

Print Services Facility



AFP Conversion and Indexing Facility User's Guide

Print Services Facility



AFP Conversion and Indexing Facility User's Guide

Note

Before using this information and the product it supports, be sure to read the general information in “Notices” on page 217.

First Edition (November 2005)

This edition applies to AFP Conversion and Indexing Facility, which is included with Print Services Facility 4.1.0 for z/OS (Program Number 5655-M32), Print Services Facility/VM 2.1.1 (Program Number 5684-141), Print Services Facility/VSE 2.2.1 (Program Number 5686-040), Infoprint Manager for AIX 4.2.0 (Program Number 5765-F68), and Infoprint Manager for Windows 2.2.0 (Program Number 5639-N49). 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.

The IBM Printing Systems Division welcomes your comments. A form for reader's comments is provided at the back of this publication. If the form has been removed, you may send your comments to the following address:

INFORMATION DEVELOPMENT
THE IBM PRINTING SYSTEMS DIVISION
DEPARTMENT H7FE, BUILDING 004N
PO BOX 1900
BOULDER, COLORADO 80301-9191
U.S.A.

If you prefer to send comments electronically, use one of these methods:

- Internet: printpub@us.ibm.com
- Fax: 1-800-524-1519 or 1-303-924-6873

Internet

Visit our home page at <http://www.ibm.com/printers>

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, 2005. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	ix
Tables	xi
About This Publication	xiii
Who Should Read This Publication	xiii
How This Publication Is Organized	xiii
What Terms Are Used in This Publication	xiv
Understanding the Notational Conventions Used in This Publication	xv
Highlighting	xv
Syntax Notation.	xv
Related Information	xvi
Using LookAt to View Message Explanations.	xvi
Summary of Changes	xix
Chapter 1. Understanding ACIF	1
Overview	1
ACIF Functions	3
Converting Data Streams	3
Indexing Documents	5
Retrieving Resources	9
Scenarios for Processing ACIF Files	10
Preparing Files for Viewing	10
Preparing Files for Printing	11
Preparing Files for Archiving and Retrieval.	13
IBM Products Used with ACIF	14
AFP Workbench Viewer	14
AFP Toolbox.	16
Document Composition Facility (DCF)	16
What Are the System Considerations for ACIF?	16
System Limitations	16
System Prerequisites	17
Chapter 2. Using ACIF.	19
Using ACIF in AIX and Windows	19
Selecting Resources	19
Invoking ACIF	21
Files Provided with Infoprint Manager	22
NLS Messages	22
Suggested Reading	22
Using ACIF in z/OS	23
z/OS JCL Statements for Invoking ACIF.	23
Using ACIF in VM.	25
VM/CMS Commands for Invoking ACIF	25
Using ACIF in VSE	26
VSE JCL Statements for Invoking ACIF	26
Chapter 3. ACIF Parameters	29
Syntax Rules	29
Syntax Rules for AIX and Windows	30
Syntax Rules for z/OS, VM, and VSE	30
Parameter Values	31

CC	33
CCTYPE	33
CHARS	34
COMSETUP	36
CPGID	37
DCFPAGENAMES	38
EXTENSIONS	39
FDEFLIB	40
FIELDn	41
FILEFORMAT	43
FONTECH	44
FONTLIB	45
FONTPATH	46
FORMDEF	47
GROUPNAME	48
IMