VM, or VSE

In this example, a line-data application generates telephone bills, as shown in Figure 29 on page 112. The objective is to make the billing application output available on customer service representatives' workstations, so that when a customer calls with a billing inquiry, the representative can view the bill in the same format as the customer's printed copy.

To achieve this objective, you must convert the output from the application into a document format that can be used with the Viewer application of AFP Workbench (hereafter called Viewer). You also need to index the file to facilitate searching the file with Viewer. To ensure that all resources used in the bills are available at the workstation, you need to use the resource retrieval function of ACIF.

The tasks are:

- Examine the input file to determine how to tag it for indexing.
- Create ACIF parameters for indexing and retrieving resources.
- Determine the names of the resource libraries used when the bills are printed.
- Determine the form definition and page definition used to print the bills.
- Create JCL for the ACIF job.
- Run the ACIF job.
- Concatenate the index object file and the resource file to the document file.
- Transfer the document file in binary format to your workstation for viewing with Viewer.

For an explanation of any parameter not specifically described in this section, see the description for that parameter in Chapter 7, "Using ACIF Parameters in MVS, VM, and VSE" on page 87.

Return this portion with your payment.

Make check payable to

OUNTAIN COMMUNICATIONS

WILLIAM R. SMITH
5280 SUNSHINE CANYON DR
BOULDER CO 80000-0000

TOTAL AMOUNT DUE: \$56.97
DATE DUE: JAN 29, 1993

1 BASIC SERVICE \$30.56
2 LONG DISTANCE CHARGES \$26.41
TOTAL \$56.97

OUNTAIN COMMUNICATIONS
BOULDER CO 80000-0000

BILL DATE: JAN 11, 1993
ACCOUNT NUMBER: 303-222-3456-6B

PREVIOUS BILL \$66.79	PAYMENT \$66.79	ADJUSTMENTS \$0.00	PAST DUE DISREGARD IF PAID	\$0.00
--------------------------	--------------------	-----------------------	-------------------------------	--------

THANK YOU FOR YOUR PAYMENT	CURRENT CHARGES	\$56.97
	DATE DUE	JAN 29, 1993
	AMOUNT DUE	\$56.97

SUMMARY OF CURRENT CHARGES

RESIDENCE SERVICE	\$25.07
911 SURCHARGE	\$0.50
CUSTOMER ACCESS SERVICE	\$3.50
WIRING MAINTENANCE PLAN	\$0.50
FEDERAL EXCISE TAX	\$0.50
STATE TAX	\$0.49
LONG DISTANCE CHARGES (ITEMIZED BELOW)	\$26.41

LONG DISTANCE CHARGES

NO.	DATE	TIME	TO PLACE	TO AREA NUMBER	MINUTES	AMOUNT
1	DEC 11	7:15P	LOVELAND CO	303 666-7777	006	\$0.82
2	DEC 15	9:16A	NIWOT CO	303 555-6666	012	\$1.56
3	DEC 24	9:32P	SANTA BARBARA CA	805 999-2222	032	\$15.80
4	DEC 25	2:18P	LAS VEGAS NV	702 888-7654	015	\$8.23
TOTAL						\$26.41

PAGE 1

Figure 29. Example of a Customer's Phone Bill

Input File

Figure 30 shows the file currently used to print the bills.

```
Carriage
Control
|
Line 0 1-----1-----2-----3-----4-----5-----6-----7-----8
0 1 WILLIAM R. SMITH
    5280 SUNSHINE CANYON DR
    BOULDER CO 80000-0000
    TOTAL AMOUNT DUE: $56.97
    DATE DUE: JAN 29, 1993
5
-
-
0 1 BASIC SERVICE. . . . . $30.56
10 0 2 LONG DISTANCE CHARGES . . . . . $26.41
    TOTAL . . . . . $56.97
-
0
    BILL DATE: JAN 11, 1993
    ACCOUNT NUMBER: 303-222-3456-68
15 - $66.79          $66.79          $0.00          $0.00
    $56.97
    JAN 29, 1993
    $56.97
-
20 0 SUMMARY OF CURRENT CHARGES
    0 RESIDENCE SERVICE $25.07
    911 SURCHARGE $0.50
    CUSTOMER ACCESS SERVICE $3.50
    WIRING MAINTENANCE PLAN $0.50
25 FEDERAL EXCISE TAX $0.50
    STATE TAX $0.49
    LONG DISTANCE CHARGES (ITEMIZED BELOW) $30.56
    0 LONG DISTANCE CHARGES
    0 NO. DATE TIME TO PLACE TO AREA NUMBER MINUTES AMOUNT
30 0 1 DEC 11 7:15P LOVELAND CO 303 666-7777 006 $0.82
    2 DEC 15 9:16A NINOT CO 303 555-6666 012 $1.56
    3 DEC 24 9:32P SANTA BARBARA CA 805 999-6666 032 $15.80
    4 DEC 25 2:18P LAS VEGAS NV 702 888-7654 015 $8.23
    TOTAL . . . . . $26.41
35
-
-
0 PAGE 1
```

Figure 30. Line-Data Phone Bill

JCL, CMS Commands, and ACIF Processing Parameters

The next three figures show the JCL, CMS commands, and ACIF processing parameters used to invoke the ACIF program to index the input file. Figure 31 on page 114 shows the MVS JCL, Figure 32 on page 115 shows the CMS commands, and Figure 33 on page 116 shows the VSE JCL.

After you concatenate the files, transfer the concatenated files to the workstation in binary format. When you open the file for viewing with Viewer, you can select groups labeled:

- Account Number
- Name
- Address
- City, State, Zip
- Date due

MVS JCL to Invoke ACIF

```

//job... JOB ...
//APKSMAN EXEC PGM=APKACIF,REGION=8M,TIME=(,30)
//*****
/* RUN APK, CREATING OUTPUT AND A RESOURCE LIBRARY *
//*****
//STEPLIB DD DSN=APKACIF.LOAD,DISP=SHR
//INPUT DD DSN=USER.ACIFEX2.DATA,DISP=SHR
//SYSIN DD *

CC = YES /* example phone bill */
CCTYPE = A /* carriage control used */
CHARS = GT15 /* carriage control type */
CPGID = 500 /* code page identifier */
FDEFLIB = SYS1.FDEFLIB
FIELD1 = 13,66,15 /* Account Number */
FIELD2 = 0,50,30 /* Name */
FIELD3 = 1,50,30 /* Address */
FIELD4 = 2,50,30 /* City, State, Zip */
FIELD5 = 4,60,12 /* Date Due */
FONTLIB = SYS1.FONTLIBB,SYS1.FONTLIBB.EXTRA
FORMDEF = F1A10110 /* formdef name */
INDEX1 = 'Account Number',field1 /* 1st INDEX attribute */
INDEX2 = 'Name',field2 /* 2nd INDEX attribute */
INDEX3 = 'Address',field3 /* 3rd INDEX attribute */
INDEX4 = 'City, State, Zip',field4 /* 4th INDEX attribute */
INDEX5 = 'Date Due',field5 /* 5th INDEX attribute */
INDEXOBJ = ALL
INDEXDD = INDEX /* index file ddname */
INPUTDD = INPUT /* input file ddname */
OUTPUTDD = OUTPUT /* output file ddname */
OVLYLIB = SYS1.OVERLIB
PAGEDEF = P1A08682 /* pagedef name */
PDEFLIB = SYS1.PDEFLIB
PSEGLIB = SYS1.PSEGLIB
RESFILE = SEQ /* resource file type */
RESOBJDD = RESLIB /* resource file ddname */
RESTYPE = FDEF,PSEG,OVLY /* resource type selection */
TRIGGER1 = *,1,'1' /* 1st TRIGGER */
TRIGGER2 = 13,50,'ACCOUNT NUMBER:' /* 2nd TRIGGER */
/*
//OUTPUT DD DSN=APKACIF.OUTPUT,DISP=(NEW,CATLG),
// SPACE=(32760,(150,150),RLSE),UNIT=SYSDA,
// DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBM,DSORG=PS)
//INDEX DD DSN=APKACIF.INDEX,DISP=(NEW,CATLG),
// SPACE=(32760,(15,15),RLSE),UNIT=SYSDA,
// DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBM,DSORG=PS)
//RESLIB DD DSN=APKACIF.RESLIB,DISP=(NEW,CATLG),
// SPACE=(12288,(150,15),RLSE),UNIT=SYSDA,
// DCB=(LRECL=12284,BLKSIZE=12288,RECFM=VBM,DSORG=PS)
//SYSPRINT DD DSN=APKACIF.SYSPRINT,DISP=(NEW,CATLG),
// SPACE=(9044,(15,15),RLSE),UNIT=SYSDA,
// DCB=(BLKSIZE=141,RECFM=FB,DSORG=PS)

```

Figure 31. Example of a Telephone Bill for an MVS ACIF Application

VM CMS Commands to Invoke ACIF

```

FILEDEF INPUT DISK ACIFEX2 SYSIN A
FILEDEF OUTPUT DISK APKACIF OUTPUT A (LRECL 32756 BLKSIZE 32760 RECFM VB
FILEDEF INDEX DISK APKACIF INDEX A (LRECL 32756 BLKSIZE 32760 RECFM VB
FILEDEF RESLIB DISK APKACIF RESLIB A (LRECL 32756 BLKSIZE 32760 RECFM VB
FILEDEF SYSPRINT DISK APKACIF SYSPRINT A
APKACIF

```

Where file ACIFEX2 SYSIN A contains the following:

```

/* Pubs example phone bill */
CC = YES /* carriage control used */
CCTYPE = A /* carriage control type */
CHARS = GT15
CPGID = 500 /* code page identifier */
FDEFLIB = FDEF3820,FDEF38PP
FIELD1 = 13,66,15 /* Account Number */
FIELD2 = 0,50,30 /* Name */
FIELD3 = 1,50,30 /* Address */
FIELD4 = 2,50,30 /* City, State, Zip */
FIELD5 = 4,60,12 /* Date Due */
FONTLIB = FONT3820,FONT38PP
FORMDEF = F1A10110 /* formdef name */
INDEX1 = 'Account Number',field1 /* 1st INDEX */
INDEX2 = 'Name',field2 /* 2nd INDEX */
INDEX3 = 'Address',field3 /* 3rd INDEX */
INDEX4 = 'City, State, Zip',field4 /* 4th INDEX */
INDEX5 = 'Date Due',field5 /* 5th INDEX */
INDEXDD = INDEX /* index file ddname */
INPUTDD = INPUT /* input file ddname */
OUTPUTDD = OUTPUT /* output file ddname */
OVLYLIB = OVLY3820,OVLY38PP
PAGEDEF = P1A08682 /* pagedef name */
PDEFLIB = PDEF3820,PDEF38PP
PSEGLIB = PSEG3820,PSEG38PP
RESFILE = SEQ /* resource file type */
RESOBJDD = RESLIB /* resource file ddname */
RESTYPE = FDEF,PSEG,OVLY /* resource type selection */
TRIGGER1 = *,1,'1' /* 1st TRIGGER */
TRIGGER2 = 13,50,'ACCOUNT NUMBER:' /* 2nd TRIGGER */

```

Figure 32. Example of a Telephone Bill for a VM ACIF Application

VSE JCL to Invoke ACIF

```
// JOB
// LIBDEF PHASE,SEARCH=(PRD2.AFP)
// ASSGN SYSLST,X'FEE'
// ASSGN SYS006,201
// DLBL INPUT,'APKACIF.INPUT',0,SD
// EXTENT SYS006,SYSWK1,1,1,9200,13
// ASSGN SYS007,201
// DLBL OUTPUT,'APKACIF.OUTPUT',0,SD
// EXTENT SYS007,SYSWK1,1,1,9213,45
// ASSGN SYS008,201
// DLBL RESOBJ,'APKACIF.RESLIB',0,SD
// EXTENT SYS008,SYSWK1,1,1,9258,15
// ASSGN SYS009,201
// DLBL INDEX,'APKACIF.INDEX',0,SD
// EXTENT SYS009,SYSWK1,1,1,9273,15
// EXEC PGM=APKACIF,SIZE=548K

CC = YES /* carriage control used */
CCTYPE = A /* carriage control type */
CHARS = GT15
CPGID = 500 /* code page identifier */
FIELD1 = 13,66,15 /* Account Number */
FIELD2 = 0,50,30 /* Name */
FIELD3 = 1,50,30 /* Address */
FIELD4 = 2,50,30 /* City, State, Zip */
FIELD5 = 4,60,12 /* Date Due */
FORMDEF = F1A10110 /* formdef name */
INDEX1 = 'Account Number',field1 /* 1st INDEX */
INDEX2 = 'Name',field2 /* 2nd INDEX */
INDEX3 = 'Address',field3 /* 3rd INDEX */
INDEX4 = 'City, State, Zip',field4 /* 4th INDEX */
INDEX5 = 'Date Due',field5 /* 5th INDEX */
INDEXDD = INDEX /* index file ddname */
INPUTDD = INPUT /* input file ddname */
OUTPUTDD = OUTPUT /* output file ddname */
PAGEDEF = P1A08682 /* pagedef name */
RESFILE = SEQ /* resource file type */
RESOBJDD = RESOBJ /* resource file ddname */
RESTYPE = FDEF,PSEG,OVLY /* resource type selection */
TRIGGER1 = *,1,'1' /* 1st TRIGGER */
TRIGGER2 = 13,50,'ACCOUNT NUMBER:' /* 2nd TRIGGER */
/*
/ &
```

Figure 33. Example of a Telephone Bill for a VSE ACIF Application

The following data values are used as the indexing attributes:

- Account number
- Name
- Address
- City, State, Zip
- Date due

The task is to specify the ACIF indexing parameters so that the first page of each bill includes group-level indexing tags containing the values of all five of these attributes.

To generate these indexing attributes, specify the **TRIGGER1** parameter first, because ACIF always scans for the data specified in **TRIGGER1** first. Because the data contains carriage control characters, including a carriage control of 1 to indicate a new page, request that ACIF locate the start of a page by searching every record in the file for a trigger value of '1' in column 1 of the data. To do this, specify:

```
TRIGGER1 = *,1,'1'
```

When ACIF finds a record that contains a '1' in column 1, that record becomes the indexing anchor record.

Subsequent **TRIGGERn** parameters are defined relative to the indexing anchor record. In this example, you want to ensure that the page being indexed is the first page of the bill, which is the only page in the bill that has the text 'ACCOUNT NUMBER:' starting at byte 50 in the 13th record following the anchor record. To specify this additional trigger for locating the correct page to index, specify:

```
TRIGGER2 = 13,50,'ACCOUNT NUMBER:'
```

ACIF uses both trigger values to locate a place in the file to begin searching for the data supplied in the **INDEX** parameters.

Next, specify the attribute name of the first indexing parameter as 'Account Number', and define the location of the attribute value in the data relative to the index anchor record set by **TRIGGER1**. Because the data value for the Account Number attribute is located in the 13th record from the index anchor record starting in byte 66 and extending for 15 bytes, specify:

```
FIELD1=13,66,15  
INDEX1='Account Number',field1
```

To create the indexing tag for the Name attribute, define 'Name' as the indexing attribute. Locate the value for Name in the anchor record in the data starting at byte 50 and extending for 30 bytes. The ACIF parameters to specify this are:

```
FIELD2=0,50,30  
INDEX2='Name',field2
```


Repeat this process to specify the other three indexing tags, so that the index attributes and values are defined as follows:

INDEX1 'Account Number',field1

'Account Number' index attribute

field1 maps to FIELD1 index value

FIELD1 13 lines down from indexing anchor record, 66 columns across, 15 bytes in length

INDEX2 'Name', field2

'Name' index attribute

field2 maps to FIELD2 index value

FIELD2 0 lines down (in indexing anchor record), 50 columns across, 30 bytes in length

INDEX3 'Address', field3

'Address' index attribute

field3 maps to FIELD3 index value

FIELD3 1 line down from indexing anchor record, 50 columns across, 30 bytes in length

INDEX4 'City, State, Zip', field4

'City, State, Zip', index attribute

field4 maps to FIELD4 index value

FIELD4 2 lines down from indexing anchor record, 50 columns across, 30 bytes in length

INDEX5 'Date Due',field5

'Date Due' index attribute

field5 maps to FIELD5 index value

FIELD5 4 lines down from indexing anchor record, 60 columns across, 12 bytes in length

The result of using these indexing parameters is that the first page of each bill in the ACIF output file will contain indexing tags for each of the five indexing attributes. Using Viewer, customer service representatives can locate a single customer bill in the ACIF document using any combination of the indexing attributes.

The examples defined the following libraries:

Figure 34. Library Names

Library Name	MVS Name	VM Name (file type)	VSE Name
FDEFLIB Form definition library	SYS1.FDEFLIB	FDEF3820 FDEF38PP	PRD2.AFP
FONTLIB Font libraries	SYS1.FONTLIBB SYS1.FONTLIBB.EXTRA	FONT3820 FONT38PP	PRD2.AFP
OVLYLIB Overlay library	SYS1.OVERLIB	OVLY3820 OVLY38PP	PRD2.AFP
PDEFLIB Page definition library	SYS1.PDEFLIB	PDEF3820 PDEF38PP	PRD2.AFP
PSEGLIB Page segment library	SYS1.PSEGLIB	PSEG3820 PSEG38PP	PRD2.AFP

The examples defined the following resources:

FORMDEF F1A10110, a standard form definition

PAGEDEF P1A08682, a standard page definition

ACIF Output

The ACIF job creates the following output files:

Type of File	MVS	VM	VSE
Document file, including indexing structured fields	APKACIF.OUTPUT	APKACIF OUTPUT	APKACIF.OUTPUT
Index object file	APKACIF.INDEX	APKACIF INDEX	APKACIF.INDEX
Resource file	APKACIF.RESLIB	APKACIF RESLIB	APKACIF.RESLIB
Message file listing: <ul style="list-style-type: none">• ACIF parameters used• Resources used• Return code	APKACIF.SYSPRINT	APKACIF SYSPRINT	APKACIF.SYSPRINT

Concatenating Files

To view the document file on a workstation using Viewer, you must concatenate the index object file and the resource file to the document file.

MVS JCL

The following is an example of MVS JCL you can use to perform the concatenation:

```
//PRINT EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=APKACIF.INDEX,DISP=SHR
// DD DSN=APKACIF.RESLIB,DISP=SHR
// DD DSN=APKACIF.OUTPUT,DISP=SHR
//SYSUT2 DD DSN=NEW.PRINT.OBJECT,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(32760,nnn),
// DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBM)
```

Where *nnn* is equal to the size of the index object file, plus the size of the resource file, plus the size of the document file.

Note: The resource file must have been created by specifying **RESFILE=SEQ**.

VM CMS Commands

The following is an example of the VM CMS commands you can use to perform the concatenation:

```
COPY APKACIF INDEX A APKACIF RESLIB A APKACIF OUTPUT A APKACIF LIST3820 A (APPEND
```

Chapter 9. User Exits and Attributes of the Input Print File in MVS, VM, and VSE

This appendix contains Product-sensitive Programming Interface and Associated Guidance Information.

The appendix describes the four user exits provided with ACIF and describes the information ACIF provides to the exits about the attributes of the input print file.

User Programming Exits

ACIF provides four programming exits, so that an installation can customize the program. The exits are optional, and the names of the modules that can be loaded are specified in the processing parameter file.

ACIF provides the following four exits:

- Input record
- Index record
- Output record
- Resource

Input Record Exit

ACIF provides an exit that allows you to add, delete, or modify records in the input file. You can also use the exit to insert indexing information. The program invoked at this exit is defined in the ACIF **INPEXIT** parameter.

This exit is called after each record is read from the input file. The exit program can request that the record be discarded, processed, or processed with control returned to the exit for the next input record. The largest record that can be processed is 32 756 bytes. ACIF does **not** call this exit when ACIF is processing resources from libraries.

In a MO:DCA-P document, indexing information can be passed in the form of a Tag Logical Element (TLE) structured field. For more information about the TLE structured field, see Appendix B, "Data Stream Information." The exit program creates these structured fields while ACIF is processing the print file. This is an alternative to modifying the application in cases where the indexing information is not consistently present in the application output.

Note: TLEs are not supported in line-mode or mixed-mode data.

Figure 35 contains a sample DSECT that describes the control block for the exit program.

PARMLIST	DSECT		Parameters for the input record exit
WORK@	DS	A	Address of 16-byte static work area
PFATTR@	DS	A	Address of print-file-attribute information
RECORD@	DS	A	Address of the input record
	DS	A	Reserved for future use
RECORDLN	DS	H	Length of the input record
	DS	H	Reserved for future use
REQUEST	DS	X	Add, delete, or process the record
EOF	DS	C	EOF indicator

Figure 35. Sample Input Record Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit in a standard parameter list pointed to by register 1:

WORK@ (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ (Bytes 5–8)

A pointer to the print-file-attribute data structure. See “Attributes of the Input Print File” on page 129 for more information on the format of this data structure and the information it contains.

RECORD@ (Bytes 9–12)

A pointer to the first byte of the input record, including the carriage control character. The record is in a buffer that resides in storage allocated by ACIF, but the exit program is allowed to modify the input record.

RECORDLN (Bytes 17–18)

Specifies the number of bytes (length) of the input record. If the input record is modified, this parameter must also be updated to reflect the actual length of the record.

REQUEST (Byte 21)

Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00', X'01', or X'02', where:

X'00' Specifies that the record be processed by ACIF.

X'01' Specifies that the record not be processed by ACIF.

X'02' Specifies that the record be processed by ACIF with control returned to the exit program to allow it to insert the next record. The exit program can set this value to save the current record, insert a record, and then supply the saved record at the next call. After the exit inserts the last record, the exit program must reset the **REQUEST** byte to X'00'.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the **REQUEST** byte value to X'01'. If you want the record to be processed, and you want to insert an additional record, change the **REQUEST** byte value to X'02'. Any value greater than X'02' is interpreted as X'00', and the exit processes the record.

Note: Only one record can reside in the buffer at any time.

EOF (Byte 22)

Specifies an end-of-file (EOF) indicator. This indicator is a 1-byte character code that specifies whether an EOF condition has been encountered. When EOF is signalled (EOF value = "Y"), the last record has already been presented to the input exit, and the input file has been closed. The pointer RECORD@ is no longer valid. Records cannot be inserted when EOF is signalled. The following are the only valid values for this parameter:

- Y** Specifies that EOF has been encountered
- N** Specifies that EOF has not been encountered

This end-of-file indicator allows the exit program to perform some additional processing at the end of the print file. The exit program cannot change this parameter.

Index Record Exit

ACIF provides an exit that allows you to modify or ignore the records that ACIF writes into the index object file. The program invoked at this exit is defined by the ACIF **INDEXEXIT** parameter.

This exit receives control before a record (structured field) is written to the index object file. The exit program can request that the record be ignored or processed. The largest record that can be processed is 32752 bytes (this does not include the record descriptor word).

Figure 36 contains a sample DSECT that describes the control block that is passed to the exit program.

PARMLIST	DSECT		Parameters for the output record exit
WORK@	DS	A	Address of 16-byte static work area
PFATTR@	DS	A	Address of print-file-attribute information
RECORD@	DS	A	Address of the record to be written
RECORDLN	DS	H	Length of the output record
REQUEST	DS	X	Delete or process the record
EOF	DS	C	Last call indicator to ACIF

Figure 36. Sample Index Record Exit DSECT

The address of the control block containing the following parameters is passed to the index record exit in a standard parameter list pointed to by register 1.

WORK@ (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ (Bytes 5–8)

A pointer to the print-file-attribute data structure. See “Attributes of the Input Print File” on page 129 for more information on the format of this data structure and the information it contains.

RECORD@ (Bytes 9–12)

A pointer to the first byte of the index record, including the carriage control character. The record resides in a 32KB buffer (where KB equals 1024 bytes). The buffer resides in storage allocated by ACIF, but the exit program is allowed to modify the index record.

RECORDLN (Bytes 13–14)

Specifies the length, in bytes, of the index record. If the index record is modified, this parameter must also be updated to reflect the actual length of the record.

REQUEST (Byte 15)

Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01', where:

X'00' Specifies that the record be processed by ACIF.

X'01' Specifies that the record not be processed by ACIF.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the **REQUEST** byte value to X'01'. Any value greater than X'01' is interpreted as X'00', and the exit processes the record.

Note: Only one record can reside in the buffer at any time.

EOF (Byte 16)

Specifies an end-of-file (EOF) indicator. This indicator is a 1-byte character code that signals when ACIF has finished processing the index object file.

When EOF is signalled (EOF value = “Y”), the last record has already been presented to the index exit. The pointer **RECORD@** is no longer valid.

Records cannot be inserted when EOF is signalled. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

This end-of-file flag, used as a last call indicator, allows the exit program to return control to ACIF. The exit program cannot change this parameter.

Output Record Exit

Using the output record exit, you can modify or ignore the records ACIF writes into the output document file. The program invoked at this exit is defined by the ACIF **OUTEXIT** parameter.

The exit receives control before a record (structured field) is written to the output document file. The exit program can request that the record be ignored or processed. The largest record that the exit can process is 32752 bytes, not including the record descriptor word. The exit is not called when ACIF is processing resources.

Figure 37 contains a sample DSECT that describes the control block passed to the exit program.

PARMLIST	DSECT		Parameters for the output record exit
WORK@	DS	A	Address of 16-byte static work area
PFATTR@	DS	A	Address of print-file-attribute information
RECORD@	DS	A	Address of the record to be written
RECORDLN	DS	H	Length of the output record
REQUEST	DS	X	Delete or process the record
EOF	DS	C	Last call indicator

Figure 37. Sample Output Record Exit DSECT

The address of the control block containing the following parameters is passed to the output record exit in a standard parameter list pointed to by register 1:

WORK@ (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ (Bytes 5–8)

A pointer to the print-file-attribute data structure. See “Attributes of the Input Print File” on page 129 for more information on the format of this data structure and the information contained in it.

RECORD@ (Bytes 9–12)

A pointer to the first byte of the output record. The record resides in a 32KB buffer (where KB equals 1024 bytes). The buffer resides in storage allocated by ACIF, but the exit program is allowed to modify the output record.

RECORDLN (Bytes 13–14)

Specifies the length, in bytes, of the output record. If the output record is modified, this parameter must also be updated to reflect the actual length of the record.

REQUEST (Byte 15)

Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01', where:

X'00' Specifies that the record be processed by ACIF.

X'01' Specifies that the record be ignored by ACIF.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the **REQUEST** byte value to X'01'. Any value greater than X'01' is interpreted as X'00', and the exit processes the record.

Note: Only one record can reside in the buffer at any time.

EOF (Byte 16)

An end-of-file (EOF) indicator. This indicator is a 1-byte character code that signals when ACIF has finished writing the output file.

When EOF is signalled (EOF value = "Y"), the last record has already been presented to the output exit. The pointer RECORD@ is no longer valid.

Records cannot be inserted when EOF is signalled. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

This end-of-file flag, used as a last-call indicator, allows the exit program to return to ACIF. The exit program cannot change this parameter.

Resource Exit

ACIF provides an exit that enables you to "filter" resources from being included in the resource file. If you want to exclude a specific type of resource (for example, an overlay), you can do so with the **RESTYPE** parameter. This exit is useful in controlling resources at the file name level. For example, assume you are going to send ACIF output to PSF/2, and you want to send only those fonts that are not shipped with the PSF/2 product. You can code this exit program to contain a table of all fonts shipped with PSF/2 and filter those fonts from the resource file. Security is another consideration for using this exit, because you can prevent certain named resources from being included. The program invoked at this exit is defined by the ACIF **RESEXIT** parameter.

This exit receives control before a resource is read from a library. The exit program can request that the resource be processed or ignored (skipped), but it cannot substitute another resource name in place of the requested one. If the exit requests that any overlay be ignored, ACIF automatically ignores any resources the overlay references (that is, fonts and page segments).

Figure 22 on page 73 contains a sample DSECT that describes the control block that is passed to the exit program.

PARMLIST	DSECT		Parameters for the output record exit
WORK@	DS	A	Address of 16-byte static work area
PFATTR@	DS	A	Address of print-file-attribute information
RESNAME	DS	CL8	Name of requested resource
RESTYPE	DS	X	Type of resource
REQUEST	DS	X	Ignore or process the resource
EOF	DS	X	

Figure 38. Sample Resource Exit DSECT

The address of the control block containing the following parameters is passed to the resource record exit in a standard parameter list pointed to by register 1:

WORK@ (Bytes 1–4)

A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ (Bytes 5–8)

A pointer to the print-file-attribute data structure. See “Attributes of the Input Print File” on page 129 for more information on the format of this data structure and the information presented.

RESNAME (Bytes 9–16)

Specifies the name of the requested resource. This value cannot be modified (changed) by the exit program.

RESTYPE (Byte 17)

Specifies the type of resource to which the name refers. This is a 1-byte hexadecimal value, where:

X'40' Specifies a font character set.

X'41' Specifies a code page.

X'FB' Specifies a page segment.

X'FC' Specifies an overlay.

ACIF does **not** call this exit for the following resource types:

- Page definition

The page definition (PAGEDEF) is a required resource for processing line-mode application output. The page definition is never included in the resource file.

- Form definition

The form definition (FORMDEF) is a required resource for processing print files. If you do not want the form definition included in the resource file, specify **RESTYPE=NONE** or explicitly exclude it from the **RESTYPE** list.

- Coded fonts

ACIF does not include any referenced coded fonts in the resource file; therefore, resource filtering is not applicable. ACIF needs to process coded fonts to determine the names of the code pages and font character sets they reference, which is necessary to create MCF-2 structured fields.

REQUEST (Byte 18)

Specifies how the resource is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01', where:

X'00' Specifies that the resource be processed by ACIF.

X'01' Specifies that the resource not be processed by ACIF.

A value of X'00' on entry to the exit program specifies that the resource be processed. If you want to ignore the resource, change the **REQUEST** byte value to X'01'. Any value greater than X'01' is interpreted as X'00', and the exit processes the resource.

EOF (Byte 19)

An end-of-file (EOF) indicator. This indicator is a 1-byte character code that signals when ACIF has finished writing the resource file.

When EOF is signalled (EOF value = "Y"), the last record has already been presented to the resource exit. The pointer RECORD@ is no longer valid. Records cannot be inserted when EOF is signalled. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

This end-of-file flag, used as a last-call indicator, returns control to ACIF. The exit program cannot change this parameter.

User Exit Search Order

When ACIF loads a specified user exit program during initialization, the operating system determines the search order and method used to locate these load modules.

MVS

Exit load modules can reside in a load library used as STEPLIB, JOBLIB, or in a system library. ACIF uses the standard MVS search order to locate the exit load module; that is, it looks first in the STEPLIB, then in the JOBLIB, and finally in the system libraries.

VM

ACIF uses standard CMS search order to locate the specified user exit load module; that is, *name* **MODULE** *.

VSE

Exit load modules are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement.

Non-Zero Return Codes

If ACIF receives a non-zero return code from any exit, it issues message APK412I and terminates processing.

Attributes of the Input Print File

ACIF provides information about the attributes of the input print file in a data structure available to ACIF's user exits. Figure 39 shows the format of this data structure.

PFATTR	DSECT		Print File Attributes
CC	DS	CL3	Carriage controls? - 'YES' or 'NO '
CCTYPE	DS	CL1	Carriage control type - A (ANSI) or M (Machine)
CHARS	DS	CL20	CHARS values, including commas (eg. GT12, GT15)
FORMDEF	DS	CL8	Form Definition (FORMDEF)
PAGEDEF	DS	CL8	Page Definition (PAGEDEF)
PRMODE	DS	CL8	Processing mode
TRC	DS	CL3	Table Reference Characters - 'YES' or 'NO '

Figure 39. Sample Print File Attributes DSECT

The address of the control block containing the following parameters is passed to the exits in a standard parameter list pointed to by register 1:

CC (Bytes 1–3)

The value of the **CC** parameter as specified in the ACIF processing parameter file. ACIF uses the default value if this parameter is not explicitly specified.

CCTYPE (Byte 4)

The value of the **CCTYPE** parameter as specified in the ACIF processing parameter file. ACIF supplies the default value if this parameter is not explicitly specified.

CHARS (Bytes 5–24)

The value of the **CHARS** parameter as specified in the ACIF processing parameter file, including any commas that separate specifications of multiple fonts. Because the **CHARS** parameter has no default value, this field contains blanks if no values are specified.

FORMDEF (Bytes 25–32)

The value of the **FORMDEF** parameter as specified in the ACIF processing parameter file. Because the **FORMDEF** parameter has no default value, this field contains blanks if no value is specified.

PAGEDEF (Bytes 33–40)

The value of the **PAGEDEF** parameter as specified in the ACIF processing parameter file. Because the **PAGEDEF** parameter has no default value, this field contains blanks if no value is specified.

PRMODE (Bytes 41–48)

The value of the **PRMODE** parameter as specified in the ACIF processing parameter file. Because the **PRMODE** parameter has no default value, this field contains blanks if no value is specified.

TRC (Bytes 49–51)

The value of the **TRC** parameter as specified in the ACIF processing parameter file. ACIF supplies the default value if this parameter is not explicitly specified.

Notes:

1. Each of the above character values is left-justified, with padding blanks added to the right end of the string. For example, if **PAGEDEF=P1TEST** is specified in the ACIF processing parameter file, the page definition value in the above data structure is 'P1TESTbb'.
2. Exit programs cannot change the values supplied in this data structure. For example, if 'P1TESTbb' is the page definition value, and an exit program changes the value to 'P1PRODbb', ACIF still uses 'P1TESTbb'.
3. This data structure showing the attributes of the print file is provided for informational purposes only.

Chapter 10. ACIF Messages for MVS, VM, and VSE

ACIF prints a message list at the end of each compilation. A return code of 0 means that ACIF completed processing without any errors. ACIF supports the standard return codes.

Note: ACIF messages contain instructions for the PSF system programmer. Please show your system programmer these messages, because they are not contained in the PSF messages publication.

ACIF detects a number of error conditions, which can be logically grouped into three categories:

- **Severe**

ACIF terminates processing the current print file. The exact method of termination may vary. For certain severe errors, ACIF abends with a return code and reason code. This is generally the case when some system service fails. In other cases, ACIF terminates with the appropriate error messages written to the message file specified when you invoked ACIF. Most error conditions detected by ACIF fall into this category. Severe errors have an “S” suffix.

- **Warning**

ACIF issues warning messages when the fidelity of the document (assuming it is reprinted) may be in question. Warning errors have a “W” suffix.

- **Informational**

When ACIF processes a print file, it issues informational messages that allow the operator or application programmer to determine if the correct processing parameters have been specified. These messages can assist in providing an audit trail. Informational errors have an “I” suffix.

Multiple Message Scenarios

ACIF may issue more than one error message as a result of a single error condition. These situations are limited to the area of parsing the structured field (for example, determining the length and type of the structured field). Some possible scenarios include:

105, 108, 109, 103

105, 108, 110, 103

106, 108, 109, 103

106, 108, 110, 103

Any subset of the listed messages is also possible, provided you start with the first one (for example, 105, 108, 109 or 105, 108, or 105, 110, and so on). The first message accurately describes the error condition; any subsequent messages provide additional information. Additional error messages may not always be accurate.

Message 101 may occur after many error conditions, because ACIF attempts to locate the end of the resource containing the error as part of its recovery procedure.

General Messages

General error messages are not limited to a particular resource, which is why they are considered general error conditions. Some general errors are limited to a few resources, while others may occur in any resource.

APK0311 AN INLINE MEDIUM MAP WAS ENCOUNTERED IN THE DATA SET, BUT INLINE MEDIUM MAPS ARE NOT SUPPORTED.

Explanation: A Begin Medium Map (BMM) structured field was encountered in the data stream after resources for the data set had been processed. ACIF does not support inline medium maps between pages. The data set may have been created by a program that creates inline medium maps, but a data set that contains inline medium maps cannot be printed.

System Action: ACIF stops processing the print data set.

User Response: Correct the error and resubmit the request.

System Programmer Response: See the I/O error message to determine an appropriate action.

APK102S AN IM IMAGE OBJECT CONTAINS INVALID OR INCORRECT DATA. THE IM IMAGE OBJECT CANNOT BE CONVERTED TO AN IO IMAGE OBJECT.

Explanation: This message is issued when ACIF converts an IM image object to an IO image object and one of the image size values is zero. For a simple IM image object, this message is issued if either the XSize or YSize parameter value of the IID structured field is zero. For a complex IM image object, this message is issued if one of the XCSize, YCSize, SFilSize, or YFilSize parameter values of the ICP structured field is zero.

System Action: ACIF stops processing the input file.

User Response: Correct the error and resubmit the request.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* and the appropriate Print Services Facility: *Diagnosis Guide* and *Reference* for assistance in determining the source of the problem.

APK103S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field name1* STRUCTURED FIELD WAS FOUND WHERE *structured field name2* STRUCTURED FIELD WAS EXPECTED.

Explanation: *Structured field name1* is incorrect at the present point in the input data stream or resource. The structured-field type expected at this point is *structured field name2*. Either the required structured field is

missing or out of sequence, or a line-data record is in the wrong place.

Subsequent error messages give additional information about the processing environment when the error occurred.

System Action: ACIF stops processing the print data set.

User Response: If you created the structured fields for the print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information on the correct format of the referenced structured field. If the structured fields are in the correct order, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK104S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field* STRUCTURED FIELD IS NOT ALLOWED OR FORMS AN INVALID SEQUENCE.

Explanation: The structured field identified in this message is either out of sequence or invalid in an object. The record may be line data. If inline resources are used with data-set header pages, multiple resource groups might be present.

System Action: ACIF stops processing the print data set.

User Response: If you created the structured fields for the print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information on the correct format of the referenced structured field. If the structured fields are in the correct order, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK105I THE ERROR REPORTED ABOVE OCCURRED IN LOGICAL RECORD NUMBER *record number*, WHOSE SEQUENCE NUMBER IS *sequence number*.

Explanation: This message is given in addition to the message that describes the error. It identifies the specific invalid input record. The object (if any) that contains the invalid record is identified in either message APK108I or message APK109I, which follows this message.

The record number specified is relative to the user data stream and is different for multiple transmissions of the data set. However, the record number may be inaccurate if the data set is using a page definition that performs conditional processing.

The sequence number may print as NOT AVAILABLE in the message. For example, a line-data record does not have a sequence number.

System Action: The disposition of the file depends upon the error described in the accompanying messages.

User Response: See the specific error conditions described in the accompanying messages to determine an appropriate response.

System Programmer Response: See the specific error conditions described in the accompanying messages to determine an appropriate response.

APK106I DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: NAME *token name* IN *Begin-type structured field* DOES NOT MATCH NAME *token name* IN *End-type structured field*.

Explanation: The Token Name parameters in the Begin-type and End-type structured fields identified in this message do not correlate. Structured fields may be out of sequence in the input data stream.

When token names are specified, the Token Name parameters in the associated Begin-type and End-type structured fields must correlate.

System Action: Processing continues, and ACIF issues a message identifying the position of the structured field in the input data stream or resource. ACIF issues additional messages identifying the processing environment when the error was found.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the

error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK108I THE ERROR REPORTED ABOVE WAS DETECTED WITHIN OBJECT TYPE *object type* WITH TOKEN NAME *token name*.

Explanation: This message is issued in addition to the message that describes the error. The objects that were being processed are listed to identify the location of the error in the input data stream or in a resource.

System Action: The disposition of the file depends on the error described in the accompanying messages.

User Response: See the specific error conditions described in the accompanying messages to determine an appropriate response.

System Programmer Response: See the specific error conditions described in the accompanying messages to determine an appropriate response.

APK109I THE ERROR REPORTED ABOVE WAS CAUSED BY THE RESOURCE *resource name* IN AN EXTERNAL LIBRARY OR AN INLINE RESOURCE.

Explanation: This message is issued in addition to the message that describes the error. The object identified in the accompanying message was either a resource being processed from an external library or an inline resource. The previous error message, APK108I, identified the member as a page definition, form definition, font, code page, font character set, page segment, or an overlay. The combined information from these two messages can be used to identify the library defined to ACIF on the *typeLIB* parameter, where *type* is the type of resource, such as OVLV for overlay. In the case of an inline form definition or page definition, the resource is not a member of an external library but is included at the beginning of the user's data set.

System Action: The disposition of the file depends on the error described in the accompanying messages.

User Response: See the specific error conditions described in the accompanying messages to determine an appropriate response.

System Programmer Response: See the specific error conditions described in the accompanying messages to determine an appropriate response.

APK110S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE LENGTH SPECIFIED IN THE SELF-DEFINING PARAMETER *identifier* OF THE STRUCTURED FIELD *structured field* IS INCORRECT.

Explanation: Insufficient data was present in the structured field for the length given in the self-defining parameter. The structured field can be contained in a bar code object, a graphics object, or an image object. The bar code object may be contained in an overlay or a composed-text print data set, or it may be imbedded in a data set containing line data. The graphics object may be contained in a composed-text print data set or an overlay, or it may be imbedded in a data set containing line data. The image object may be contained in a composed-text print data set, an overlay, or a page segment, or it may be imbedded in a data set containing line data.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the object, correct the error and resubmit the print request. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the object, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the object with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK112S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: RECORD CONTAINS NO DATA, EVEN THOUGH AT LEAST A CONTROL CHARACTER IS EXPECTED.

Explanation: ACIF read an input record without a control character following the record descriptor word (RDW). A minimum of 1 byte of control-character data is needed to make the record valid.

System Action: ACIF stops processing the print data set.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem. Refer to *MVS/Extended Architecture Data Administration Guide* for more information about construction and placement of RDWs.

APK113S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD LENGTH IS LESS THAN THE INTRODUCER LENGTH.

Explanation: A structured field must have at least 8 bytes of data, the minimum length necessary for a structured-field introducer. The Extension Indicator flag in the structured-field introducer indicates whether the minimum length of the structured field can be greater than 8 bytes.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK114S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: RDW LENGTH DOES NOT AGREE WITH LENGTH IN STRUCTURED FIELD INTRODUCER.

Explanation: All structured fields are preceded by a record descriptor word (RDW) that specifies the length of that record, including the RDW. The record length in the RDW for the current record is less than the sum of the Length parameter in the structured-field introducer and the number of bytes for both the RDW (4 bytes) and the control character (1 byte).

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or resource, ensure that the RDW for the invalid structured field contains a valid record length, and resubmit the print request. If you used a program to create the structured fields for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK116S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: PADDING LENGTH OR EXTENSION LENGTH IS INCORRECT FOR STRUCTURED FIELD.

Explanation: The length of padding or extension specified in the Length or Extension parameter in the structured-field introducer indicates more data than was found in the structured field.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or resource, ensure that the Extension Indicator flag is set correctly and that the Length parameter in the structured-field introducer specifies the actual length of padding for the invalid structured field. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured-field introducer. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK117S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: LENGTH INDICATED IN THE STRUCTURED FIELD INTRODUCER IS INCORRECT FOR *structured field name* STRUCTURED FIELD.

Explanation: The length indicated by the structured-field introducer specifies an invalid number of bytes for the structured field identified in this message. This error is caused by one of the following:

- The Extension or Padding Indicator flags in the structured-field introducer are set incorrectly.
- One or more of the parameters in the invalid structured field contain too many bytes of data.

In some cases, the length of a structured field is specified in a parameter located in another structured field. For example, the length of Fixed Data Text (FDX) structured field is specified in the Size parameter of the Fixed Data Size (FDS) structured field.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or resource, ensure that the Length parameter in the structured-field introducer specifies a valid length for the structured field. Also ensure that the number of bytes in the structured-field parameter matches the length specified in the structured-field introducer. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured-field introducer.

If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK118W UNSUPPORTED STRUCTURED FIELD *structured field code* WAS IGNORED, AND, IF IT BEGAN AN OBJECT, THE OBJECT WAS IGNORED.

Explanation: The Identifier parameter in the structured-field introducer for the invalid structured field specified a structured-field code that was not recognized as a valid structured-field code.

System Action: If the structured field began an object, the object was ignored. Otherwise, only the structured field was ignored, and processing of the rest of the data set continues as usual.

ACIF issues a message identifying the position of the structured field in the input data stream or containing resource. ACIF issues additional messages identifying the processing environment when the error was found.

User Response: If the printed output was unacceptable, and you created the structured fields for the print data set or resource, give the invalid structured field a valid code for its structured-field type. Refer to

Mixed Object Document Content Architecture Reference or *Advanced Function Presentation: Programming Guide and Line Data Reference* for a list of valid structured-field types.

If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured field for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK120S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field name1* **STRUCTURED FIELD CONTAINS AN INCORRECT VALUE FOR THE SIZE OF THE** *structured field name2* **REPEATING GROUP.**

Explanation: *Structured field name1* specifies the length of each repeating group found in *structured field name2*. Either the value specified in *structured field name1* for the size of the repeating group is too small, or the actual length of the repeating-group data is not a multiple of the size specified.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK130S DATA IN AN INPUT RECORD IS INVALID: *structured field name* **STRUCTURED FIELD IS NOT ACCEPTABLE AT THE START OF A DATA STREAM.**

Explanation: The structured-field type identified in this message is not valid at the start of the data stream. Subsequent error messages give additional information about the processing environment when the error occurred.

System Action: ACIF stops processing the print data set.

User Response: If you created the structured fields for the print data set, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK135I DATA IN A FORMDEF RESOURCE IS INVALID: DUPLICATE OVERLAY LOCAL IDENTIFIER WAS FOUND IN THE *structured field* **STRUCTURED FIELD.**

Explanation: The same local identifier was found assigned to more than one Overlay Local Identifier parameter in the Map Medium Overlay (MMO) or Map Page Overlay (MPO) structured field repeating groups. The MMO structured field is contained in the form definition. The MPO is contained in the page definition or the print data set.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to

Advanced Function Printing: Diagnosis Guide for assistance in determining the source of the problem.

APK138S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: OVERLAY LOCAL IDENTIFIER VALUE IS NOT ACCEPTABLE IN THE structured field STRUCTURED FIELD.

Explanation: An invalid Overlay Local Identifier was encountered in the Map Medium Overlay (MMO), Map Page Overlay (MPO), or Medium Modification Control (MMC) structured field repeating groups. The MMO and MMC structured fields are contained in the form definition. The MPO is contained in the page definition or the print data set.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK139S DATA IN A FORMDEF RESOURCE IS INVALID: SUPPRESSION LOCAL IDENTIFIER VALUE IS NOT ACCEPTABLE IN THE MSU STRUCTURED FIELD.

Explanation: The Suppression Local Identifier parameter in the Map Suppression (MSU) structured field is invalid. The MSU structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK140S DATA IN A FORMDEF RESOURCE IS INVALID: TWO MMC STRUCTURED FIELDS ARE DEFINED WITH THE SAME IDENTIFIER, identifier.

Explanation: Two Medium Modification Control (MMC) structured fields in a single form environment group have the same value in their Medium Modification Control Identifier parameters. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK141S DATA IN A FORMDEF RESOURCE IS INVALID: MEDIUM SUPPRESSION TOKEN NAME IS REPEATED IN MSU STRUCTURED FIELD.

Explanation: The Token Name parameters in two repeating groups in a Map Suppression (MSU) structured field have the same value. The MSU structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the

form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK143S DATA IN A FORMDEF RESOURCE IS INVALID: COPY SPECIFICATIONS IN THE MCC STRUCTURED FIELD ARE NOT ACCEPTABLE.

Explanation: Either a gap or an overlap exists in the Starting and Stopping Copy Numbers, or the maximum number of copies for one set of modifications has been exceeded. The Copy Number parameters are specified in the Medium Copy Count (MCC) structured field. The MCC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, ensure that the Starting Copy Number and Stopping Copy Number parameters in a repeating group in an MCC structured field have valid values that correlate. Also, verify that fewer than 255 copies have been requested. If 255 or more copies with the same modifications are needed, define two or more MCC structured fields. Refer to *Mixed Object Document Content Architecture Reference* for more information on the MCC structured field. If the MCC has no errors, the error may be an ACIF logic error.

If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK145S DATA IN A FORMDEF RESOURCE IS INVALID: THE FORMS-FLASH VALUE IN MMC STRUCTURED FIELD, ID *identifier*, IS NOT ACCEPTABLE.

Explanation: The Medium Modification Control (MMC) structured field contains an invalid value for the repeating group that contains forms-flash modification. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has

no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK146S DATA IN A FORMDEF RESOURCE IS INVALID: MORE THAN 8 OVERLAYS ARE SPECIFIED IN MMC STRUCTURED FIELD, ID *identifier*.

Explanation: In a Medium Modification Control (MMC) structured field, the maximum number of overlays allowed in one set of modifications has been exceeded. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK147S DATA IN A FORMDEF RESOURCE IS INVALID: MORE THAN 8 SUPPRESSIONS ARE SPECIFIED IN MMC STRUCTURED FIELD, ID *identifier*.

Explanation: In a Medium Modification Control (MMC) structured field, the maximum number of suppressions allowed in one set of modifications has been exceeded. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you

used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK152S DATA IN A FORMDEF RESOURCE IS INVALID: MMC STRUCTURED FIELD WAS NOT FOUND TO COMPARE WITH IDENTIFIER *identifier value* IN MCC STRUCTURED FIELD.

Explanation: The Medium Modification Control Identifier parameter in the Medium Copy Count (MCC) structured field contains a value that did not match the Medium Modification Control Identifier parameter in any Medium Modification Control (MMC) structured field in the form environment group. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MCC or MMC structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC and MMC have no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK154S DATA IN A FORMDEF RESOURCE IS INVALID: OVERLAY LOCAL IDENTIFIER IN MMC STRUCTURED FIELD, ID *identifier*, WAS NOT FOUND IN MMO STRUCTURED FIELD.

Explanation: The overlay modification in the Medium Modification Control (MMC) structured field was not present in the Map Medium Overlay (MMO) structured field. The MMC and MMO structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content*

Architecture Reference for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK155S DATA IN A FORMDEF RESOURCE IS INVALID: TOO MANY COPY CONTROLS WERE SPECIFIED FOR THE CURRENT FORM ENVIRONMENT GROUP.

Explanation: For a given physical page, up to 256 bytes of data can be specified for the printer command that describes the copies and modifications to be made. The current form environment group causes the data for the command to exceed 256 bytes. ACIF builds the printer command from data contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, either reduce the number of copy groups in the Medium Copy Count (MCC) structured field or reduce the number of modifications specified in the Medium Modification Control (MMC) structured field. Otherwise, split these functions between two or more form environment groups in two or more medium maps. Then, include in your input two or more identical copies of the same page that each select an appropriate copy group by use of the Invoke Medium Map (IMM) structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the MMC and MMO structured fields.

If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK156S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: NULL NAME IS NOT ACCEPTABLE IN *structured field name* STRUCTURED FIELD.

Explanation: All Begin-type and End-type structured fields can include an 8-byte token name. A null token name is not allowed for the listed structured field.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK157S MISMATCH BETWEEN PRINT DATA SET AND FORMDEF RESOURCE: MEDIUM MAP *medium map* SPECIFIED IN IMM STRUCTURED FIELD WAS NOT FOUND IN FORMDEF *form definition name*.

Explanation: The Token Name parameter in the Invoke Medium Map (IMM) structured field specifies the token name used to locate a medium map in the form definition. This parameter must match the Token Name parameter specified in bytes 0–7 in one of the Begin Medium Map (BMM) structured fields in the current form definition. The IMM structured field is contained in the print data set.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: Ensure that the correct form definition was specified. If it was, and if you added the Invoke Medium Map structured field to the print data set, change the Token Name in the IMM structured field and run ACIF. Refer to *Mixed Object Document Content Architecture Reference* for more information about the BMM and IMM structured fields. If the correct form definition was specified, and if you used a program to imbed the IMM structured field in the print data set, verify that the copy group name that you gave the program is valid for the form definition you have specified.

System Programmer Response: No response is necessary.

APK158S PAGEDEF PARAMETER MUST BE SPECIFIED IN ORDER TO PRINT THIS DATA SET. DETERMINE THE PERMISSIBLE VALUES USED IN YOUR INSTALLATION FOR THE PAGEDEF PARAMETER.

Explanation: The current data set contains at least one record of line data. A data set containing line data cannot be printed without an active page definition. No PAGEDEF parameter was provided for this job.

This error can also occur if a composed-text page in the print data set contains a structured field without the required X'5A' control character preceding the structured-field introducer. The missing control character makes the record appear to be line data. A page definition is necessary to process line data.

This error can also occur if a Presentation-Text Data (PTX) structured field or a Begin Image (BIM) structured field is encountered outside of a page.

System Action: ACIF stops processing the print data set.

User Response: If you intended to print line data, specify a page definition with the ACIF PAGEDEF parameter.

If you did not intend to print line data, and you used a program to create the structured fields for the print data set, ensure that all composed-text data records begin with the X'5A' control character.

If you did not intend to print line data, and you used a program to create the structured fields for the print data set, contact your system programmer.

System Programmer Response: No response is necessary.

APK159S THE END OF THE DATA STREAM WAS ENCOUNTERED BEFORE THE LOGICAL END OF AN OBJECT WITHIN THE DATA STREAM.

Explanation: ACIF was processing an object that began with a Begin-type structured field. However, the input data stream ended before a corresponding End-type structured field was found. The message can also occur if the system operator prematurely interrupts or ends a print request by issuing an INTERRUPT, RESTART, or CANCEL Job Entry Subsystem (JES) command.

System Action: ACIF stops processing the print data set.

User Response: If you created the structured fields for the print data set, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured

field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK162S MISMATCH BETWEEN PRINT DATA SET AND PAGEDDEF RESOURCE: DATA MAP data map SPECIFIED IN IDM STRUCTURED FIELD WAS NOT FOUND IN PAGEDDEF page definition.

Explanation: The Token Name parameter in the Invoke Data Map (IDM) structured field specifies the token name used to locate a data map in the page definition. The name must match the value specified in the Token Name parameter in the Begin Data Map (BDM) structured field in the current page definition. The IDM structured field is contained in the print data set.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: Ensure that the correct page definition was specified. If it was, and if you added the Invoke Data Map structured field to the print data set, change the Token Name in the IDM structured field and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the BDM and IDM structured fields. If the correct page definition was specified, and if you used a program to imbed the IDM structured field in the print data set, verify that the data map name that you supplied the program is one that is valid for the page definition you have specified.

System Programmer Response: No response is necessary.

APK163S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE SCALE FACTOR VALUE IN THE IOC STRUCTURED FIELD IS NOT ACCEPTABLE.

Explanation: The Image Block Scale Factor parameter in the Image Output Control (IOC) structured field is invalid. The image block or image cell may be contained in an overlay, a page segment, or a composed-text print data set. It may also be imbedded in a data set containing line data, using a Begin Image (BIM) structured field.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the resource or print data set containing the image, correct the error in the referenced structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the resource or print data set containing the image, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK166S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: AN ENTRY IN A MCF STRUCTURED FIELD CONTAINS AMBIGUOUS IDENTIFICATION.

Explanation: Two ways to identify a font in the Map Coded Font (MCF) structured field are either with a Coded Font Name parameter or with a combination of the Font Character Set Name parameter and the Code Page Name parameter. One of the repeating groups in an MCF structured field specified both a Coded Font Name parameter and at least a Font Character Set Name or a Code Page Name parameter. The MCF structured field is contained in a composed-text print data set, an overlay, or a page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK167S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: AN ENTRY IN AN MCF STRUCTURED FIELD CONTAINS INCOMPLETE IDENTIFICATION.

Explanation: One of the repeating groups in a Map Coded Font (MCF) structured field does not contain enough information to identify a coded font. Two ways to identify a font in the Map Coded Font (MCF) structured field are either with a Coded Font Name parameter or with a combination of the Font Character Set Name parameter and the Code Page Name parameter. An entry contains only a Font Character Set Name parameter or a Code Page Name parameter. The MCF structured field is contained in a composed-text print data set, an overlay, or a page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK169S INSUFFICIENT VIRTUAL STORAGE PREVENTED FURTHER PROCESSING. INCREASE REGION SIZE, AND RESUBMIT THE PRINT REQUEST.

Explanation: Insufficient storage is available in the ACIF address space to contain the internal control block needed to read an object.

System Action: ACIF stops processing the print data set.

User Response: Inform your system programmer that this error occurred.

System Programmer Response: The value of the REGION parameter used for the ACIF job should be increased.

APK170S DATA IN A FORMDEF RESOURCE IS INVALID: THE SIMPLEX/DUPLEX VALUE IN MMC STRUCTURED FIELD, ID identifier, IS NOT ACCEPTABLE.

Explanation: In the Medium Modification Control (MMC) structured field with the specified identifier, either the simplex or the duplex keyword-parameter value is invalid. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK171S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: FONT LOCAL IDENTIFIER VALUE IS NOT ACCEPTABLE IN THE structured field STRUCTURED FIELD.

Explanation: The Map Coded Font (MCF) structured field consists of repeating groups. In one of the groups, the value of the Coded Font Local Identifier parameter for the font (section) being mapped is invalid. The MCF structured field is contained in a composed-text print data set, an overlay, or a page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was

valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK172S DATA IN A FORMDEF RESOURCE IS INVALID: THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD INCLUDES BOTH NORMAL AND TUMBLE DUPLEX.

Explanation: The Medium Copy Count (MCC) structured field refers to one or more Medium Modification Control (MMC) structured fields, which include requests for both normal duplex and tumble duplex. You cannot request both normal duplex and tumble duplex within the same medium map. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MCC or MMC structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC and MMC have no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK178S DATA IN A FORMDEF RESOURCE IS INVALID: THE MCC STRUCTURED FIELD HAS AN ODD NUMBER OF COPY GROUPS, BUT SPECIFIES DUPLEX.

Explanation: The Medium Copy Count (MCC) structured field specifies an odd number of copy groups, but the copy group modifications specified in the Medium Modification Control (MMC) structured field include duplex, which requires an even number of copy groups. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MCC or MMC structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC and MMC have no errors, the error may be an ACIF logic error. If you

used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK179S DATA IN A FORMDEF RESOURCE IS INVALID: THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD INCLUDES BOTH SIMPLEX AND DUPLEX.

Explanation: The Medium Copy Count (MCC) structured field refers to two or more Medium Modification Control (MMC) structured fields, which include requests for both simplex and duplex printing. You cannot specify both simplex and duplex printing within the same medium map. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MCC or MMC structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC and MMC have no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK181S DATA IN A FORMDEF RESOURCE IS INVALID: UNEQUAL COPY COUNTS FOR DUPLEX SHEETS ARE SPECIFIED IN THE MCC STRUCTURED FIELD.

Explanation: The set of modifications referred to by the Medium Copy Count (MCC) structured field includes duplexing, but the numbers of copies in two corresponding repeating groups are not equal. The repeating groups are defined in the Medium Map Control structured field (MMC). The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MCC or MMC

structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC and MMC have no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK188S DATA IN A FORMDEF RESOURCE IS INVALID: THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD SELECTS MORE THAN ONE INPUT SOURCE.

Explanation: The Medium Copy Count (MCC) structured field refers to one or more Medium Modification Control (MMC) structured fields, which include requests for the primary input source and the alternate source. You cannot specify both the primary input source and the alternate source for one copy group. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MCC or MMC structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC and MMC have no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK190S DATA IN A FORMDEF RESOURCE IS INVALID: THE BIN-SELECTION VALUE IN MMC STRUCTURED FIELD, ID *identifier*, IS NOT ACCEPTABLE.

Explanation: In the Medium Modification Control (MMC) structured field with the identifier specified in the message text, the bin-selection parameter value was invalid. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK191S DATA IN A FORMDEF RESOURCE IS INVALID: THE SUPPRESSION LOCAL IDENTIFIER VALUE IN MMC STRUCTURED FIELD, ID *identifier*, IS NOT ACCEPTABLE.

Explanation: The Medium Modification Control Identifier parameter in a Medium Modification Control (MMC) structured field is invalid. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK210S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: A REQUIRED SELF-DEFINING PARAMETER WITH ID *id* WAS MISSING FROM A *structured field name* STRUCTURED FIELD.

Explanation: The self-defining parameter specified in the message was not found in the indicated structured field. This is a required self-defining parameter. The structured field is contained in an image object. The image object may be contained in a composed-text print data set, an overlay, or a page segment; or it may be imbedded in a data set containing line data.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the image object, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the image object, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the image object with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK212S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE UNIT BASE PARAMETER IN THE *structured field name* STRUCTURED FIELD IS INVALID.

Explanation: An invalid Unit Base value was encountered in the structured field identified in this message. The structured field is contained in the Object Environment Group of an image object. The image object may be contained in a composed-text print data set, an overlay, or a page segment; or it may be imbedded in a data set containing line data.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the image object, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the image object, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the image object with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK213S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE L-UNITS PER UNIT BASE PARAMETER IN THE *structured field name* STRUCTURED FIELD IS INVALID.

Explanation: An invalid L-Units value was encountered in the structured field identified in this message. The structured field is contained in the Object Environment Group of an image object. The image object may be contained in a composed-text print data set, an overlay, or a page segment; or it may be imbedded in a data set containing line data.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the image object, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the image object, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the image object with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK217S DATA IN AN INPUT RECORD IS INVALID: PARAMETER IN A BR STRUCTURED FIELD CONTAINS UNACCEPTABLE DATA.

Explanation: One of the parameters in the Begin Resource (BR) structured field is invalid. The BR structured field is contained in the print data set.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you placed the BR structured field in the print data set, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If you used a program to place the BR structured field in the print data set, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to place the BR structured field in the print data set, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* and the appropriate Print Services Facility: *Diagnosis Guide and Reference* for assistance in determining the source of the problem.

APK221S DATA IN A FORMDEF RESOURCE IS INVALID: THE ORIENTATION VALUE *value* IN THE MDD STRUCTURED FIELD IS UNACCEPTABLE.

Explanation: The Medium Descriptor (MDD) structured field has an invalid orientation value. The MDD structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK250S DATA IN A PAGE OR RESOURCE IS MISSING: THE REQUIRED STRUCTURED FIELD *structured field name* COULD NOT BE FOUND TO COMPLETE THE PROCESSING OF A PAGE OR RESOURCE.

Explanation: The structured field identified in this message is required to complete the processing of a page or resource. This structured field was not found before the end of the page or resource was encountered.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was

valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK251S DATA IN A FORMDEF RESOURCE IS MISSING: THE FORMDEF DOES NOT CONTAIN ANY MEDIUM MAPS.

Explanation: The form definition did not specify any medium maps; however, a medium map is required to print a page.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK253S DATA IN A FORMDEF RESOURCE IS INVALID: THE PRINT QUALITY VALUE IN MMC STRUCTURED FIELD, ID *identifier*, IS NOT ACCEPTABLE.

Explanation: The Medium Modification Control (MMC) structured field specified a print quality value of 0, which is outside the valid range. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK254S DATA IN A FORMDEF RESOURCE IS INVALID: THE OFFSET STACKING VALUE IN MMC STRUCTURED FIELD, ID *identifier*, IS NOT ACCEPTABLE.

Explanation: The Medium Modification Control (MMC) structured field specified an offset stacking value other than 0 or 1. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the MMC structured field and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MMC has no errors, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK258S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field* STRUCTURED FIELD IS NOT ALLOWED BETWEEN OBJECTS.

Explanation: The structured field identified in this message is not allowed at the point in the input data stream or resource at which it was found.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured fields are in the correct order, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK259S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE X-DIRECTION AND Y-DIRECTION L-UNITS PER UNIT BASE VALUES SPECIFIED IN STRUCTURED FIELD *structured field* DO NOT MATCH.

Explanation: The X-direction and Y-direction L-Units per Unit Base values in the structured field identified in the message are not identical.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK261S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD *structured field* CONTAINED A CODED-FONT-LOCAL-IDENTIFIER VALUE THAT WAS USED IN A PREVIOUS FONT MAPPING STRUCTURED FIELD.

Explanation: One or more font mapping structured fields in the same active environment group or object environment group used the same coded font local identifier for different coded fonts. The Map Coded Font (MCF) structured field that attempted to use the already-mapped coded font local identifier is identified in the message. The MCF structured field can be contained in a composed-text print data set, an overlay, or a page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you were printing a composed-text print data set or an overlay, and you created the structured fields in the object containing the error, check the Coded Font Local Identifiers in the MCF structured field for duplicates. If the MCF structured field has no error, the error may be an ACIF logic error. If you used

a program to create the structured fields in the object containing the error, contact your system programmer.

If you were printing a data set containing line data using a page definition, and if you created the structured fields for the page definition, check the Coded Font Local Identifiers in the MCF structured field for duplicates. If the MCF structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK262S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD *structured field* CONTAINS AN INVALID ROTATION VALUE.

Explanation: The rotation value specified in the named structured field was invalid.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK263S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: OVERLAY *overlay name* NAMED IN AN IPO STRUCTURED FIELD IS NOT NAMED IN AN MPO STRUCTURED FIELD.

Explanation: An Include Page Overlay (IPO) structured field names a page overlay, but the overlay was not previously defined in the Map Page Overlay (MPO) structured field in the Active Environment Group (AEG) of the page, which contains the IPO. The MPO may be contained in the AEG of a composed-text page or a page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK264S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: A CODED FONT NAMED IN AN OBJECT ENVIRONMENT GROUP IS NOT NAMED IN THE ACTIVE ENVIRONMENT GROUP OF THE PAGE OR RESOURCE.

Explanation: A Map Coded Font (MCF) structured field in an object environment group names a coded font, but that coded font is not defined in the MCF structured field in the active environment group of the page or resource containing the object environment group.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource. ACIF stops processing and printing the data set.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the

error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK267S EITHER NO ENVIRONMENT GROUP WAS SPECIFIED FOR THE PAGE OR AN ERROR OCCURRED IN THE ENVIRONMENT GROUP.

Explanation: Either no environment group was specified, or an error occurred in one of the structured fields in the environment group. If an environment group was present but contained an error, a previous ACIF message identifies the error. The environment group causing this error may be contained in an overlay, a page definition, or a composed-text print data set.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or the resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* or *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set or the resource, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or the resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK268S AN ENTRY IN AN MCF STRUCTURED FIELD DOES NOT CONTAIN CODE PAGE INFORMATION.

Explanation: One of the repeating groups in a Map Coded Font Format 2 (MCF-2) structured field specifies a font character set but no code page information. This error was detected while processing a graphics object within a page or overlay.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the object, correct the error and resubmit the print request. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the object, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the object with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK269S A VALUE OF ZERO WAS SPECIFIED AS THE L-UNITS PER UNIT BASE IN THE structured field name STRUCTURED FIELD.

Explanation: Several structured fields specify an L-Units per Unit Base value: Medium Descriptor (MDD), Page Descriptor (PGD), Presentation Text Descriptor (PTD-2), Object Area Descriptor (OBD), Graphics Data Descriptor (GDD), Image Data Descriptor (IDD), Barcode Data Descriptor (BDD), and Image Input Descriptor (IID). The value of zero is invalid for the L-Units per Unit Base.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for information about the structured fields. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the print data set, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set or resource with the error, verify that the input to that program was valid. If the input was valid, refer to *Mixed Object Document Content Architecture Reference*, *Advanced Function Presentation: Programming Guide and Line Data Reference*, and the appropriate Print Services Facility System Programming Guide for assistance in determining the source of the problem.

APK270S DATA IN A PAGEDEF RESOURCE IS MISSING: THE PAGEDEF DOES NOT CONTAIN ANY DATA MAPS.

Explanation: The page definition did not specify any data maps and a data map is required to print a data set containing line data.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK271S DATA IN A FORMDEF RESOURCE IS INVALID: THE DUPLEX SPECIFICATION IN THE PGP STRUCTURED FIELD IS NOT ACCEPTABLE.

Explanation: The duplex specification value in the Page Position (PGP) structured field is not acceptable. The PGP structured field is contained in the form definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK272S DATA IN A FORMDEF RESOURCE IS INVALID: THE PGP STRUCTURED FIELD DOES NOT CONTAIN A PAGE ORIGIN POSITION FOR THE FRONT SIDE OF A SHEET.

Explanation: The Page Position format-2 (PGP) structured field must contain a repeating group that defines the Page Origin Position for the front side. This value will also be used for the back side of a duplex sheet unless the PGP structured field contains a repeating group that specifies the Page Origin Position for the back side of the sheet. The PGP structured field is contained in the form definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK273S DATA IN A FORMDEF RESOURCE IS INVALID: THE CONSTANT FORMS CONTROL VALUE IN THE MMC STRUCTURED FIELD IDENTIFIER, IS NOT ACCEPTABLE.

Explanation: The Constant Forms Control modification in the Medium Modification Control (MMC) structured field contained an unsupported value. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK274S DATA IN A FORMDEF RESOURCE IS INVALID: THE MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD INCLUDE CONFLICTING CONSTANT FORMS CONTROL VALUES FOR THE SAME SIDE OF THE SHEET.

Explanation: All Medium Modification Control (MMC) structured fields referenced by the Medium Copy Count (MCC) structured field must use the same Constant Forms Control value for the same side of a sheet. The

MMC and MCC structured fields are contained in the form definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK275S DATA IN A FORMDEF RESOURCE IS INVALID: A MEDIUM MAP SPECIFIES ONLY CONSTANT DATA FOR A PAGE.

Explanation: An attempt was made to process a page using a medium map specifying Constant Forms Control for both the front and back sides of a duplexed page or for the front side of a simplex page. Another medium map must be invoked to allow processing of the remaining line or page data. The Constant Forms Control is contained in a Medium Modification Control (MMC) structured field. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the form definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the form definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK278S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE OUTPUT OPTION SPECIFIED IN THE *name* STRUCTURED FIELD IS INVALID.

Explanation: The structured field in error contained an invalid Output Option value. The structured field can be contained in a bar code object, a graphics object, or an image object. The bar code object may be contained in an overlay or a composed-text print data set or it may be imbedded in a data set containing line data. The graphics object may be contained in a composed-text print data set or an overlay; or it may be imbedded in a data set containing line data. The image object may be contained in a composed-text print data set, an overlay, or a page segment; or it may be imbedded in a data set containing line data.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the object, correct the error and resubmit the print request. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the object, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the object with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK300I DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SKIPPING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* IS 0.

Explanation: The current record contains a control character that indicates a skip to a Line Descriptor (LND) structured field with a specific channel control. However, the LND structured field identified in this message had a value of 0 in its Next Line Descriptor IF SKIPPING parameter. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK301S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SKIPPING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* IS *parameter value*. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF *parameter value*.

Explanation: In the Line Descriptor (LND) structured field identified in this message, the value of the next LND IF SKIPPING parameter is greater than the total number of LND structured fields in the page definition.

System Action: ACIF stops processing the print data set and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK307S DATA IN A PAGEDEF RESOURCE IS INVALID: IN LND STRUCTURED FIELD NUMBER *structured field number*, THE REUSE RECORD FLAG WAS SET BUT THE NEXT LINE DESCRIPTOR IF REUSING DATA PARAMETER WAS 0.

Explanation: In the Line Descriptor (LND) structured field identified in this message, the Reuse Record flag had a value of B'1', indicating that the line data being processed in this LND structured field should be reused and processed. The Next Line Descriptor If Reusing Data parameter should point to the LND structured field used to continue processing. However, the value for

the Reusing Data parameter was X'0000', indicating the end of the chain. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK309S DATA IN A PAGEDEF RESOURCE IS INVALID: THE REPEATING GROUP LENGTH PARAMETER VALUE IN CCP STRUCTURED FIELD *CCP identifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. Either the Length of Repeating Groups parameter is zero, or the length of the repeating group data is not a multiple of the size specified in that parameter. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK310S DATA IN A PAGEDEF RESOURCE IS INVALID: THE COUNT PARAMETER VALUE IN THE LNC STRUCTURED FIELD WAS 0.

Explanation: The Count parameter in the Line Descriptor Count (LNC) structured field had a value of zero. The LNC structured field is contained in the page definition.

System Action: ACIF stops processing the print data set and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK312S DATA IN A PAGEDEF RESOURCE IS INVALID: THE SIZE PARAMETER VALUE IN THE FDS STRUCTURED FIELD WAS 0.

Explanation: The Size parameter in the Fixed Data Size (FDS) structured field has a value of 0. The FDS structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK314S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NUMBER OF REPEATING GROUPS PARAMETER VALUE IN CCP STRUCTURED FIELD CCP identifier IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. Either the Number of Repeating Groups parameter contained in the CCP structured field is zero, or the number of repeating groups does not match the number specified in the parameter. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK315S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SPACING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER structured field number IS 0.

Explanation: The logical-record control character indicates that the Next Line Descriptor If Spacing parameter should be followed. However, in the Line Descriptor (LND) structured field identified in this message, the Next Line Descriptor If Spacing parameter value was zero. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK316S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SPACING PARAMETER IN LND STRUCTURED FIELD NUMBER *structured field number* IS *parameter value*. THIS VALUE IS TOO LARGE.

Explanation: The logical record control character indicates that the Next Line Descriptor If Spacing parameter in the Line Descriptor (LND) structured field should be followed. However, in the Line Descriptor (LND) structured field identified in this message, the Next Line Descriptor If Spacing parameter value was greater than the total number of line descriptors in the data map. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK317S DATA IN A PAGEDEF RESOURCE IS INVALID: THE LENGTH OF COMPARISON STRING PARAMETER VALUE IN CCP STRUCTURED FIELD *CCP identifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. Either the Length of Comparison String parameter is zero, or the length of the comparison string data does not match the length of a repeating group minus the fixed lengths of the remaining fields of the repeating group. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK319S DATA IN A PAGEDEF RESOURCE IS INVALID: SUPPRESSION TOKEN NAME = *token name* IS INVALID IN LND STRUCTURED FIELD NUMBER = *structured field number*.

Explanation: The Suppression Token Name parameter in the Line Descriptor (LND) structured field in the page definition has a null value. A null value is any value that contains X'FFFF' in the first two bytes. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK320S DATA IN A PAGEDEF RESOURCE IS INVALID: THE IDENTIFIER *identifier1* SPECIFIED IN THE NEXT CCP IDENTIFIER PARAMETER IN CCP STRUCTURED FIELD *identifier2* WAS NOT FOUND.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Next Conditional Processing Control Identifier parameter in the CCP structured field specifies the identifier used to

locate a CCP, if the CCP structured fields are chained. The identifier must match a value specified in the CCP Identifier parameter of another CCP within the same page definition. The identifier specified in the Next CCP Identifier parameter did not match the CCP Identifier of any CCPs in the page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK321S DATA IN A PAGEDEF RESOURCE IS INVALID: THE TIMING OF ACTION PARAMETER VALUE *value* IN CCP STRUCTURED FIELD *CCP identifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Timing of Action parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set. ACIF issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK322S DATA IN A PAGEDEF RESOURCE IS INVALID: THE MEDIUM MAP ACTION PARAMETER VALUE *value* IN CCP STRUCTURED FIELD *CCP identifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Medium Map Action parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK323S DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA MAP ACTION PARAMETER VALUE *value* IN CCP STRUCTURED FIELD *CCP identifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Data Map Action parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that

program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK324S DATA IN A PAGEDEF RESOURCE IS INVALID: THE COMPARISON PARAMETER VALUE *value* IN CCP STRUCTURED FIELD *CCP identifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Comparison parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK326S DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA MAP *data map name* SPECIFIED IN THE DATA MAP NAME PARAMETER OF CCP STRUCTURED FIELD *CCP identifier* WAS NOT FOUND.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Data Map Name parameter in one of the repeating groups of the CCP structured field specifies the token name of a data map used to locate a data map in the page definition. The name must match the value specified in the Token Name parameter in one of the Begin Data Map (BDM) structured fields in the current page definition. No data map with name *data map name* was found in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function*

Presentation: Programming Guide and Line Data Reference for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK327S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF REUSING DATA PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* WILL CAUSE AN INFINITE LOOP.

Explanation: The Next Line Descriptor If Reusing Data parameter in the Line Descriptor (LND) structured field identified in this message caused an infinite-loop condition. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK329S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF REUSING DATA PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* IS *parameter value1*. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF *parameter value2*.

Explanation: The Next Line Descriptor If Reusing Data parameter in the Line Descriptor (LND) structured field identified in this message has an invalid value. The value is greater than the Count parameter in the Line Descriptor Count (LNC) structured field in the

current data map. The LNC and LND structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK330S DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA START POSITION PARAMETER VALUE WHEN ADDED TO THE DATA LENGTH PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* EXCEEDS THE FDS STRUCTURED FIELD SIZE PARAMETER VALUE OF *parameter value*.

Explanation: The Use Fixed Data flag in byte 0 in the Line Descriptor (LND) structured field was set to B'1'. This indicates that data from Fixed Data Text (FDX) structured fields is to be added to the data placed within the page by the LND structured field. The FDX and LND structured fields are contained in the page definition.

The Data Start Position parameter in the LND structured field indicates the offset of the first byte of data. The Data Length parameter indicates the number of bytes of FDX data to be placed within the page. This error was caused when these two parameters specified more data than was contained in the FDX structured fields. The number of bytes of data in the FDX structured fields can be found in the Size parameter of the Fixed Data Size (FDS) structured field.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create

the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK334S DATA IN A PAGEDEF RESOURCE IS INVALID: THE AMOUNT OF FIXED DATA RECEIVED DID NOT AGREE WITH THE VALUE SPECIFIED IN THE FDS STRUCTURED FIELD SIZE PARAMETER.

Explanation: The Fixed Data Text (FDX) structured field contained more bytes of data than what was indicated in the Size parameter of the Fixed Data Size (FDS) structured field. The FDS and FDX structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK335S DATA IN A PAGEDEF RESOURCE IS INVALID: THE MEDIUM MAP *medium map name* SPECIFIED IN THE MEDIUM MAP NAME PARAMETER OF CCP STRUCTURED FIELD *CCP identifier* WAS NOT FOUND.

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Medium Map Name parameter in one of the repeating groups of the CCP structured field specifies the token name of a medium map used to locate a medium map in the form definition. The name must match the value specified in the Token Name parameter in one of the Begin Medium Map (BMM) structured fields in the current form definition. No medium map with name *medium map name* was found in the form definition. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK337S DATA IN A PAGEDEF RESOURCE IS INVALID: IN LND STRUCTURED FIELD NUMBER *structured field number*, THE CONDITIONAL PROCESSING FLAG WAS SET BUT THE CONDITIONAL PROCESSING CONTROL IDENTIFIER PARAMETER WAS ZERO.

Explanation: In the Line Descriptor (LND) structured field, the Conditional Processing flag had a value of B'1', indicating that the line data to be processed by this LND structured field is to be compared with a value specified in a Conditional Processing Control (CCP) structured field. The CCP Identifier parameter in the LND structured field is used to locate one of the CCP structured fields in the current page definition. This Identifier parameter was set to 0, which is not a valid value if the Conditional Processing flag is on. The LND and CCP structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK339S DATA IN A PAGEDEF RESOURCE IS INVALID: THE IDENTIFIER *identifier* SPECIFIED IN THE CONDITIONAL PROCESSING CONTROL IDENTIFIER PARAMETER IN LND STRUCTURED FIELD NUMBER *structured field number* WAS NOT FOUND.

Explanation: In the Line Descriptor (LND) structured field, the Conditional Processing flag had a value of B'1', indicating that the line data to be processed by this LND structured field is to be compared with a value specified in a Conditional Processing Control (CCP) structured field. The CCP Identifier parameter in the LND structured field is used to locate one of the CCP structured fields in the current page definition. The identifier specified in *identifier* does not match the value specified in the CCP Identifier parameter in any of the CCP structured fields in the current page definition. The LND and CCP structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK340S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* IS *value1*. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF *value2*.

Explanation: The Next Line Descriptor If Conditional Processing parameter in the Line Descriptor (LND) structured field has an invalid value. The value is greater than the Count parameter in the Line Descriptor Count (LNC) structured field in the current data map. The LNC and LND structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK342S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* WILL CAUSE AN INFINITE LOOP.

Explanation: The Next Line Descriptor If Conditional Processing parameter in the Line Descriptor (LND) structured field caused an infinite-loop condition. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK344S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NUMBER OF LND STRUCTURED FIELDS DOES NOT MATCH THE VALUE SPECIFIED IN THE LNC STRUCTURED FIELD.

Explanation: The number of Line Descriptor (LND) structured fields found in a page definition is either greater than or less than the value specified in the Line Descriptor Count (LNC) structured field. The LND and LNC structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the print data set with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK346W DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS INVALID: A SKIP TO A NONEXISTENT CHANNEL = *channel* ON RECORD NUMBER = *record number* WAS DETECTED WITHIN THE LND STRUCTURED FIELDS. OUTPUT WAS FORCED TO SINGLE SPACING, WHICH MAY CAUSE BLANK PAGES.

Explanation: An attempt was made to skip to a channel not defined in the current data map. The Line Descriptor (LND) structured fields in the page definition are invalid. During scanning, the entire Next Line Descriptor If Skipping parameter could not be followed because an LND had the End Page If Skipping flag set. This created an infinite loop on the same input record. The LND structured field is contained in the page definition.

System Action: The record containing the error was forced to single spacing. When forced single spacing occurs, the carriage control character on the record is ignored. The record is treated as if a X'09' machine control character or a X'40' ANSI control character was specified in the record that caused the error.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function*

Presentation: Programming Guide and Line Data Reference for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK348S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SPACING PARAMETER VALUE *value* IN LND STRUCTURED FIELD NUMBER *structured field number* IS INVALID.

Explanation: The logical-record control character had indicated that the Next Line Descriptor If Spacing parameter should be followed. However, in the Line Descriptor (LND) structured field identified by *structured field number*, the Next Line Descriptor If Spacing parameter value was either zero or greater than the total number of Line Descriptors in the data map. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK353S DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA LENGTH PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* DOES NOT MATCH THE LENGTH OF COMPARISON STRING PARAMETER VALUE IN CCP STRUCTURED FIELD *CCP identifier*.

Explanation: In the Line Descriptor (LND) structured field, the value of the Data Length parameter is used in identifying the field of the current input record for which conditional processing is to be performed. This field is to be compared with the Comparison String specified in the Conditional Processing Control (CCP) structured field. The length specified in the Data Length parameter in the LND structured field does not match the length specified in the Length of Comparison String parameter of the CCP structured field. The LND and CCP structured fields are contained in the page definition.

System Action: ACIF stops processing the print data set, and issues a message identifying the position of the structured field in the data stream or resource.

User Response: If you created the structured fields for the page definition, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the error may be an ACIF logic error. If you used a program to create the structured fields for the page definition, contact your system programmer.

System Programmer Response: If an IBM licensed program was used to create the structured fields for the page definition with the error, verify that the input to that program was valid. If the input was valid, refer to *Advanced Function Printing: Diagnosis Guide* for assistance in determining the source of the problem.

APK400S THE *parameter* NUMBER VALUE IS NOT NUMERIC.

Explanation: A numeric value must be specified after the parameter.

System Action: ACIF terminates.

User Response: Use a numeric value after the parameter and resubmit the job.

System Programmer Response: No response is necessary.

APK401S THE *parameter* NAME MUST BE DELIMITED WITH QUOTES.

Explanation: The attribute name of the parameter must begin and end with single quotes.

System Action: ACIF terminates.

User Response: Use single quotes before and after the attribute name in the parameter.

System Programmer Response: No response is necessary.

APK402S THE PARAMETER 'xxxxxxx' IS INVALID.

Explanation: A parameter that is not valid for ACIF was specified.

System Action: ACIF terminates.

User Response: Correct the parameter and resubmit the job.

System Programmer Response: No response is necessary.

APK403S THE REQUESTED RESOURCE *number* IS UNKNOWN.

Explanation: A resource I/O has been requested, but the resource type is unknown to ACIF. This condition is caused by an ACIF logic error. The resource type codes are listed below:

Type	Resource Name
1	- Print input file
2	- FORMDEF file
3	- PAGEDEF file
4	- OVERLAY file
5	- SEGMENT file
6	- Coded FONT file
7	- Coded PAGE file
8	- FONT Character Set file
9	- FONT Metric file
10	- FONT Shape file
20	- Print output file
21	- Messages output file
22	- SPOOL file
23	- Dummy input file
24	- Dummy output file
25	- Parameter file
26	- Resource Object File

System Action: ACIF terminates.

User Response: Contact IBM Service.

System Programmer Response: No response is necessary.

APK404S THE ATTRIBUTE NAME USED IN *indexn* HAS AN IMPROPER USE OF QUOTES.

Explanation: An unpaired set of quotes was found in the attribute name for an **INDEX** parameter.

System Action: ACIF terminates.

User Response: Correct the **INDEX** parameter and resubmit the job.

System Programmer Response: No response is necessary.

APK405S A VALUE OF 'xxxxxxx' IS INVALID FOR PARAMETER 'xxxxxxx'.

Explanation: The value supplied for a parameter is invalid.

System Action: ACIF terminates.

User Response: Correct the parameter value and resubmit the job.

System Programmer Response: No response is necessary.

APK406S PARAMETER 'xxxxxxx' HAS TOO MANY DATA SETS SPECIFIED.

Explanation: More than 8 data sets have been supplied for the parameter.

System Action: ACIF terminates.

User Response: Correct the number of data sets and resubmit the job.

System Programmer Response: No response is necessary.

APK407S A RESTYPE PARAMETER OF 'xxx' IS NOT VALID.

Explanation: A resource type of NONE was found with another value in the **RESTYPE** parameter. Examples of other values are: **FONT**, **OVLY**, **FDEF**, or **PSEG**. A resource type of **NONE** cannot be specified with another value.

System Action: ACIF terminates.

User Response: Correct the **RESTYPE** parameter and resubmit the job.

System Programmer Response: No response is necessary.

APK408S A VIRTUAL STORAGE REQUEST WAS UNSUCCESSFUL - REQUEST SIZE

storage request size RETURN CODE return code.

Explanation: A GETMAIN macro made an unsuccessful attempt to obtain virtual storage. This message indicates the storage size and the return code from the system GETMAIN macro.

System Action: ACIF terminates.

User Response: Increase the REGION size and resubmit the job.

System Programmer Response: To interpret the GETMAIN return code, refer to *MVS/Extended Architecture Supervisor Services and Macro Instructions* or *MVS/ESA System Programming Library: Application Development Macros*.

APK409S A DDNAME FOR {MSGDD | PARMDD} WAS NOT SUPPLIED. {SYSPRINT | SYSIN} WAS USED.

Explanation: No DDname was specified for either the **MSGDD** or the **PARMDD** parameter.

System Action: If the missing DDname was **MSGDD**, the DDname assigned to **SYSPRINT** was used. If the missing DDname was **PARMDD**, the DDname assigned to **SYSIN** was used.

User Response: If the DDname used was not acceptable, specify a DDname for the parameter and submit the job again.

System Programmer Response: No response is necessary.

APK410S AN ACIF STORAGE REQUEST WAS UNSUCCESSFUL - REQUEST SIZE
storage request size, request type RETURN CODE return code.

Explanation: An unsuccessful attempt has been made to obtain/free ACIF subpool storage. This error message returns the following information:

- Storage request size
- Request type
- Return code

System Action: ACIF terminates.

User Response: No response is necessary.

System Programmer Response: Use the information provided in the message to correct the error and resubmit the job.

APK411S AN ERROR OCCURRED WHILE ATTEMPTING TO { READ FROM|WRITE TO|CLOSE } THE DDNAME xxxxxxxx, RETURN CODE return code.

Explanation: The file I/O macro made an unsuccessful attempt to write to the named DD. The return codes are listed below:

Return Codes	
0	- Successful
1	- Permanent I/O error
2	- Specified number of bytes is zero or negative
3	- Invalid data buffer address
4	- Address not word aligned
6	- Invalid FILE_CB@
7	- Invalid MODE parameter
8	- Data record longer than LRECL or buffer
9	- File is not supported type
10	- Storage allocation/deallocation failed
11	- Invalid record number
12	- End of file detected
13	- Disk is full
14	- RECFM invalid
20	- Invalid file id

Return Codes	
28	- File not found
51	- Length exceeds maximum
310	- File format invalid

System Action: ACIF terminates.

User Response: Use the information provided in the return code to correct the problem.

System Programmer Response: No response is necessary.

APK412W MODULE *module name* HAS RETURNED WITH RETURN CODE *return code*.

Explanation: A non-zero return code has been returned from the called module. This message indicates that an abnormal occurrence has taken place in the called module. This message is informational and further action will take place in higher level modules if required.

System Action: None; this message is for information only.

User Response: See the accompanying message to determine a response.

System Programmer Response: No response is necessary.

APK413S ATTEMPTED { OPEN|CLOSE|READ|WRITE } OF RESOURCE FILE '*ddname*', RESOURCE MEMBER NAME '*member name*' FAILED. RETURN CODE *nnnn*.

Explanation: An attempt to open, close, read, or write a resource failed. This message indicates that an abnormal occurrence has taken place in the called module. This message is informational and further action will take place in higher level modules if required.

Return Codes	
0	- Successful
1	- Permanent I/O error
2	- Specified number of bytes is zero or negative
3	- Invalid data buffer address
4	- Address not word aligned
6	- Invalid FILE_CB@
7	- Invalid MODE parameter
8	- Data record longer than LRECL or buffer
9	- File is not supported type
10	- Storage allocation/deallocation failed
11	- Invalid record number
12	- End of file detected
13	- Disk is full
14	- RECFM invalid
20	- Invalid file id
28	- File not found
51	- Length exceeds maximum
310	- File format invalid

System Action: None; this message is for information only

User Response: See the accompanying message to determine a response.

System Programmer Response: No response is necessary.

APK414I THE FOLLOWING PARAMETERS WILL BE USED FOR THIS RUN:

Explanation: This message is issued before APK415I, APK416I, and APK417I to begin the listing of the parameters to be used for this run.

System Action: None

User Response: No response is necessary.

System Programmer Response: No response is necessary.

APK415I *parameter = value.*

Explanation: For this run, the parameter listed has been used with the associated value.

System Action: None

User Response: No response is necessary.

System Programmer Response: No response is necessary.

APK416I THESE { DATA SETS|FILETYPES } HAVE BEEN SPECIFIED FOR *library name.*

Explanation: This message is issued before message APK417I and shows the DD name or file type for a specific resource type contained in the ACIF parameter file.

System Action: None

User Response: No response is necessary.

System Programmer Response: No response is necessary.

APK417I { DATASETNAME|FILETYPE }: *name*

Explanation: This message follows APK416I and lists the name of the data set or file type for a particular resource type.

System Action: None

User Response: No response is necessary.

System Programmer Response: No response is necessary.

APK418S THE MAXIMUM RECORD ID WAS EXCEEDED.

Explanation: The current job contains more than 999 999 999 documents.

System Action: ACIF terminates.

User Response: Break the job up into a smaller number of documents.

System Programmer Response: No response is necessary.

APK419S USER {INPUT|OUTPUT|RESOURCE } EXIT *program* RETURNED CODE *nnnn.*

Explanation: The indicated user exit program has returned a non-zero return code.

System Action: ACIF terminates.

User Response: Correct the error in the exit program and resubmit the job.

System Programmer Response: No response is necessary.

APK424I PARAMETER 'RESFILE=PDS' IS VALID ONLY IN MVS, DEFAULTING TO 'RESFILE=SEQ'.

Explanation: The supplied value for the RESFILE parameter is incorrect for the VM operating system.

System Action: ACIF produces a sequential resource file.

User Response: No response is necessary.

System Programmer Response: No response is necessary.

APK425S USER *xxxxxx* EXIT '*xxxxxxxx*' WAS NOT FOUND.

Explanation: The user exit program named on the exit's DD parameter does not exist.

System Action: ACIF terminates.

User Response: Correct your exit program and resubmit the job.

System Programmer Response: No response is necessary.

APK426S PARAMETER MISMATCH: RESTYPE *type* SPECIFIED, BUT NO SUPPORTING LIBRARY DEFINITIONS WERE SUPPLIED.

Explanation: The resource type *type* was specified on the RESTYPE parameter, but no DD parameter for that resource type was supplied in the ACIF parameter file.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

APK427I AN ERROR OCCURRED WITH FILEDEF
'filename', RETURN CODE = rc, THE
DEFAULT OF *'fn' 'ft' 'fm'* FOR *'DDname'*
WILL BE USED.

Explanation: An invalid *filename* was supplied. The defaults listed will be used instead.

System Action: ACIF continues.

User Response: No response is necessary.

System Programmer Response: No response is necessary.

APK428S A 'resource' HAS BEEN REQUESTED,
BUT NO NAME WAS GIVEN.

Explanation: The resource listed in the message was requested to be handled by ACIF, but the name to get was not passed to ACIF. This condition is caused by an ACIF logic error.

System Action: ACIF terminates.

User Response: Contact IBM Service.

System Programmer Response: No response is necessary.

APK435W THE *ddname* DD STATEMENT SPECIFIED
FOR *parameter* IS MISSING.

Explanation: An ACIF DD parameter specified a DDname that was not specified in the JCL (MVS and VSE) or FILEDEF statement (VM).

System Action: ACIF terminates.

User Response: Ensure that the ACIF parameter specifies a DDname that is defined in the job commands.

System Programmer Response: No response is necessary.

APK436S THE GROUPNAME VALUE '*value*' IS NOT
WITHIN THE ALLOWABLE RANGE.

Explanation: ACIF processing has encountered the **GROUPNAME** parameter with an invalid **INDEX** number specified. The **INDEX** range is 1–8.

System Action: ACIF terminates.

User Response: Correct the resource and resubmit the job.

System Programmer Response: No response is necessary.

APK440I ACIF HAS COMPLETED PROCESSING
NORMALLY WITH RETURN CODE *nn*.

Explanation: ACIF processing has completed with the return code shown.

System Action: This message is for information only.

User Response: See any accompanying messages to determine a response.

System Programmer Response: No response is necessary.

APK441I ACIF HAS COMPLETED PROCESSING
ABNORMALLY WITH RETURN CODE *nn*.

Explanation: ACIF processing has completed with the return code shown.

System Action: This message is for information only.

User Response: See any accompanying messages to determine a response.

System Programmer Response: No response is necessary.

APK448S INDEXING WAS REQUESTED, BUT
NEITHER 'TRIGGER1' NOR ANY 'FIELD'
WAS SATISFIED WITHIN THE PAGE
RANGE SPECIFIED BY THE
INDEXSTARTBY PARAMETER.

Explanation: Indexing was requested, but the first **INDEX** satisfier was outside the range of pages specified in the **INDEXSTARTBY** parameter.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

APK449S INDEX FIELDS REFERENCE OUTSIDE OF
THE RECORD, FIELD# *nn* INPUT
RECORD# *nnnnnn*

Explanation: The **FIELDn** value specified on the **INDEXn** parameter references an area that is outside the length of the requested record.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

**APK450S A REQUIRED ACIF PARAMETER
parameter name WAS NOT FOUND IN
THE PARAMETER FILE.**

Explanation: A required ACIF parameter was not found in the parameter file.

System Action: ACIF terminates.

User Response: Add the missing parameter to the parameter file and resubmit.

System Programmer Response: No response is necessary.

**APK451S FILE { ALLOCATION | CONCATENATION
| OUTADD } ERROR DURING *ddname*
PROCESSING. SVC 99 ERROR *nnnn*
INFORMATION CODE *nnnn*.**

Explanation: An error occurred during the allocation, concatenation, or outadd of AFP resource libraries.

System Action: ACIF terminates.

User Response: Inform your system programmer that this error occurred.

System Programmer Response: Use the return code and reason code to determine the cause of the error and information code, and the appropriate response. Refer to *MVS/ESA Application Development Guide: Authorized Assembler Language Programs* for information on the SVC 99.

**APK452S A TRIGGER NUMBER OF *nnn* IS INVALID
FOR FIELD*n***

Explanation: The trigger number specified in the field parameter is invalid.

System Action: ACIF terminates.

User Response: Triggers used in field definitions must be defined. Correct the parameter and rerun ACIF.

System Programmer Response: No response is necessary.

**APK453S THE *xxxxxxxxnn* LENGTH OF *nnnn* IS
GREATER THAN THE ALLOWED
MAXIMUM OF *nnnn*.**

Explanation: The combined length of all of the FIELD*n* values on an INDEX*n* parameter is too long.

System Action: ACIF terminates.

User Response: Check the FIELD and INDEX parameters to find where this happens. Correct the parameter and resubmit the job.

System Programmer Response: No response is necessary.

**APK454S A VALUE OF *nnn* IS INVALID FOR
*xxxxxnn***

Explanation: A parameter value contains invalid characters.

System Action: ACIF terminates.

User Response: Correct the parameter value and resubmit the job.

System Programmer Response: No response is necessary.

**APK455S FIELD*n* USED BY INDEX*nn* WAS NOT
DEFINED.**

Explanation: An INDEX*n* parameter referred to a FIELD*n* that was not defined in the parameter file.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

**APK456S THE TRIGGER1 RELATIVE RECORD
NUMBER IS NOT EQUAL TO ASTERISK.**

Explanation: The record number associated with the TRIGGER1 parameter was not an asterisk.

System Action: ACIF terminates.

User Response: Correct the parameter and resubmit the job.

System Programmer Response: No response is necessary.

**APK457S TRIGGER1 WAS NOT DEFINED, BUT
SECONDARY TRIGGERS ARE PRESENT.**

Explanation: TRIGGER1 must be specified if secondary TRIGGERs are present.

System Action: ACIF terminates.

User Response: If no indexing is required, delete all TRIGGERs from the parameter file, otherwise supply a TRIGGER1 parameter for this run of ACIF.

System Programmer Response: No response is necessary.

**APK458S A NON-LITERAL VALUE OF *xxxxxxx*
HAS BEEN SUPPLIED FOR *xxxxxnn***

Explanation: The supplied TRIGGER value was not a literal.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

APK460S TRIGGERS SATISFIED, BUT INDEXES WERE INCOMPLETE AT END-OF-FILE.

Explanation: The **TRIGGERn** parameters specified in the parameter file were met, but the end of the file was reached before the **INDEXn** parameters were located.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

APK461S TRIGGER SUPPLIED, BUT ALL INDEX VALUES WERE LITERALS.

Explanation: A value for **TRIGGER** has been supplied, but all **INDEXn** values were literals.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

APK462S A TRIGGER PARAMETER WAS SPECIFIED, BUT THE INPUT FILE IS ALREADY INDEXED.

Explanation: The parameter file included a **TRIGGER** parameter, but the input file contains indexing structured fields. ACIF cannot index a file that is already indexed.

System Action: ACIF terminates.

User Response: If you want to create an index object file for the input file, remove all **TRIGGER** parameters from the ACIF parameter file and resubmit the job.

System Programmer Response: No response is necessary.

APK463S INDEXnn USED BY THE GROUPNAME PARAMETER WAS NOT DEFINED.

Explanation: The **INDEXn** specified by the **GROUPNAME** parameter was not defined.

System Action: ACIF terminates.

User Response: Correct the parameters and resubmit the job.

System Programmer Response: No response is necessary.

APK476I MESSAGE TEXT NOT AVAILABLE FOR MESSAGE NUMBER: nnnnnnnn

Explanation: ACIF attempted to write a message that is not defined in the message catalog.

System Action: ACIF processing continues depending upon the significance of undefined message.

User Response: Contact IBM service and inform them that ACIF attempted to write an undefined message. This situation should be corrected by IBM.

APK532S A FORM/PAGE DEFINITION WITH A MEMBER NAME nnnnnnnn WAS NOT FOUND - RETURN CODE nn, REASON CODE nn.

Explanation: The requested page or form definition does not exist in any of the available paths.

Return Codes

0	-	Successful
1	-	Permanent I/O error
2	-	Specified number of bytes is zero or negative
3	-	Invalid data buffer address
4	-	Address not word aligned
6	-	Invalid FILE_CB@
7	-	Invalid MODE parameter
8	-	Data record longer than LRECL or buffer
9	-	File is not supported type
10	-	Storage allocation/deallocation failed
11	-	Invalid record number
12	-	End of file detected
13	-	Disk is full
14	-	RECFM invalid
20	-	Invalid file id
28	-	File not found
51	-	Length exceeds maximum
310	-	File format invalid

Reason Codes

1	-	Resource name missing
2	-	File system open error
3	-	File system close error
4	-	File system read error
5	-	Storage module error
6	-	Resource type error
7	-	File system write error
8	-	Indexer error
9	-	Message write error

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

**APK599S INTERNAL ERROR in MODULE ____ AT
FUNCTION ____.**

Explanation: An internal error has occurred.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message. Make note of the module and function specified in the message.

APK610I GTF RETURN CODE = rc.

Explanation: Generalized Trace Facility (GTF) has returned a nonzero return code from the GTRACE request. The return code *rc* and *error text* explain the error. The return codes and error text are:

RC Error Text

04	inactive MVS GTF
08	invalid length = xxxx
0C	invalid data address = xxxx
10	invalid FID = xx
14	invalid EID = xx
18	no GTF buffer space
1C	invalid parameter address = xxxx
20	data paged out
xx	unknown GTF return code

System Action: The action depends on the return code; ACIF may or may not continue tracing. For return codes 18 and 20, GTF tracing continues. For the other return codes listed, GTF tracing stops. For unknown return codes, GTF tracing stops.

User Response: No response is necessary.

System Programmer Response: Refer to *MVS/Extended Architecture Service Aids Logic* or *MVS/ESA Component Diagnosis and Logic: Service Aids* for more information on the return codes.

APK900S MISSING DAT POINTER IN CCM.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK901S MISSING FORMDEF POINTER IN CCM.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK902S MISSING PAGEDEF POINTER IN CCM.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK903S MISSING OBJECT STACK POINTER IN
CCM.**

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK904S MISSING CODE PAGE POINTER IN CCM.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK905S MISSING FONT METRIC POINTER IN
CCM.**

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK906S UNEXPECTED OTHERWISE STATEMENT
ENCOUNTERED.**

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK907S CCM CANNOT FIND REQUESTED
MEDIUM MAP.**

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK908S CCM CANNOT FIND REQUESTED DATA MAP.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK909S CCM CANNOT FIND REQUESTED MEG.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK910S INPUT BIN LIST CHANGED DURING PROCESSING.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK911S DAT DID NOT SPECIFY ANY INPUT BIN INFORMATION.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK912S OVERLAY LOCAL ID HAS BEEN CHANGED IN LIST.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK913S STARTING COPY COUNT EXCEEDS TOTAL COPIES IN MM.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

APK914S CONDITIONAL PROCESSING INFORMATION PASSED TO CCM AT DOCUMENT INTERFACE BUT PAGEDEF DOES NOT REQUEST CONDITIONAL PROCESSING.

Explanation: An internal error has occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM Service and inform them that you have received this message indicating an internal error.

Part 4. Additional Information Common to the AIX, MVS, VM, and VSE Environments

Appendix A. Helpful Hints

This chapter contains General-use Programming Interface and Associated Guidance Information.

When using ACIF, the following topics may prove to be helpful to you:

- Working with control statements that contain numbered lines (MVS, VM, VSE environments only)
- Understanding how ACIF processes fonts
- Understanding how ACIF processes unbounded box fonts (3800)
- Placing TLEs in named groups
- Working with file transfer and AIX (AIX environment only)
- Understanding how ANSI and machine carriage controls are used
- Understanding common methods of transferring files to AIX from other systems
 - Physical media such as tape
 - PC file transfer program
 - FTP
- Invoke Medium Map (IMM) structured field
- Indexing considerations
- Concatenating the resource file and the document.

Working with control statements that contain numbered lines

(If you work in the AIX environment, you may skip this section, as the information about control statements pertains only to the MVS, VM, VSE environments.)

You sometimes can receive unexpected results when data set names are continued and the control statements have line numbers in columns 73-80, because ACIF reads all 80 columns of the control statements for processing purposes. (ACIF attempts to use the line number as a data set name, and issues MSGAPK451S and MSGAPK417I with a numeric value.) To resolve this problem, remove any line numbers from the control statements and rerun the job, or use a comment indicator (“/*”) before each line number.

Understanding how ACIF processes fonts

ACIF will always read the font library, even if you did not request that fonts be saved in the resource object file. The reason for this action is that ACIF creates MO:DCA-compliant output files. In MO:DCA-compliant files, the font list structured field, Map Coded Font, must be an MCF format 2 (x'D3AB8A'), which does not support any means of specifying host-style coded font names. ACIF must read the coded font object to obtain the names of the code page and character set.

Understanding how ACIF processes unbounded box fonts (3800)

When ACIF reads a font object (coded font, character set) from a library, it searches for the bounded box version of the font (X0, C0 prefix), regardless of the specifications in the page definition or document MCF record. ACIF does not know what device might be displaying or printing the document. ACIF also does not know whether the correct version of fonts in the library is bounded-box or unbounded-box. The reason ACIF searches the bounded box version is that all printer models except the d/t3800 accept bounded-box fonts. The resulting print file will print correctly, regardless of whether the document contains bounded or unbounded font names, because PSF knows the destination printer type and uses the corresponding fonts.

You should be aware that if only unbounded-box fonts are present in the libraries defined for FONTLIB and USERLIB, ACIF will be unable to locate a bounded-box equivalent, in which case, ACIF issues MSGAPK413 and terminates the job. Therefore, use a bounded-box font library if it is available. If, however, a bounded-box font library is not available, you can allow ACIF to find the correct font and build the output page correctly, but prevent an attempt to access C0 character sets by doing the following:

- Make copies of the unbounded-box CODED fonts
- Rename the copies to begin with X0
- Omit the collection of fonts from the **RESTYPE** parameter.
(Do *not* specify **RESTYPE=ALL** or **RESTYPE=FONT**.)

Note: The archiving of unbounded-box fonts is *not* recommended. No current or future MO:DCA receiver (printer or viewer, for example) will support unbounded-box fonts.

Placing TLEs in named groups

To avoid having your job terminated by ACIF, IBM recommends that you place page-level TLEs inside named groups, using one named group per page.

You should be aware that if you request **INDEXLEVEL=ALL** for a job that has an input data set containing composed (AFPDS) pages, page-level TLEs (TLE records after the AEG) and no named groups (BNG/ENG), your job may end with MSGAPK410S or MSGAPK408S. The reason for this action is that no named groups are present, and the page-level TLE records must be collected in memory until the end of the input document or file. MO:DCA index structures contain the extent (size) of the object being indexed. Indexed objects are delimited by a named group (or end document-EDT). If no named groups are present, ACIF will continue to build the index in memory. If the input file is large enough, there will not be enough memory, and ACIF will terminate. The ACIF memory manager currently limits the number (but not the size) of memory blocks that can be allocated; therefore, increasing REGION size may not alleviate the problem.

Working with file transfer and AIX

(If you work outside the AIX environment, you may skip this section, as the information about working with file transfer does not pertain to the MVS, VM, or VSE environments.)

ACIF needs to know two things about a file in order to print it:

- How long is each print record
- What kind of carriage control is used

As simple as this sounds, it is the source of most of the difficulty people have printing with ACIF in an AIX environment.

ACIF processes print records. A record is a sequence of contiguous characters, usually representing a printed line or a MO:DCA (AFPDS) structured field². Each record has a defined boundary or length. Some files contain information in each record that describes the record's length; these are called variable-length files. Other files require an external definition of length; these are called fixed-length files.

- **Variable-length files**

Variable-length files may use a length prefix, which means they contain a prefix that identifies the length of the record in the file. Each record contains a field that gives the length of the record. If the record contains a length, that length must be a prefix for each record and it must be a 16-bit binary number that includes the length of the 2-byte length prefix. Use the **FILEFORMAT=RECORD** control statement to identify files with length prefixes.

Variable-length files may use a separator or delimiter to indicate the end of a record, instead of using a length prefix. All of the bytes up to, but not including, the delimiter are considered to be part of the record. For AIX, the delimiter is X'0A'. If the file uses EBCDIC encoding, the newline character is X'25'.

Use the **FILEFORMAT=STREAM** control statement to designate files that use newlines to indicate record boundaries.

ACIF reads the first six bytes and tests for all ASCII characters³, to determine if a file is encoded in ASCII or EBCDIC. If no non-ASCII characters are found, ACIF assumes the file uses the ASCII newline character, X'0A'. Otherwise, ACIF assumes the file uses the EBCDIC newline character, X'25'. Because an input file can misguide ACIF, either intentionally or by accident, a set of rules has been established to determine how ACIF will interpret how a file will be processed. The following combinations are possible:

² Structured fields are similar to print commands.

³ code points from X'00' to X'7F'

Data Type	Newline Character
All EBCDIC	EBCDIC X'25'
All EBCDIC	ASCII X'0A' (Note 1)
All ASCII	EBCDIC X'25' (Note 1)
All ASCII	ASCII X'0A'

Note 1: These combinations are possible only if a file contains a prefix with a string that indicates a different code set than actually exists. For EBCDIC data with ASCII newlines, use X'0320202020200A'. For ASCII data with EBCDIC newlines, use X'03404040404025'.

- **Fixed-length files**

Fixed-length files contain records that are all the same length. No other separators or prefixes or self-identifying information exists that indicates the record length. You must know the record length and use the **FILEFORMAT=RECORD,nnn** control statement, where *nnn* represents the length of each record.

For variable- and fixed-length files using length prefixes, MO:DCA structured fields are treated as a special case. All such structured fields are self-identifying and contain their own length. They need not contain a length prefix to be correctly interpreted, but will be processed correctly if there is a length prefix.

Understanding how ANSI and machine carriage controls are used

In many environments (including IBM mainframes and most minicomputers), printable data normally contains a carriage control character. The carriage control character acts as a vertical tab command to position the paper at the start of a new page, at a specified line on the page, or to control skipping to the next line. The characters can be one of two types: ANSI carriage control or machine carriage control.

- **ANSI carriage control characters**

The most universal carriage control is ANSI, which consists of a single character that is a prefix for the print line. The standard ANSI characters are:

ANSI	Command
space	Single space the line and print
0	Double space the line and print
-	Triple space the line and print
+	Don't space the line and print
1	Skip to channel 1 (the top of the form, by convention)
2-9	Skip to hardware-defined position on the page
A,B,C	Defined by a vertical tab record or FCB

Note that all ANSI control perform the required spacing before the line is printed. ANSI controls may be encoded in EBCDIC (for ACIF, **CCTYPE=A**) or in ASCII (**CCTYPE=Z**).

- **Machine carriage control characters**

Machine carriage controls were originally the actual hardware control commands for IBM printers, and are often used on non-IBM systems. Machine controls are literal values, not symbols. They are not represented

as characters in any encoding and, therefore, machine controls cannot be translated. Typical machine controls are:

Machine	Command
X'09'	Print the line and single space
X'11'	Print the line and double space
X'19'	Print the line and triple space
X'01'	Print the line and don't space
X'0B'	Space one line immediately (don't print)
X'89'	Print the line, then skip to channel 1 (top of form, by convention)
X'8B'	Skip to channel 1 immediately (don't print)

Note that machine controls print before performing any required spacing. There are many more machine control commands than ANSI. Carriage controls may be present in a print file or not, but every record in the file must contain a carriage control if the controls are to be used. If the file contains carriage controls, but **CC=NO** is specified to ACIF, the carriage controls will be treated as printing characters. If no carriage controls are specified, the file will be printed as though it were single spaced.

Understanding common methods of transferring files to AIX from other systems

You can transfer files from other systems to AIX using a variety of methods. Each method results in a different set of possible outputs. Some methods produce output that cannot be used by ACIF. Methods commonly used to transfer files from other systems to AIX and produce output that ACIF can use are:

- Physical media (such as tape)
- PC file transfer program
- FTP

Physical media

Normally, you can copy fixed-length files without any transformation using a physical media such as tape. For variable-length files, however, either the creator of the tape or the copy program must include a 2-byte binary length as a prefix to each record.

PC file transfer program

You may transfer files from other systems to AIX by using an implementation of the most common PC file transfer program (IND\$FILE). You may also transfer files from a host to a personal computer. The variety of possible parameters that can affect printing are host-dependent. IBM recommends the following:

- For MVS and VM/CMS files, the default is binary.
- For CICS and VSE, binary is recommended.
- For files with fixed-length records, binary is recommended (you must know the record length).
- For files with variable-length records that contain only printable characters and either ANSI carriage control characters, or no carriage control characters:
 - Use ASCII and CRLF

- Specify the control statement **INPEXIT=asciipe** to remove the otherwise unprintable carriage return (X'0D') that is inserted in the file.
- For VSE files, additional file transfer parameters are available.
- For files with machine carriage control, you can specify **BINARY**, **CRLF** and **CC**. This provides an EBCDIC file with correct carriage controls separated by ASCII newlines and carriage returns. You must, however, “trick” ACIF by using a prefix of X'0320202020200A'.

FTP

From most systems, FTP works similarly to PC file transfer, and most of the same options are provided. Also, when executing FTP on an AIX system, you can omit the extraneous carriage return. However, you must test and check your implementation; some FTPs use **IMAGE** as a synonym for **BINARY**.

Other Considerations for Transferring Files to AIX

Conventional file transfer programs cannot correctly handle the combination of variable-length files, which contain bytes that cannot be translated from their original representation to ASCII, and may also contain machine control characters, mixed line data and structured fields, or special code points that have no standard mapping⁴. Your best solution is to either NFS-mount the file, or write a small filter program on the host system that appends the 2-byte record length to each record and transfer the file binary.

Generally, NFS-mounted files are not translated. However, NFS includes a 2-byte binary record length as a prefix for variable-length records. (Check your NFS implementation; you may have to use special parameters.)

Note: Some NFS systems do not supply the binary record length for fixed-length files.

ACIF treats a file that contains only structured fields (MO:DCA or AFPDS or LIST3820) as a special case. You can always transfer such a file as binary with no special record separator, and ACIF can always read it because structured fields are self-defining, containing their own length; ACIF handles print files and print resources (form definitions, fonts, page segments, overlays, and so on) in the same way.

Invoke Medium Map (IMM) Structured Field

Retrieval programs must be able to detect which medium map is active, to ensure that pages are reprinted (or viewed) using the correct medium map. To ensure that the correct medium map is used, use the Active Medium Map triplet and the Medium Map Page Number triplet (from the appropriate Index Element [IEL] structured field in the index object file), which designate the name of the last explicitly invoked IMM structured field and the number of pages produced since the IMM was invoked. The retrieval system can use this information to dynamically create IMM structured fields at the appropriate locations when it retrieves a group of pages from the archived document file.

⁴ When ASCII is specified, for example, the file transfer program may destroy the data in translation. When binary is specified, the file transfer program may not be able to indicate record lengths.

Indexing Considerations

The index object file contains Index Element (IEL) structured fields that identify the location of the tagged groups in the print file. The tags are contained in the Tagged Logical Element (TLE) structured fields.

The structured field offset and byte offset values are accurate at the time ACIF creates the output document file. However, if you extract various pages or page groups for viewing or printing, you will have to dynamically create from the original a temporary index object file that contains the correct offset information for the new file. For example, assume the following:

- ACIF processed all the bank statements for 6 branches, using the account number, statement date, and branch number.
- The resultant output files were archived using a system that allowed these statements to be retrieved based on any combination of these three indexing values.

If you wanted to view all the bank statements from branch 1, your retrieval system would have to extract all the statements from the print file ACIF created (possibly using the IELs and TLEs in the index object file) and create another document for viewing. This new document would need its own index object file containing the correct offset information. The retrieval system would have to be able to do this.

Under some circumstances, the indexing that ACIF produces may not be what you expect, for example:

- If your page definition produces multiple-up output, and if the data values you are using for your indexing attributes appear on more than one of the multiple-up subpages, ACIF may produce two indexing tags for the same physical page of output. In this situation, only the first index attribute name will appear as a group name, when you are using the Viewer application of AFP Workbench. To avoid this, specify a page definition that formats your data without multiple-up when you submit the indexing job to ACIF.
- If your input file contains machine carriage control characters, and you use the new-page carriage control character as a TRIGGER, the indexing tag created will point to the page on which the carriage control character was found, not to the new page created by the carriage control character.
- If your input file contains application-generated separator pages (for example, banner pages), and you want to use data values for your indexing attributes, you can write an Input Data exit program to remove the separator pages. Otherwise, the presence of those pages in the file will make the input data too unpredictable for ACIF to reliably locate the data values. As alternatives to writing an exit program, you can also change your application program to remove the separator pages from its output, or you can use the **INDEXSTARTBY** parameter to instruct ACIF to start indexing on the first page after the header pages.
- If you want to use data values for your indexing attributes, but none of the values appear on the first page of each logical document, you can cause ACIF to place an indexing tag on the first page by defining a FIELD parameter with a large enough negative relative record number from the anchor record to “page” backward to the first page. Without referencing this FIELD parameter in an

INDEX parameter, the tag generated by any INDEX parameter will be placed on the first page.

Concatenating the Resource Group to the Document

You can create a print file containing all the required print resources by concatenating the output document file to the end of the resource file. Do, however, remember two things when doing this:

- First, although Workbench Viewer and the other PSF products support all types of inline resources, PSF/VSE supports only inline page definitions and form definitions.
- Second, the offset information in the index object file applies to the document; that is, to the Begin Document (BDT) structured field. The offset information also applies to the file I/O level, because a single document is in the output document file. When you concatenate these two files, the offset information in the index object file no longer applies to the resultant file; that is, you cannot use this information to randomly access a given page or page group without first determining the location of the BDT structured field. This is not a problem for Workbench Viewer, because it removes any inline objects before using the offset information.

Specifying the IMAGEOUT Parameter

ACIF converts IM1 format images in the input file, in overlays, and in page segments to uncompressed IOCA format, if **IMAGEOUT=IOCA** (the default) is specified. An uncompressed IOCA image may use a significantly higher number of bytes than an IM1 image and may take more processing time to convert, especially for shaded or patterned areas. Although IOCA is the MO:DCA-P standard for image data, and some data stream receivers may require it, all products may not accept IOCA data. All software products from the IBM Printing Systems Company do, however, accept IOCA data as well as IM1 image data.

IBM recommends that you specify **IMAGEOUT=ASIS**, unless you have a specific requirement for IOCA images.

Appendix B. Data Stream Information

General-use Programming Interface and Associated Guidance Information is contained in this appendix.

This appendix describes the Tag Logical Element (TLE) structured field and the formats of the resource data sets.

Tag Logical Element (TLE) Structured Field

TLE structured fields are allowed only in AFP data stream (MO:DCA-P) documents. AFP Application Programming Interface (AFP API) supports the TLE structured field and can be used from host COBOL and PL/I applications to create indexed AFP data stream (MO:DCA-P) documents. Document Composition Facility (DCF), with APAR PN36437, can also be used to insert TLE structured fields in an output document.

The format of the TLE structured field that ACIF supports and generates is as follows:

Carriage Control Character (X'5A')

Specifies the carriage control character, which is required in the first position of the input record to denote a structured field.

Structured Field Introducer (8 bytes)

Specifies the standard structured-field header containing the structured field identifier and the length of the entire structured field, including all of the data.

Tag Identifier Triplet (4–254 bytes)

Specifies the application-defined identifier or attribute name associated with the tag value. An example is 'Customer Name'. This is a Fully Qualified Name triplet (X'02') with a type value of X'0B' (Attribute Name). For more information, refer to *Mixed Object Document Content Architecture Reference*.

Tag Value Triplet (4–254 bytes)

Specifies the actual value of the index attribute. If the attribute is 'Customer Name', the actual tag value might be 'Bob Smith'. This triplet contains a length in byte 1, a type value of X'36' (Attribute Value) in byte 2, and 2 reserved bytes (X'0000').

The following is an example of a 39-byte TLE structured field containing two index values. For the purposes of illustration, each field within the structured field is listed on a separate line. X' ' denotes hexadecimal data, and " " denotes EBCDIC or character data.

```
X'5A0026D3A090000000'  
X'11020B00'  
"Customer Name"  
X'0D360000'  
"Bob Smith"
```

TLE structured fields can be associated with a group of pages or with individual pages. Consider a bank statement application. Each bank statement is a group of pages, and you may want to associate specific indexing information at the statement level (for example, account number, date, customer name, and so on).

You may also want to index (tag) a specific page within the statement, such as the summary page. The following is an example of a print file that contains TLEs at the group level as well as at the page level:

```
BDT
  BNG
    TLE Account #, 101030
    TLE Customer Name, Mike Smith
    BPG
      Page 1 data
    EPG
    BPG
      Page 2 data
    EPG
    ...
    ...
    BPG
      TLE Summary Page, n
      Page n data
    EPG
  ENG
  ...
EDT
```

ACIF can accept as input files that contain both group-level and page-level indexing tags. In the case where ACIF indexes the print file, it creates only TLE structured fields for a group of pages; it does not support indexing specific pages. The only way to index both individual pages and groups of pages is through changes to the application, for example, using the AFP Application Programming Interface. You can also use the input record exit of ACIF to insert TLE structured fields into an AFP data stream (MO:DCA-P) file, where applicable. The indexing information in the TLE structured field applies to the page or group containing them. In the case of groups, the TLE structured field can appear anywhere between a Begin Named Group (BNG) structured field and the first page (BPG structured field) in the group. In the case of composed-text pages, the TLE structured field can appear anywhere following the Active Environment Group, between the End Active Environment (EAG) and End Page (EPG) structured fields. Although ACIF does not limit the number of TLE structured fields that can be placed in a group or page, you should consider the performance and storage ramifications of the number included.

ACIF does not require the print file to be indexed in a uniform manner; that is, every page containing TLE structured fields does not have to have the same number of tags as other pages or the same type of index attributes or tag values. This allows a great deal of flexibility for the application. When ACIF completes processing a print file that contains TLE structured fields, the resultant indexing information file may contain records of variable length.

Format of the Resources File

ACIF retrieves referenced AFP resources from specified libraries and creates a single file that contains these resources. Using ACIF, you can control the number of resources as well as the type of resources in the file by using a combination of **RESTYPE** values and processing in the resource exit.

ACIF can retrieve all the resources used by the print file and can place them in a separate resource file. The resource file contains a resource group structure whose syntax is as follows:

```
BRG
  BR
    AFP Resource 1
  ER
  BR
    AFP Resource 2
  ER
  ..
  BR
    AFP Resource n
  ER
ERG
```

ACIF does not limit the number of resources that can be included in this object, but available storage is certainly a limiting factor.

Begin Resource Group (BRG) Structured Field

ACIF assigns a null token name (X'FFFF') to this structured field and also creates three additional triplets: an FQN type X'01' triplet, an Object Date and Time Stamp triplet, and an FQN type X'83' triplet. The FQN type X'01' triplet contains the data set name identified in the DDname statement for **RESOBJDD**. The Object Date and Time Stamp triplet contains date and time information from the operating system on which ACIF runs. The date and time values reflect when ACIF was invoked to process the print file. The FQN type X'83' triplet contains the AFPDS output print file name identified by the DDname specified in the **OUTPUTDD** parameter.

Begin Resource (BR) Structured Field

ACIF uses this structured field to delimit the resources in the file. ACIF also identifies the type of resource (for example, overlay) that follows this structured field. The type is represented as a 1-byte hexadecimal value where:

- X'40'** Specifies a font character set.
- X'41'** Specifies a code page.
- X'FB'** Specifies a page segment.
- X'FC'** Specifies an overlay.
- X'FE'** Specifies a form definition.

End Resource (ER) and End Resource Group (ERG) Structured Fields

ACIF always assigns a null token name (X'FFFF') to the Exx structured fields it creates. The null name forces a match with the corresponding BR and BRG structured fields.

Appendix C. Format of the Index Object File

General-use Programming Interface and Associated Guidance Information is contained in this appendix.

One of the optional files ACIF can produce contains indexing, offset, and size information. The purpose of this file is to enable applications such as archival and retrieval applications to selectively determine the location of a page group or page within the AFP data stream print file, based on its index (tag) values.

The following example shows the general internal format of this object:

```
BDI
  IEL GroupName=G1
    TLE (INDEX1)
    ...
    TLE (INDEXn)
      IEL PageName=G1P1
        TLE (INDEX1)
        ...
        TLE (INDEXn)
      ...
      IEL PageName=G1Pn
    ...
  IEL GroupName=Gn
    TLE (INDEX1)
    ...
    TLE (INDEXn)
      IEL PageName=GnP1
        TLE (INDEX1)
        ...
        TLE (INDEXn)
      ...
      IEL PageName=GnPn
EDI
```

The example illustrates an index object file containing both page-level and group-level Index Element (IEL) structured fields. Because ACIF can create TLE structured fields only at the group level, the print file that was processed by ACIF to create this example of an index object file already contained page-level and group-level TLE structured fields.

Group-Level Index Element (IEL) Structured Field

If **INDEXOBJ=GROUP** is specified, ACIF creates an index object file with the following format:

```
BDI
  IEL Groupname=G1
    TLE
    ...
    TLE
  ...
  IEL Groupname=Gn
    TLE
    ...
    TLE
EDI
```

This format is useful to reduce the size of the index object file, but it allows manipulation only at the group level; that is, you cannot obtain the offset and size information for individual pages. You also lose any indexing information (TLEs) for pages; the TLE structured fields for the pages still exist in the output print file, however.

Page-Level Index Element (IEL) Structured Field

If **INDEXOBJ=ALL** is specified, ACIF creates an index object file with the following format:

```
BDI
  IEL Groupname=G1
    TLE
    ...
    IEL Pagename=G1P1
      TLE
      ...
    IEL Pagename=G1Pn....
  ...
  IEL Groupname=Gn
    TLE
    ...
    IEL Pagename=GnP1
      ...
    IEL Pagename=GnPn
      TLE
      ...
EDI
```

This example contains IEL structured fields for both pages and groups. Notice that TLE structured fields are associated with both pages and groups. When ACIF performs the actual indexing function, it does not support page-level indexing; therefore, it cannot create TLE structured fields for individual pages. Consequently, it can create only page-level IEL structured fields without any associated TLE structured fields. In this example, where an application created an indexed AFP print file containing both page-level and group-level TLE structured fields, ACIF can create IEL structured fields for the appropriate TLE structured fields.

An index object file containing both page-level and group-level IEL structured fields can provide added flexibility and capability to applications that operate on the files created by ACIF. This type of index object file provides the best performance when you are viewing a file using the Viewer application of AFP Workbench.

Begin Document Index (BDI) Structured Field

ACIF assigns a null token name (X'FFFF') and an FQN type X'01' triplet to this structured field. The FQN type X'01' value is the file name identified by the DDname specified in the **INDEXDD** parameter. ACIF also creates an FQN type X'83' triplet containing the name of the AFP output print file, identified by the DDname specified in the **OUTPUTDD** parameter.

ACIF also creates a Coded Graphic Character Set Global Identifier triplet X'01' using the code page identifier specified in the **CPGID** parameter. For more information on the **CPGID** parameter, see page92 . ACIF assigns a null value (X'FFFF') to the Graphic Character Set Global Identifier.

Index Element (IEL) Structured Field

The IEL structured field associates indexing tags with a specific page or group of pages in the output document file. It also contains the byte and structured-field offset to the page or page group and the size of the page or page group in both bytes and structured-field count. The following is a list of the triplets that compose this structured field:

- FQN Type X'8D'

This triplet contains the name of the active medium map associated with the page or page group. In the case of page groups, this is the medium map that is active for the first page in the group, because other medium maps can be referenced after subsequent pages in the group. If no medium map is explicitly invoked with an Invoke Medium Map (IMM) structured field, ACIF uses a null name (8 bytes of X'FF') to identify the default medium map; that is, the first medium map in the form definition.

- Object Byte Extent (X'57')

This triplet contains the size, in bytes, of the page or group this IEL structured field references. The value begins at 1.

- Object Structured Field Extent (X'59')

This triplet contains the number of structured fields that compose the page or group referenced by this IEL structured field. In the host environment, each record contains only one structured field, so this value also represents the number of records in the page or group. The value begins at 1.

- Direct Byte Offset (X'2D')

This triplet contains the offset, in bytes, from the start of the output print file to the particular page or group this IEL structured field references. The value begins at 0.

- Object Structured Field Offset (X'58')

This triplet contains the offset, in number of structured fields, from the start of the output print file to the start of the particular page or group this IEL structured field references. The value begins at 0.

- FQN Type X'87'

This triplet contains the name of the page with which this IEL structured field is associated. The name is the same as the FQN type X'01' on the BPG structured field. This triplet applies **only** to page-level IEL structured fields.

- FQN Type X'0D'

This triplet contains the name of the page group with which this IEL structured field is associated. The name is the same as the FQN type X'01' on the BNG structured field. This triplet applies **only** to group-level IEL structured fields.

- Medium Map Page Number (X'56')

This triplet defines the relative page count since the last Invoke Medium Map (IMM) structured field was processed or from the logical invocation of the default medium map. In the case of page groups, this value applies to the first page in the group. The value begins at 1 and is incremented for each page.

Tag Logical Element (TLE) Structured Field

ACIF creates TLE structured fields as part of its indexing process, or it can receive these structured fields from the input print file. When ACIF creates TLE structured fields, the first TLE structured field is **INDEX1**, the next TLE structured field is **INDEX2**, and so on, to a maximum of eight per page group. When ACIF processes a print file that contains TLE structured fields, it always outputs the TLE structured fields in the same order and position. The TLE structured fields in this object are exactly the same as those in the output document file, and they follow the IEL structured field with which they are associated.

End Document Index (EDI) Structured Field

ACIF assigns a null token name (X'FFFF') to this structured field, which forces a match with the BDI structured field name.

Appendix D. Format of the Output Document File

This appendix contains General-use Programming Interface and Associated Guidance Information.

Although ACIF can create three separate output files, only one of the files is required. ACIF always creates a print file in AFP data stream format. In doing so, ACIF may create the following structured fields:

- Tag Logical Element (TLE)
- Begin Named Group (BNG)
- End Named Group (ENG)

The TLE was described in appendix C; the other two structured fields will be described later in this appendix. The examples on the next two pages illustrate the two possible AFP data stream document formats ACIF may produce.

```

BDT
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        Page 1 of group 1
      EPG
      BPG
        Page 2 of group 1
      EPG
      ...
      BPG
        Page n of group 1
      EPG
  ENG
  ...
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        Page 1 of group n
      EPG
      BPG
        Page 2 of group n
      EPG
      ...
      BPG
        Page n of group n
      EPG
  ENG
EDT

```

Figure 40. Example of Code Containing Group-Level Indexing

Figure 40 illustrates the only format ACIF can produce when it converts and indexes a print file, because ACIF supports indexing only at the group level.

```

BDT
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        TLE (INDEX1)
        ...
        TLE (INDEXn)
          Page 1 of group 1
      EPG
    BPG
      Page 2 of group 1
    EPG
    ...
    BPG
      TLE (INDEX1)
      ...
      TLE (INDEXn)
        Page n of group 1
    EPG
  ENG
  ...
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        Page 1 of group n
      EPG
    BPG
      TLE (INDEX1)
      ...
      TLE (INDEXn)
        Page 2 of group n
      EPG
    ...
    BPG
      Page n of group n
    EPG
  ENG
EDT

```

Figure 41. Example of Code Containing Group- and Page-Level Indexing

Figure 41 illustrates an input file that has already been indexed (tagged) and converted to MO:DCA-P format, which the AFP API program can do. This example shows that you can index (tag) both groups and pages from an application.

Page Groups

Page groups are architected groups of one or more pages to which some action or meaning is assigned. Consider the example of the bank statement application. Each bank statement in the print file comprises one or more pages. By grouping each statement in a logical manner, you can assign specific indexing or tag information to each group (statement). You can then use this grouping to perform actions such as archival, retrieval, viewing, preprocessing, postprocessing, and so on. The grouping also represents a natural hierarchy. In the case of the Viewer application of AFP Workbench, you can locate a group of pages and then locate a page within a group. If you again use the example of the bank statement application, you can see how useful this can be. You can retrieve from the archival (storage) system all of the bank statements for a specific branch. You can then select a specific bank statement (group-level) to view and select a tagged summary page (page-level).

Begin Document (BDT) Structured Field

When ACIF processes an AFP data stream print file, it checks for an FQN type X'01' triplet in the BDT structured field. If the FQN triplet exists, ACIF uses it; otherwise, ACIF creates one using the file name identified in the DDname statement for **OUTPUTDD**. ACIF uses the FQN value when it creates an FQN type X'83' triplet on the Begin Document Index (BDI) structured field in the index object file and on the Begin Resource Group (BRG) structured field in the resource file. Although the input file may contain multiple BDT structured fields, the ACIF output will contain only one BDT structured field. (The same is true of End Document (EDT) structured fields.)

In the case of line-mode files, ACIF creates the BDT structured field. ACIF assigns a null token name (X'FFFF') and creates an FQN type X'01' triplet using the file name identified in the DDname statement for **OUTPUTDD**.

ACIF also creates a Coded Graphic Character Set Global Identifier triplet X'01' using the code page identifier specified in the **CPGID** parameter. For more information on the **CPGID** parameter, see page92 . ACIF assigns a null value (X'FFFF') to the Graphic Character Set Global Identifier.

ACIF also creates two additional FQN triplets for the resource name (type X'0A') and the index object name (type X'98'). These two values are the same as those contained in their respective type X'01' triplets on the BDI and BRG structured fields.

Begin Named Group (BNG) Structured Field

When ACIF processes an AFP data stream print file containing page groups, it checks for an FQN type X'01' triplet on each BNG structured field. If the FQN triplet exists, ACIF uses the value when it creates an FQN type X'0D' triplet on the corresponding Index Element (IEL) structured field in the index object file. ACIF appends an 8-byte rolling sequence number to ensure uniqueness in the name. If no FQN triplet exists, ACIF creates one. Here too, ACIF appends a rolling, 8-byte EBCDIC sequence number to ensure uniquely named groups, up to a maximum of 99999999 groups within a print file.

When ACIF indexes a print file, it creates the BNG structured fields. It assigns a rolling 8-byte EBCDIC sequence number to the token name (for example, 00000001 where 1=X'F1'). The sequence number begins with 00000001 and is incremented by 1 each time a group is created. ACIF also creates an FQN type X'01' triplet by concatenating the specified index value (**GROUPNAME**) with the same sequence number used in the token name. If the value of the index specified in **GROUPNAME** is too long, the trailing bytes are replaced by the sequence number. This occurs only if the specified index value exceeds 242 bytes in length. A maximum of 99999999 groups can be supported before the counter wraps. This means that ACIF can guarantee a maximum of 99999999 unique group names.

Tag Logical Element (TLE) Structured Field

As was mentioned in a previous chapter, ACIF creates TLE structured fields as part of its indexing process, or it can receive these structured fields from the input print file. When ACIF creates TLE structured fields, the first TLE is **INDEX1**, the next TLE is **INDEX2**, and so on to a maximum of eight per page group. When ACIF processes a print file that contains TLE structured fields, it always outputs the TLE structured fields in the same order and position.

Begin Page (BPG) Structured Field

When ACIF processes an AFP data stream print file, it checks for an FQN type X'01' triplet on every page. If the FQN triplet exists, ACIF uses the value when it creates an FQN type X'87' triplet on the corresponding Index Element (IEL) structured field in the index object file. If one does not exist, ACIF creates one, using a rolling 8-byte EBCDIC sequence number. This ensures uniquely named pages up to a maximum of 99999999 pages within a print file. ACIF creates IEL structured fields for pages only if **INDEXOBJ=ALL** is specified.

When ACIF processes a line-mode print file, it creates the BPG structured fields. It assigns a rolling 8-byte EBCDIC sequence number to the token name (for example, 00000001, where 1=X'F1'). The sequence number begins with 00000001 and is incremented by 1 each time a group is created. ACIF also creates an FQN type X'01' triplet using the same sequence number value, and uses this value in the appropriate IEL structured field if **INDEXOBJ=ALL** is specified. A maximum of 99999999 groups can be supported before the counter wraps. This means that ACIF can guarantee a maximum of 99999999 unique group names.

End Named Group (ENG), End Document (EDT), and End Page (EPG) Structured Fields

ACIF always assigns a null token name (X'FFFF') to the Exx structured fields it creates. It does not modify the Exx structured field created by an application unless it creates an FQN type X'01' triplet for the corresponding Bxx structured field. In this case, it assigns a null token name (X'FFFF'), which forces a match with the Bxx name.

Output MO:DCA-P Data Stream

Regardless of the input data stream, ACIF always produces output files in the MO:DCA-P format. Each structured field in the file is a single record preceded by a X'5A' carriage control character. The following sections describe the required changes ACIF must make to support MO:DCA-P output format.

Composed Text Control (CTC) Structured Field

Because this structured field has been declared obsolete, ACIF ignores it and does not pass it to the output file.

Map Coded Font (MCF) Format 1 Structured Field

ACIF converts this structured field to an MCF Format 2 structured field. In the case of coded fonts, ACIF resolves the coded font into the appropriate font character set and code page pairs.

Map Coded Font (MCF) Format 2 Structured Field

ACIF does not modify this structured field, and it does **not** map any referenced GRID values to the appropriate font character set and code page pairs. This may affect document integrity in the case of archival, because no explicit resource names are referenced for ACIF to retrieve.

Presentation Text Data Descriptor (PTD) Format 1 Structured Field

ACIF converts this structured field to a PTD Format 2 structured field.

Inline Resources

MO:DCA-P does not support inline resources at the beginning of a print file (before the BDT structured field); therefore, inline resources must be removed. The resources will be saved and used as requested.

Page Definitions

Because page definitions are used only to compose line-mode data into pages, this resource is not included in the resource file. The page definition is not included because it is no longer needed to view or print the document file.

Glossary

Source Identifiers

This glossary includes definitions from the following sources:

- Definitions reprinted from the *American National Dictionary for Information Processing Systems* are identified by the symbol (A) following the definition.

Definitions reprinted from a published section of the International Organization for Standardization's *Vocabulary—Information Processing* or from a published section of the ISO *Vocabulary—Office Machines* are identified by the symbol (I) following the definition. Because many ISO definitions are also reproduced in the *American National Dictionary for Information Processing Systems*, ISO definitions may also be identified by the symbol (A).

- Definitions reprinted from working documents, draft proposals, or draft international standards of ISO Technical Committee 97, Subcommittee 1 (Vocabulary) are identified by the symbol (T) following the definition, indicating that final agreement has not yet been reached among its participating members.
- Definitions that are specific to IBM products are so labeled, for example, "In SNA," or "In VM."

References

The following cross references are used in this glossary:

Contrast with. This refers to a term that has an opposite or substantively different meaning.

See. This refers the reader to multiple-word terms in which this term appears.

See also. This refers the reader to terms that have related, but not synonymous, meanings.

Synonym for. This appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

Synonymous with. This appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning.

A

ACIF. See Advanced Function Presentation Conversion and Indexing Facility.

Advanced Function Presentation (AFP). A set of licensed programs that use the all-points-addressable concept to print data on a wide variety of printers or display data on a variety of display devices. AFP also includes creating, formatting, archiving, viewing, retrieving, and distributing information.

Advanced Function Presentation Application Programming Interface. An AFP program shipped with PSF/MVS 2.1.1, PSF/VM 2.1.1, and PSF/VSE 2.2.1 programs that creates the AFP data stream from the COBOL and PL/I high-level programming languages.

Advanced Function Presentation Conversion and Indexing Facility. An AFP program you can use to convert a print file into a MO:DCA-P document, to retrieve resources used by the document, and to index the file for later retrieval and viewing.

Advanced Function Presentation data stream. A presentation data stream that is processed in the AFP environment. MO:DCA-P is the strategic AFP interchange data stream. IPDS is the strategic AFP printer data stream.

Advanced Function Presentation Workbench for OS/2 and Windows. (1) An IBM-licensed PC product that allows you to see AFP output in a WYSIWYP (what-you-see-is-what-you-print) format. (2) A platform for the integration of AFP-enabling applications and services.

AFP. See Advanced Function Presentation.

AFP API. Advanced Function Presentation Application Programming Interface

AFPS. A term formerly used to identify the composed page, MO:DCA-P-based data stream interchanged in AFP environments.

AIX (Advanced Interactive Executive). An IBM operating system.

anchor point. The point in a document that signals to ACIF the beginning of a group of pages, after which it adds indexing structured fields to delineate this group.

ANSI. American National Standards Institute

architecture. The set of rules and conventions that govern the creation and control of data types such as text, image, graphics, font, fax, color, audio, bar code, and multimedia.

ASCII. American National Standard Code for Information Interchange data encoding, which is the normal (default) type of data encoding in an AIX environment. Contrast with EBCDIC.

B

Bar Code Object Content Architecture (BCOCA). An architected collection of control structures used to interchange and present bar code data.

BCOCA. See Bar Code Object Content Architecture.

C

character. One set of symbols used for representing, organizing, or controlling data. Characters can be letters, digits, punctuation marks, or other symbols.

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) Often a synonym for font character set. See coded font.

carriage control character. An optional character in an input data record that specifies a write, space, or skip operation.

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 and default character.

coded font. An AFP font that associates a code page and a font character set.

copies. See copy group.

copy group. In Print Services Facility, an internal object in a form definition that identifies the overlays and defines page placement and modifications to the form such as paper source or the number of copies.

D

data stream. (1) All information (data and control commands) sent over a data link, usually in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

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 Document Composition Facility (DCF).

download. (1) To transfer programs or data from a computer to a connected device, typically a personal computer. (T) (2) To transfer data from a computer to a connected device, such as a workstation or a microcomputer. Contrast with upload.

E

EBCDIC. Extended binary-coded decimal interchange code. This is the normal (default) type of data encoding in an MVS, VM, or VSE environment. Contrast with ASCII.

F

font. (1) A family of characters of a given size and style. For example, 9-point Helvetica. (T) (2) See font character set.

font character set. Part of an AFP font that contains the raster patterns, identifiers, and descriptions of characters. Often synonymous with character set. See also coded-font.

form definition (FORMDEF). A resource that defines the characteristics of the form which include: overlays to be used (if any), text suppression, and position of page data on the form, and the number and modifications of a page. Synonymous with FORMDEF, medium map. Contrast with page definition.

G

GOCA. See Graphic Object Content Architecture.

Graphic Object Content Architecture (GOCA). An architecture that provides a collection of graphics values and control structures used to interchange and present graphics data.

group. A named collection of sequential pages that form a logical subset of a document.

H

hexadecimal. Pertaining to a numbering system with a 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

image. An electronic representation of a picture formed by toned and untoned pels. An image can also be generated directly by software without reference to an existing picture.

Image Object Content Architecture (IOCA). An architected collection of constructs used to interchange and present images.

indexing. In ACIF, a process of matching reference points within a file and creating structured field tags within the MO:DCA-P document and the separate index object file.

index object file. A file created by ACIF that contains Index Element (IEL) structured fields, which identify the location of the tagged groups in the AFP file. The indexing tags are contained in the Tagged Logical Element (TLE) structured fields.

indexing with data values. Adding indexing tags to a MO:DCA-P document using data that is already in the document and that is consistently located in the same place in each group of pages.

indexing with literal values. Adding indexing tags to a MO:DCA-P document by assigning literal values as indexing tags, because the document is not organized such that common data is located consistently throughout the document.

IOCA. See Image Object Content Architecture

J

JCL. job control language.

JES. Job Entry Subsystem.

job control language (JCL). A language of control statements used to identify a computer job or describe its requirements to the operating system.

Job Entry Subsystem (JES). A licensed program that receives jobs into the system and processes all output data produced by the jobs.

L

library. A data file that contains files and control information that allows them to be accessed individually.

licensed program. A utility that performs a function for the user and usually interacts with and relies upon system control programming or some other IBM-provided control program. A licensed program

contains logic related to the user's data and is usable or adaptable to meet specific requirements.

line data. Data prepared for printing on a line printer such as an IBM 3800 Printing Subsystem Model 1. Line data is usually characterized by carriage control characters and table reference characters. Contrast with MO:DCA-P data.

M

Mixed Object Document Content Architecture. A strategic, architected, device-independent data stream for interchanging documents.

MO:DCA-P. Mixed Object Document Content Architecture

Multiple Virtual Storage (MVS). Multiple Virtual Storage, consisting of MVS/System Product Version 1 and the MVS/390 Data Facility Product operating on a System/390 processor.

MVS. Multiple Virtual Storage.

O

offset. The number of measuring units from an arbitrary starting point in a record, area, or control block to some other point.

object. (1) 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, graphics, and image objects. (2) A resource or a sequence of structured fields contained within a larger entity, such as a page segment or a composed page.

Operating System/2. IBM's operating system for the IBM Personal System/2 or a compatible.

OS/2. See Operating System/2.

outline fonts. (1) Fonts whose graphic character shapes are defined as mathematical equations rather than by raster patterns. (2) Fonts created in the format described in *Adobe Type 1 Font Format*, a publication available from Adobe Systems Inc. Synonymous with Type 1 fonts.

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.

P

page. Part of an AFP document bracketed by a pair of Begin Page and End Page structured fields.

page definition. A resource containing a set of formatting controls for printing logical pages of data. Includes controls for number of lines per printed sheet, font selection, print direction, and mapping individual fields in the data to positions on the printed sheets.

page segment. An AFP 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 the object in which it is included. See also image.

parameter. A variable that is given a constant value for a specified application and that may denote the application.

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 of an inch used in measuring typographical material.

point size. The height of a font in points. See also point.

Print Services Facility (PSF). PSF is a sophisticated IBM print subsystem that drives IPDS page printers. PSF is supported under MVS, VSE, VM, OS/2, AIX, and is a standard part of the operating system under OS/400. PSF manages printer resources such as fonts, images, electronic forms, form definitions, and page definitions, and provides error recovery for print jobs.

When printing line data, PSF supports external formatting using page definitions and form definitions. This external formatting extends page printer functions such as electronic forms and use of typographic fonts without any change to applications programs.

Print Services Facility/2 (PSF/2). PSF/2 is an OS/2-based print server that drives IPDS page printers as well as IBM PPDS and HP-PCL compatible printers. PSF/2 manages printer resources and provides error recovery for print jobs. PSF/2 supports distributed printing of AFP print jobs from PSF/MVS, PSF/VSE, PSF/VM, and OS/400. PSF/2 also supports printing from a wide range of workstation applications, including Microsoft Windows and OS/2 Presentation Manager, as well as the ASCII, PostScript, and AFPDS data streams.

PSF. See Print Services Facility.

PSF/2. See Print Services Facility/2.

R

resource. A collection of printing instructions and sometimes data to be printed consisting entirely of structured fields. A resource can be stored as a member of a library and can be called for by Print Services Facility when needed. Coded fonts, font character sets, code pages, page segments, overlays, form definitions, and page definitions are all resources.

rotation. The number of degrees a graphic character is turned relative to the page coordinates.

S

structured field. A self-identifying, variable-length, bounded record that can have a content portion that provides control information, data, or both.

syntax. The rules and keywords that govern the use of a programming language.

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.

trigger. Data values for which ACIF searches, to delineate the beginning of a new group of pages. The first trigger is then the anchor point from which ACIF locates the defined index values. See *anchor point*.

typeface. (1) A specific type style, such as Helvetica or Times New Roman. (2) One of the many attributes of a font, others, for example, being size and weight. (3) A collection of fonts, each having a different height or size of character sets. See also fonts.

typographic font. A typeface originally designed for typesetting systems. Typographic fonts are usually proportionally spaced fonts.

V

Viewer application. An application on AFP Workbench that runs under WIN-OS/2 or Microsoft Windows.

Virtual Machine. A functional simulation of a computer and its associated devices.

Virtual Storage Extended. The notion of storage space that can be regarded as addressable main storage by the user of the computer system in which addresses are mapped to real addresses.

VM. Virtual Machine.

VSE. Virtual Storage Extended.

W

Workbench Viewer. See Advanced Function Presentation Workbench for OS/2 and Windows

Index

A

- ACIF
 - See AFP Conversion and Indexing Facility (ACIF)
- ACIF command
 - See *also* acif command, AIX
 - notational conventions xiv
- acif command, AIX
 - automatically invoking 28
 - defined, AIX 29
 - flags, AIX 29
 - format 25
 - line2afp 28
 - notational conventions 25
 - parameters 25
 - running, AIX 62
 - syntax rules 25
 - using 25
- ACIF exit
 - See exits
- ACIF input record exits
 - See input record exit
- ACIF JCL statement
 - defined 81
- Adobe Type 1 outline fonts 18
- AFP API
 - See AFP Application Programming Interface
- AFP Application Programming Interface
 - defined 13
 - group-level tags 19
 - page-level tags 19
 - support for the TLE structured field 181
 - Tag Logical Element structured field 181
- AFP Conversion and Indexing Facility (ACIF)
 - application planning 3
 - archiving files 10
 - asciinp, AIX 68
 - asciinpe, AIX 68
 - batch application development utility xi
 - conversion functions 10
 - converting data streams 11
 - data path to prepare files for archiving 10
 - data path to prepare files for printing 8
 - data path to prepare files for retrieving 10
 - data path to prepare files for viewing 7
 - defined 3
 - definition xi
 - distributed printing 8
 - example application 111
 - example application, AIX 53
 - example processing parameters, AIX 56, 59
 - exit, AIX 68
 - AFP Conversion and Indexing Facility (ACIF)
 - (continued)
 - functions of 3
 - indexing functions 12
 - input record exit, AIX 68
 - input record exits, AIX 68
 - invoking program to index input file 113
 - message file 82, 84
 - messages 130
 - MVS JCL statement 81
 - output file format 196
 - overview, graphic 4
 - parameter file 82, 84
 - parameters syntax 87
 - parameters, MVS, VM, VSE 87
 - related products 17
 - relationship to other AFP products 4
 - retrieving files 10
 - retrieving resources 16
 - running a job, AIX 62
 - steps for using 6
 - syntax rules, MVS, VM, VSE 87
 - tasks 3
 - VM CMS commands 83
 - VSE JCL statements 85
- AFP data as input to ACIF 11
- AFP resources
 - having reserved prefixes 110
- AFP Workbench
 - 64-byte limit for attribute names 18
 - Adobe Type 1 outline fonts 18
 - considerations 18
 - group names used by 97
 - ignored objects 19
 - indexing for 98
 - indexing for, AIX 40
 - preparing files for 7
 - retrieving resources for 107
 - retrieving resources for, AIX 46
 - supports subset of MO:DCA-P 18
- AIX
 - acif command 25
 - ACIF input record exit 68
 - apka2e input record exit 68
 - ASCII fonts 77
 - ASCII input data 56
 - asciinp input record exit 68
 - asciinpe input record exit 68
 - CC parameter 29, 75
 - cctype parameter 29, 75
 - concatenation 62
 - converting literal values 61

AIX (continued)

- EBCDIC input data 58
- EBCDIC literal values 61
- example, ACIF application 53
- example, ASCII input data 56
- example, input file 55
- example, triggers for indexing 56, 59
- exit, index 69
- exit, input file 66
- exit, output record 71
- exit, resource retrieval 73
- exit, user programming 65
- file provided with ACIF 52
- font names, linking 77
- font names, mapping 77
- fonts, ASCII 77
- fonts, specifying 50
- FORMDEF parameter 75
- GROUPNAME parameter 38
- index exits 69
- index object file 39
- INDEX parameter 38
- index, exit 66
- indexing 66
- indexing files with acif command 52
- INDEXPP parameter 39
- input exits 66
- input file, example 55
- input file, exit 66
- input, user exit 65
- INPUTDD parameter 41
- Invoke Medium Map structured field 178
- invoking ACIF automatically 28
- library, locating resource directory 51
- line2afp 28
- linking font names 77
- locating resource directory 51
- mapping font names 77
- mounting directories on workstation 64
- MSGDD parameter 41
- non-zero return codes 75
- OUTEXIT parameter 42
- output record exits 71
- OUTPUTDD parameter 42
- OVLYLIB parameter 42
- PAGEDEF parameter 42, 75, 76
- PARMDD parameter 43
- PDEFLIB parameter 44
- print file attributes 75
- print file attributes, CC parameter 75
- print file attributes, CCTYPE parameter 75
- print file attributes, FORMDEF parameter 75
- print file attributes, PAGEDEF parameter 75, 76
- print file attributes, PRMODE parameter 75
- print file attributes, TRC parameter 76
- PRMODE parameter 44, 75

AIX (continued)

- processing parameter 55, 56
- PSEGLIB parameter 44
- RESEXIT parameter 45
- RESLIB parameter 45
- RESOBJDD parameter 46
- resource directory, locating 51
- resource provided with ACIF 73
- resource retrieval 73
- resource retrieval, example 50
- RESTYPE parameter 46
- return code, non-zero 75
- running a job 62
- search order for resources 27
- shell commands 61
- specifying fonts 50
- syntax rules 25
- system prerequisites 20
- tasks, ACIF application 53
- transferring files to workstation 63
- TRC parameter 47, 76
- triggers for indexing, example 56, 59
- user exit input 65
- user exit provided with ACIF 65
- user exit, print file attributes 75
- user programming exit 65
- viewing files, access 63

AIX acif command

- See acif command, AIX

AIX workstation

- See workstation

anchor record

- defined 15
- field parameter, AIX 34
- set by INDEX parameter 97
- set by index parameter, AIX 38
- set by TRIGGER parameter 108
- set by trigger parameter, AIX 47
- used with field parameter, AIX 34
- used with FIELDD parameter 93

ANSI carriage-control characters

- See carriage-control characters

apka2e exit

- AIX 68

apka2e exit program

- converting ASCII to EBCDIC, AIX 41

APKACIF CMS statement 84

application programmer

- converting data streams 11
- indexing 12
- planning ACIF application 3
- purpose of ACIF publication xi
- skills needed xi
- task, converting data streams 11
- task, indexing 12
- tasks, ACIF 3

- application programming publication, ACIF
 - audience xi
 - comments, how to submit xvii
 - order number xvii
 - organization, overview xii
 - reader comments welcomed xvii
 - related publications xvi
 - terms used xiii
 - users xi
 - who should use this xi
- archiving, ACIF
 - how to prepare files 10
 - indexing considerations 179
 - indexing data for 12
 - retrieving resources for 16
 - steps to perform 10
- ASCII
 - fonts, AIX 77
 - input data
 - specifying parameters for, AIX 56
 - input data to ACIF 12
- ASCIINP exit
 - input record, AIX 68
- ASCIINPE exit
 - input record, AIX 68
- attributes
 - indexing 98, 117
 - indexing, AIX 38
 - print file, AIX 75

B

- bars in commands xv
- BCOCA objects
 - ignored by the Viewer application of AFP Workbench 19
- Begin Document Index structured field
 - defined 187
- Begin Document structured field
 - defined 194
- Begin Named Group structured field
 - defined 194
- Begin Page structured field
 - defined 195
- Begin Resource Group structured field
 - described 183
- Begin Resource structured field
 - defined 183
- braces in commands xv
- brackets in commands xv
- BTD
 - See Begin Document structured field

C

- carriage-control characters
 - AIX 30, 90
 - encoded in ASCII, AIX 29, 90
 - encoded in EBCDIC, AIX 30, 90
 - indexing considerations 179
 - machine code, AIX 30, 90
 - specifying presence 89
 - specifying presence, AIX 29
 - specifying type 89
 - specifying type, AIX 29
- CC parameter
 - AIX 75
 - defined 89
 - flags and values, AIX 29
 - print file attributes, AIX 75
- CC print file attribute
 - default or specified value 129
- CCTYPE parameter
 - AIX 29, 75
 - defined 89
 - flags and values, AIX 29
 - print file attributes, AIX 75
- CCTYPE print file attribute
 - default or specified value 129
- CHARS parameter
 - AIX 30, 47
 - blanks or specified value 129
 - defined 90
 - font order 108
 - TRCs, font order 107
- CHARS print file attribute
- CMS commands
 - for ACIF job, VM 83
 - for concatenating VM files 120
 - for example 115
 - invoking ACIF program to index input file 113
 - USERAPPL 83
- code page
 - CPGID parameter 92
 - identifier, AIX 33
- coded fonts
 - code page names 128
 - font character sets 128
 - MCF-2 128
 - resource filtering 128
- COM setup file
 - inline, AIX 32
 - inline, MVS, VM, VSE 91
 - search parameter for, AIX 41
 - search parameter for, MVS 100
 - specified in setup parameter, AIX 32
 - Specified in setup parameter, MVS, VM, VSE 91
- commands
 - concatenation, AIX 62

- commands (*continued*)
 - converting literal values, AIX 61
 - format xiv
 - notation xiv
 - shell, AIX 61
 - shell, concatenation, AIX 62
- comments
 - in parameter file 88
- comments in parameter file, AIX 44
- Composed Text Control (CTC) structured field
 - obsolete 196
- concatenation
 - AIX 62
 - files, output, AIX 62
 - MVS files 120
 - output files, AIX 62
 - resource group to document 180
 - VM files 120
- conventions
 - highlighting xiv
 - notational xiv, 25
- converting
 - AFP data 11
 - AIX file, example 50
 - example, AIX file 50
 - files, ASCII data 12
 - line-mode data 11
 - literal values, AIX 61
 - mixed-mode data 11
 - MO:DCA-P data 11
- CPGID parameter
 - code page identifier, AIX 33
 - defined 92

D

- data set series, concatenated
 - See library
- data streams
 - AFP data stream 11
 - converting 10
 - line-mode data 11
 - mixed-mode data 12
 - MO:DCA-P 11
- DCB requirements
 - index object file 98
 - message file, MVS 82
 - message file, VM 84
 - output file 100
 - output file, MVS 82
 - resource file 106
- DCF
 - See Document Composition Facility (DCF)
- DCFPAGENAMES parameter
 - defined, AIX 33
 - defined, MVS, VM, VSE 92

- directory
 - See library
- disclaimer about examples x
- distributed printing
 - See printing, ACIF
- distributing, ACIF
- document
 - DD statement for, MVS 81
 - DD statement for, VM 83
 - defined xiii
 - output format 191
- Document Composition Facility (DCF)
 - group-level tags 19
 - page-level tags 19
- document file
 - DLBL statement for, VSE 86
- documentation
- double byte fonts
 - PRMODE parameter 104
 - requirement, AIX 44
- dummy form definition
 - AIX 37, 43
- dummy setup files
 - COM setup file name 32, 91

E

- EBCDIC
 - attribute name for indexing 98
 - input data, specifying parameters, AIX 58
 - using the shell with literal values, AIX 61
- EDI
 - See End Document Index structured field
- EDT
 - See End Document structured field
- End Document Index structured field
 - defined 189
- End Document structured field
 - defined 195
- End Named Group structured field
 - defined 195
- End Page structured field
 - defined 195
- End Resource Group structured field
 - defined 184
- End Resource structured field
 - defined 184
- ENG
 - See End Named Group structured field
- environment variables
 - PATH
 - index record exit, AIX 40
 - input record exit, AIX 40
 - output record exit, AIX 42
 - resource exit, AIX 45
 - PSFPATH, AIX 31

EPG
 See End Page structured field

ER
 See End Resource structured field

ERG
 See End Resource Group structured field

examples

- ACIF application, AIX 53
- ACIF output for indexed input file 193
- ACIF processing parameters, AIX 56, 59
- AFP document output formats 191
- ASCII input data, AIX 56
- blank characters in parameter file 87
- CMS commands 115
- CMS commands to invoke ACIF, VM 83
- conversion, AIX 50
- disclaimer x
- EBCDIC input data, AIX 59
- indexing using data values 14
- input print file attributes 129
- invoking ACIF program to index input file 113
- JCL, CMS commands, and ACIF processing parameters 113
- line-data application 111
- locating resource directories, AIX 51
- MVS JCL to invoke ACIF 81
- output record 125
- parameter file for ASCII input data, AIX 56
- parameter file for EBCDIC input data, AIX 59
- print file attributes, AIX 75
- resource retrieval, AIX 50
- specifying fonts, AIX 50
- VSE JCL commands 116

exits

- apka2e, AIX 68
- asciinp, AIX 68
- asciinpe, AIX 68
- excluded resources 127
- index record 99
- index, AIX 69
- input print file attributes 129
- input record 99
- input record, AIX 68
- input, AIX 40, 66
- load modules, MVS 128
- load modules, VM 128
- load modules, VSE 128
- non-zero return codes 129
- non-zero return codes, AIX 75
- output, AIX 42, 71
- print file attributes provided, AIX 75
- programs, MVS 128
- programs, VM 128
- programs, VSE 128
- provided with ACIF
 - index record 123
 - input record 121

exits (*continued*)

- provided with ACIF (*continued*)
 - output record 125
 - resource 126
- resource 17, 105, 127
- resource, AIX 45
- return codes, non-zero 129
- search order 128
- user programming 121

F

FDEF resource

- requirements for resource file 106

FDEFLIB parameter

- (equivalence of) VSE 93
- AIX 33
- defined 92, 93
- defined, AIX 33
- MVS 92
- VM 93
- VSE (equivalence of) 93

FIELD parameter

- defined, AIX 34
- specified in multiple INDEX parameters 97

FIELDn parameter 93

- xs.output record 100

fields

- example of 117
- field parameter, AIX 34
- FIELDn parameter 93
- INDEX parameter 98
- INDEX parameter, AIX 39

file

- access for viewing, AIX 63
- concatenating, AIX 62
- defined xiii
- message, ACIF 82
- output, AIX 42
- parameter, ACIF 82
- prepare for archiving 10
- prepare for printing 8
- prepare for retrieving 10
- prepare for viewing 7
- provided with ACIF, AIX 52

file extensions, search order for resources, AIX 27

FILEFORMAT parameter

- defined, AIX 35

flags

- acif command, AIX 29

FONT resource

- requirements for resource file 107

FONTLIB parameter

- AIX 36
- defined 94, 95
- defined, AIX 36

FONTLIB parameter (*continued*)

MVS 94
VM 95
VSE 95

fonts

converting ASCII to EBCDIC, AIX 41
directory, AIX 36
double-byte 104
double-byte requirements, AIX 44
example of specifying, AIX 50
for line data, AIX 31
library, AIX 36
library, MVS 94
library, VM 95
library, VSE 95
linking names, AIX 77
mapping font names, AIX 77
order specified, CHARS parameter 108
specifying with CHARS 90
specifying with chars, AIX 30
to print ASCII or S/370 line data, AIX 77

form definition

provided with ACIF 20

form definitions

inline 96
inline, AIX 37
library, MVS 92
library, VM 93
library, VSE 93
search path for line data transform, AIX 33
specified with FORMDEF parameter 95
specified with formdef parameter, AIX 31

format

of commands xiv, 25
of input file, AIX 35

FORMDEF parameter

AIX 75
defined 95
defined, AIX 36
print file attributes, AIX 75

FORMDEF print file attribute

blanks or specified value 129

G

GOCA objects

ignored by the Viewer application of AFP
Workbench 19

group-level IEL structured field 186

GROUPNAME parameter

defined 97
defined, AIX 37
defined, AIX index 38
index values, AIX 38

groups for indexing

defined 12, 13

groups for indexing (*continued*)

GROUPNAME parameter 97
GROUPNAME parameter, AIX 37
structured fields 183, 194

GTF trace 107

H

highlighting conventions xiv

I

IEL

See Index Element structured field

image output

IMAGEOUT parameter 97
parameter, AIX 38

IMAGEOUT parameter

defined 97
defined, AIX 38

IMM

See Invoke Medium Map structured field

INDEX

CMS statement, VM 83
defined, AIX parameter 38
indexing anchor record, AIX 38
JCL statement, MVS 82
literal values, AIX 38
MVS, JCL statement 82
parameter, AIX 38
VM, CMS statement 83

Index Element structured field

considerations 179
defined 188
group-level 186
index object file 179

index exit

AIX 40, 69

index object file

amount of information in 99
amount of information in, AIX 40
archiving considerations 179
DCB characteristics 98
DD statement for, MVS 82
DD statement for, VM 83
defined 13
defined, AIX INDEXPP parameter values 39
DLBL statement for, VSE 85
INDEXDD parameter, MVS and VM 98
INDEXDD parameter, VSE 98
INDEXPP parameter, AIX 39
structured fields 185
used by the Viewer application of AFP
Workbench 18

index record exit

INDEXOBJ parameter 99

- index record exit (*continued*)
 - INDEXEXIT parameter 99, 123
- INDEXDD parameter
 - AIX 39
 - defined 98
 - defined, AIX 39
 - MVS 98
 - VM 98
- indexing
 - anchor point 15
 - anchor record 93, 108
 - anchor record, AIX 34, 47
 - description 12
 - effect on document 191
 - example of 14, 115
 - example, ASCII input data, AIX 56
 - example, EBCDIC input data, AIX 59
 - field parameter, AIX 34
 - FIELDn parameter, VSE 93
 - files with acif command, AIX 52
 - functions 12
 - helpful hints 179
 - index exit, AIX 66
 - index parameter, AIX 38
 - INDEXn parameter 97
 - input files 13
 - limitations 16
 - record exit 123
 - specifying when ACIF should start, AIX 40
 - structured fields 181
 - TLE structured field 181
 - to print output file 20
 - trigger parameter, AIX 47
 - triggers 15
 - using data values 14
 - using literal values 14
 - with AFP API 19
 - with AFP Toolbox 19
 - with DCF 19
- indexing tags
 - actual location of indexing information, AIX 38
 - defined 13
 - End Resource Group structured field 184
 - End Resource structured field 184
 - example of 117
 - example, AIX 56, 59
 - INDEX parameter 97
 - INDEX parameter, AIX 38
 - structured fields 183, 184
- INDEXn parameter
 - defined 97
- INDEXOBJ parameter
 - defined 18
 - defined, AIX 40
 - information put in index object file 99
 - resources included in resource file 106
- INDEXSTARTBY parameter
 - AIX 40
 - defined 99
 - defined, AIX 40
- INDEXEXIT parameter
 - defined 99
- inline resources
 - COM setup files, AIX 32
 - COM setup files, MVS, VM, VSE 91
 - form definition 96
 - form definition, AIX 37
 - page definition 102
 - page definition, AIX 43, 44
 - PSF/MVS 20
 - PSF/VM 20
 - PSF/VSE 20
- INPEXIT parameter
 - AIX 40
 - defined 99
 - defined, AIX 40
- input
 - MVS 82
 - VM 83
- input data streams
 - AFP data 11
 - line-mode 11
 - mixed-mode 12
 - MO:DCA-P data 11
 - output of 10
- input exit 40
- input file
 - example, AIX 55
 - exit, AIX 66
 - format, specifying AIX 35
 - INPUTDD parameter, MVS, VM 99
 - INPUTDD parameter, VSE equivalent 99
 - newline character, AIX 35
 - specifying AIX format 35
 - specifying file name, AIX 41
- input record
 - exit 121
- input record exit
 - apka2e, AIX 68
- INPUTDD parameter
 - AIX 41
 - defined, AIX 41
 - equivalence of, VSE 99
 - MVS 99
 - VM 99
 - VSE, equivalence of 99
- Invoke Medium Map structured field 178

J

- JCL
 - ACIF JCL statement defined 81

JCL (*continued*)

- example, MVS 114
- example, VSE 116
- for ACIF job, MVS 114
- for ACIF MVS jobs 81
- for ACIF VSE jobs 85
- for concatenating MVS files 120
- in the VSE environment 85
- invoking ACIF program to index input file 113
- MVS example 114
- OUTPUT JCL statement defined 82
- PRINTOUT JCL statement defined 81
- statement defined, ACIF JCL 81
- statement defined, OUTPUT JCL 82
- statement defined, PRINTOUT JCL 81
- VSE example 116

L

library

- defined xiii
- example of locating, AIX 51
- font, MVS 94
- font, VM 95
- font, VSE 95
- font, AIX 36
- form definition, AIX 33
- form definition, MVS 92
- form definition, VM 93
- form definition, VSE 93
- mounting AIX directories on workstation 64
- MVS user 110
- overlay, AIX 42
- overlay, MVS 101
- overlay, VM 101
- overlay, VSE 102
- page definition, AIX 44
- page definition, MVS 103
- page definition, VM 103
- page definition, VSE 104
- page segment, AIX 44
- page segment, MVS 104
- page segment, VM 105
- page segment, VSE 105
- resource 51
 - example of locating, AIX 51
 - setup file, AIX 41
 - setup file, MVS 100

limitations

- indexing 16
- PSF/MVS 20
- PSF/VM 20
- PSF/VSE 20

limitations of indexing 16

line data

- carriage controls, AIX 29

line data (*continued*)

- fonts, AIX 31
- S/370 used by AIX 77
- search paths for resources
 - AFP resources, AIX 49
 - fonts, AIX 36
 - form definitions, AIX 33
 - overlays, AIX 42
 - page definitions, AIX 44
 - page segments, AIX 44
 - system resources, AIX 45

line mode data

- defined 11
- input to ACIF conversion 11
- input to ACIF indexing 11, 12
- single carriage control character 11
- single table reference character 11

line2afp command

- relation to acif command 28

literal values

- converting, AIX 61

M

machine code carriage-control characters, AIX 30, 90

Map Coded Font Format 1 structured field

- converted 196

Map Coded Font Format 2 structured field

- archival, document integrity 196
- coded fonts 128

mapping font names, AIX 77

MCF-1

- See Map Coded Font Format 1 structured field

MCF-2

- See Map Coded Font Format 2 structured field

member, partitioned data set

- See file

memory for ACIF job 84

message file

- DD statement for, MVS 82
- DD statement for, VM 84
- DLBL statement for, VSE 86

messages

- MVS, VM, VSE 130
- system programmer instructions xi

microfilm setup file

mixed-mode data

- as input to ACIF 12

MO:DCA-P data

- as input to ACIF 11

mounting AIX directories on the workstation 64

- resource, provided with ACIF, AIX 73

MSGDD parameter

- AIX 41
- defined, AIX 41

- multiple=up output
 - page definition 179
- MVS
 - DD statement for document file 81
 - INDEX JCL statement 82
 - index object file 82
 - input 82
 - invoking ACIF 81
 - JCL example 114
 - JCL for ACIF job 81
 - JCL statement 81
 - JCL to invoke ACIF 81
 - message file, ACIF 82
 - OUTPUT JCL statement 82
 - partitioned data set format 17
 - RESOBJ statement 82
 - SYSIN JCL statement 82
 - SYSPRINT JCL statement 82
 - system prerequisites 21
 - user exit load modules 128
 - USERAPPL statement 81
 - USERLIB parameter 110

N

- newline character, AIX 35
- NONE resource
 - requirements for resource file 107
- notational conventions xiv

O

- OBJCONLIB parameter
 - AIX 41
 - MVS 100
- organization
 - publication overview xii
- out-of-storage problem
 - See Tag Logical Element structured field
- OUTEXIT parameter
 - AIX 42
 - defined 100
 - defined, AIX 42
- OUTPUT CMS statement
 - VM 83
- output file
 - concatenating, AIX 62
 - format 191
 - MO:DCA-P-P data stream 196
 - OUTPUTDD parameter 100
 - specifying name, AIX 42
- OUTPUT JCL statement
 - defined, MVS 82
 - MVS 82
- output record exit
 - AIX 71

- output record exit (*continued*)
 - OUTEXIT parameter 100, 125
- OUTPUTDD parameter
 - AIX 42
 - defined, AIX 42
 - defined, MVS, VM 100
 - defined, VSE 100
 - MVS 100
 - VM 100
 - VSE 100
- overlay
 - directory, AIX 42
 - library, AIX 42
 - library, MVS 101
 - library, VM 101
 - library, VSE 102
- OVLY resource
 - requirements for resource file 106
- OVLYLIB parameter
 - AIX 42
 - defined, AIX 42
 - defined, MVS 101
 - defined, VM 101
 - defined, VSE equivalent 102
 - MVS 101
 - VM 101
 - VSE equivalent 102

P

- page definition
 - and resource file 196
 - directory, AIX 43
 - fonts 90
 - fonts, AIX 31
 - inline 102
 - inline, AIX 43
 - library, AIX 43
 - library, MVS 103
 - library, VM 103
 - library, VSE 104
 - multiple-up output 179
 - specified with PAGEDEF parameter 102
 - specified with pagedef parameter, AIX 42
- page segment
 - directory, AIX 44
 - library, AIX 44
 - library, MVS 104
 - library, VM 105
 - library, VSE 105
- page-level IELs 187
- PAGEDEF
 - PAGEDEF parameter
 - AIX 42, 75, 76
 - defined 102
 - defined, AIX 42

PAGEDEF parameter (*continued*)

print file attributes, AIX 75, 76

PAGEDEF print file attribute

blanks or specified value 129

parameter file

ASCII input data, AIX 56

comments 88

comments, AIX 44

DD statement for, MVS 82

DD statement for, VM 84

EBCDIC input data, AIX 58, 59

example 115, 116

syntax rules, MVS, VM, VSE 87

values spanning multiple records 88

parameter values

spanning multiple records 88

parameters

ACIF, AIX 29

ACIF, MVS, VM, VSE 87

ASCII input data processing, AIX 55

ASCII input data, AIX 56

CC, AIX 29

CC, MVS, VM, VSE 89

CCTYPE, MVS, VM, VSE 89

CHARS, AIX 30

CHARS, MVS, VM, VSE 90

COMSETUP, AIX 32

COMSETUP, MVS, VM, VSE 91

CPGID, MVS, VM, VSE 92

DCFPAGENAMES, AIX 33

DCFPAGENAMES, MVS, VM, VSE 92

FDEFLIB, AIX 33

FDEFLIB, MVS 92

FDEFLIB, VM 93

FDEFLIB, VSE 93

FIELD, AIX 34

FIELDn 98

FIELDn, multiple 97

FIELDn, MVS, VM, VSE 93

FONTLIB, MVS 94

FONTLIB, VM 95

FONTLIB, VSE 95

FORMDEF, MVS, VM, VSE 95

GROUPNAME, AIX 37

GROUPNAME, MVS, VM, VSE 97

IMAGEOUT, MVS, VM, VSE 97

INDEX, AIX 38

INDEXDD, AIX 39

INDEXDD, MVS 98

INDEXDD, VM 98

INDEXDD, VSE 98

INDEXn, MVS, VM, VSE 97

INDEXOBJ 18

INDEXOBJ, AIX 40

INDEXOBJ, MVS, VM, VSE 99

INDEXSTARTBY, AIX 40

parameters (*continued*)

INDEXSTARTBY, MVS, VM, VSE 99

INDEXEXIT 123

INDEXEXIT, AIX 40

INDEXEXIT, MVS, VM, VSE 99

INPEXIT 121

INPEXIT, AIX 40

INPEXIT, MVS, VM, VSE 99

INPUTDD, AIX 41

INPUTDD, MVS 99

INPUTDD, VM 99

INPUTDD, VSE 99

MSGDD, AIX 41

OBJCONLIB, AIX 41

OBJCONLIB, MVS 100

OUTEXIT 100, 125

OUTEXIT, AIX 42

OUTPUTDD, AIX 42

OUTPUTDD, MVS 100

OUTPUTDD, VM 100

OUTPUTDD, VSE 100

OVLYLIB, AIX 42

OVLYLIB, MVS 101

OVLYLIB, VM 101

OVLYLIB, VSE 102

PAGEDEF, AIX 42

PAGEDEF, MVS, VM, VSE 102

PARMDD, AIX 43

PDEFLIB, AIX 44

PDEFLIB, MVS 103

PDEFLIB, VM 103

PDEFLIB, VSE 104

PRMODE, AIX 44

PRMODE, MVS, VM, VSE 104

processing, ASCII input data, AIX 55

PSEGLIB, AIX 44

PSEGLIB, MVS 104

PSEGLIB, VM 105

PSEGLIB, VSE 105

RESEXIT 17

RESEXIT, AIX 45

RESEXIT, MVS, VM, VSE 105

RESFILE, MVS 105

RESLIB, AIX 45

RESOBDD, AIX 46

RESOBJDD, MVS 106

RESOBJDD, VM 106

RESOBJDD, VSE 106

RESTYPE 16, 126

RESTYPE, MVS, VM, VSE 106

TRACE, MVS, VM, VSE 107

TRACEDD, VM 107

TRACEDD, VSE 107

TRC, AIX 47

TRC, MVS, VM, VSE 107

TRIGGER, AIX 47

parameters (*continued*)

- TRIGGER, MVS, VM, VSE 108
- UNIQUEBNGS, AIX 49
- UNIQUEBNGS, MVS, VM, VSE 109
- USERLIB, AIX 49
- USERLIB, MVS 110

PARMDD parameter

- AIX 43
- defined, AIX 43

partitioned data set

- See library

partitioned data set (PDS), MVS 17

PATH environment variable

- defaults used for acif command, AIX 40, 42

paths

- font directory, AIX 36
- form definition directory, AIX 33
- inline, AIX 43
- overlay directory, AIX 42
- overlay library, AIX 42
- page definition directory, AIX 44
- page definition library, AIX 44
- page segment directory, AIX 45
- page segment library, AIX 45
- resource directory, AIX 45
- resource library, AIX 45
- setup file directory, AIX 41
- setup file library, AIX 41
- user directory, AIX 49
- user library, AIX 49

PDEFLIB parameter

- AIX 44
- defined, AIX 44
- defined, MVS 103
- defined, VM 103
- defined, VSE equivalent 104
- MVS 103
- VM 103
- VSE equivalent 104

piping symbols (bars) in commands xv

prefix, reserved

- AFP resources 110

prerequisites 20

- AFP concepts xi
- assumptions. user skills xi
- MO:DCA-P architecture and structured fields xi
- PSF print parameters xi
- terms used xiii

Presentation Text Data Descriptor structured field 196

print file attributes

- AIX 75
- CC 129
- CC parameter, AIX 75
- CCTYPE 129
- cctype parameters, AIX 75
- CHARS 129

print file attributes (*continued*)

- example 129
- FORMDEF 129
- formdef parameters, AIX 75
- input 129
- PAGEDEF 129
- pagedef parameters, AIX 75, 76
- PAGEDEF=P1TEST 130
- parameter, cc, AIX 75
- parameters, cctype, AIX 75
- parameters, formdef, AIX 75
- parameters, pagedef, AIX 75, 76
- parameters, prmode, AIX 75
- parameters, trc, AIX 76
- PRMODE 129
- prmode parameters, AIX 75
- TRC 130
- TRC parameters, AIX 76
- user exits 121
- user exits, AIX 75

printing, ACIF

- how to prepare files 8
- PSF/MVS limitations 20
- PSF/VM limitations 20
- PSF/VSE limitations 20

PRINTOUT JCL statement

- defined 81

PRMODE parameter

- AIX 44, 75
- defined 104
- defined, AIX 44
- print file attributes, AIX 75

PRMODE print file attribute

- blanks or specified value 129

PRNTOUT VSE statement 85

processing mode 44

- flds.example, AIX 56, 59
- PRMODE parameter 104
- resource, AIX 45

processing parameter

- AIX 55, 58, 59
- ASCII input data, AIX 55
- EBCDIC input data, AIX 58, 59
- example, AIX 56

programming interface information

- AFP Application Programming Interface (AFP API) 19
- AFP Toolbox C Library 19
- AFP Toolbox C++ Object Library 19
- overview ix

PSEG resource

- requirements for resource file 106

PSEGLIB parameter

- AIX 44
- defined, AIX 44
- defined, MVS 104

PSEGLIB parameter (*continued*)
 defined, VM 105
 defined, VSE equivalent 105
 MVS 104
 VM 105
 VSE equivalent 105
 PSF/VSE
 limitations 20
 PSFPATH environment variable, AIX 31

R

REGION size for ACIF 82
 rfile.DD statement for, VM 83
 related information
 Advanced Function Presentation publications xvii
 AIX publications xvi
 font publications xvi
 MO:DCA-P publications xvii
 MVS publications xvii
 VM publications xvii
 VSE publications xvii
 where to find xvi
 related products 17
 reserved prefix
 AFP resources 110
 RESEXIT parameter
 AIX 45
 defined 17, 105
 defined, AIX 45
 RESFILE parameter
 defined 105
 RESLIB parameter
 AIX 45
 defined, AIX 45
 RESOBJ VSE statement 86
 RESOBJDD parameter
 AIX 46
 defined 106
 defined, AIX 46
 MVS 106
 VM 106
 VSE 106
 RESOBJDD statement
 MVS 82
 VM 83
 resource directory
 example of locating, AIX 51
 resource exit
 filtering resources 17
 provided with ACIF, AIX 73
 RESEXIT parameter 17, 105
 resexit parameter, AIX 45
 RESTYPE parameter 126
 resource file
 AFP data-stream resource group 17

resource file (*continued*)
 contents set with RESTYPE parameter 106
 contents set with restype parameter, AIX 46
 DCB characteristics 106
 DD statement for, MVS 82
 DLBL statement for, VSE 86
 format 105, 183
 output format
 AFP data-stream resource group 17
 partitioned data set, MVS 17
 partitioned data set, MVS 17
 structured fields 183
 resource retrieval
 description 16
 example, AIX 50
 exit 105
 exit, AIX 45
 file format 105, 183
 RESEXIT parameter 105
 RESFILE parameter 105
 resource exit, AIX 73
 RESTYPE parameter
 AIX 46
 defined 16, 106
 defined, AIX 46
 specifying all resources for viewing or printing 17
 retrieving, ACIF
 how to prepare files 10
 return code, not zero
 AIX 75

S

sample code
 See examples
 sample VSE JCL to invoke ACIF 85
 search order for resources, AIX
 by file extension 27
 fonts, AIX 36
 form definitions, AIX 33
 objconlib, AIX 41
 OBJCONLIB, MVS 100
 overlays, AIX 42
 page definitions, AIX 44
 page segments, AIX 44
 resource directory, AIX 45
 resource library, AIX 45
 setup files, AIX 41
 setup files, MVS 100
 user directory, AIX 49
 user library, AIX 49
 separator pages
 application-generated 179
 removal from output 179
 sequential data set
 See file

- setup file
 - directory, AIX 41
 - directory, MVS 100
 - library, AIX 41
 - library, MVS 100
- shell commands
 - concatenation, AIX 62
 - EBCDIC literal values, AIX 61
 - for converting literal values, AIX 61
- shift-out, shift-in
 - AIX 44
- storage problem
 - See Tag Logical Element structured field
- structured fields
 - Begin Document 180, 194
 - Begin Document Index 187
 - Begin Named Group 191, 194
 - Begin Page 195
 - Begin Resource 183
 - Begin Resource Group 183
 - Composed Text Control (obsolete) 196
 - End Document 195
 - End Document Index 189
 - End Named Group 191, 195
 - End Page 195
 - End Resource 184
 - End Resource Group 184
 - group level 185
 - Index Element 179, 186, 187, 188
 - Invoke Medium Map 178
 - Map Coded Font Format 1 196
 - Map Coded Font Format 2 128, 196
 - page level 185
 - placing in the index object file 13
 - Presentation Text Data Descriptor 196
 - Tag Logical Element 121, 179, 181, 189, 191, 195
- syntax rules, AIX 25
- SYSIN JCL statement
 - MVS 82
- SYSPRINT JCL statement
 - MVS 82
- system resource directories
 - example of locating, AIX 51
 - paths, AIX 45

T

- table reference characters 107
 - AIX 47
 - nputf.example 113
- Tag Logical Element structured field
 - as part of the indexing process 121, 189, 195
 - created in the output document file 191
 - defined 189
 - examples and rules 181
 - in named groups 174
 - out-of-storage problem, possible cause 174

- Tag Logical Element structured field (*continued*)
 - in named groups (*continued*)
 - storage problem, possible cause 174
- tags for indexing
 - defined 13
 - example 117
 - example, AIX 56, 59
 - INDEXn parameter 97
 - structured fields 183
- tasks, ACIF application
 - ACIF application 111
 - ACIF application, AIX 53
- terminology
 - document xiii
 - file xiii
 - library xiii
- TLE
 - See Tag Logical Element structured field
- TRACE CMS statement 83
- TRACE parameter
 - defined 107
- TRACEDD parameter
 - defined, VM 107
 - defined, VSE 107
 - VM 107
 - VSE 107
- tracing ACIF
 - VM 107
 - VSE 107
- trademarks x
- transferring
 - AIX files to workstation 63
 - files to AIX workstation 63
- TRC parameter
 - AIX 47, 76
 - CHARS, determine font order 107
 - defined 107
 - defined, AIX 47
 - print file attributes, AIX 76
- TRC print file attribute
 - default or specified value 130
- TRIGGERn parameter
 - AIX 47
 - defined 108
- triggers for indexing
 - defined 15
 - example 117
 - example, AIX 56, 59
 - set by TRIGGER parameter 108
 - set by trigger parameter, AIX 47
 - used with field parameter, AIX 34
 - used with FIELDn parameter 93
 - used with INDEX parameter 97
 - used with index parameter, AIX 38

U

- underscore in commands xv
- unformatted ASCII
 - as input to ACIF
 - input to ACIF 12
 - search path for resources
 - AFP resources, AIX 49
 - fonts, AIX 36
 - overlays, AIX 42
 - page definitions, AIX 44
 - page segments, AIX 44
 - setup files, AIX 41
 - setup files, MVS 100
 - system resources, AIX 45
 - search paths for resources
 - form definitions, AIX 33
- UNIQUEBNGS parameter
 - AIX 49
 - MVS, VM, VSE 109
- user exits
 - excluded resources 127
 - index record 99, 123
 - index, AIX 40, 69
 - input print file attributes 121, 129
 - input record 99, 121
 - input, AIX 40, 65
 - load modules, MVS 128
 - load modules, VM 128
 - load modules, VSE 128
 - non-zero return codes 129
 - output record 100
 - output record, AIX 71
 - output, AIX 42
 - print file attributes, AIX 75
 - programs, MVS 128
 - programs, VM 128
 - programs, VSE 128
 - provided with ACIF, AIX 52, 65
 - resexit parameter, AIX 45
 - resource 17, 105, 126, 127
 - resource, provided with ACIF, AIX 73
 - return codes, non-zero 129
 - search order 128
- user library
 - AIX 49
 - fonts 91
 - form definitions 96
 - MVS 17
 - overlays 101
 - page definitions 102
 - page segments, MVS 104
 - requesting access to a user library 110
- user programming exit
 - AIX 65

- USERAPPL
 - CMS command, VM 83
 - MVS statement 81
- USERAPPL VSE statement 85
- USERLIB parameter
 - defined, MVS 110
 - MVS 110
- using ACIF
 - archiving 10
 - command, AIX 62
 - create files for archiving 10
 - create files for printing 8
 - create files for retrieving 10
 - create files for viewing 7
 - distributed printing 8
 - how to 6
 - in the MVS environment 81
 - in the VM environment 83
 - in the VSE environment 85
 - prepare files for archiving 10
 - prepare files for printing 8
 - prepare files for retrieving 10
 - prepare files for viewing 7
 - printing 8
 - retrieving 10
 - steps 6
 - to prepare files for printing 8
 - viewing 7
- utility, ACIF
 - batch application development xi
 - definition xi

V

- vertical bars in commands xv
- Viewer application of AFP Workbench
 - See AFP Workbench
- viewing, ACIF
 - accessing AIX files 63
 - AIX file access 63
 - considerations 18
 - how to prepare files 7
 - indexing data for 12
 - retrieving resources for 16
- virtual memory for ACIF job 84
- VM
 - CMS commands for ACIF job 81, 83
 - CMS commands in example 115
 - INDEX CMS statement 83
 - input 83
 - OUTPUT CMS statement 83
 - RESOBJ CMS statement 83
 - system prerequisites 21
 - user exit load modules 128
 - USERAPPL CMS command 83
 - using ACIF 83

VSE

- JCL example 116
- JCL for ACIF job 81
- system prerequisites 21
- user exit load modules 128

W

Workbench, Viewer application
See AFP Workbench

workstation

- mounting AIX directories on 64
- transferring files to, AIX 63

Readers' Comments — We'd Like to Hear from You

**IBM Print Services Facility
AFP Conversion and Indexing Facility:
User's Guide**

Publication No. S544-5285-00

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.

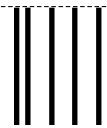


Cut or Fold
Along Line

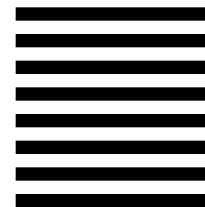
Fold and Tape

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

Cut or Fold
Along Line



File Number: S370-40
Program Number: 5648-062
5765-505



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.



S544-5285-00

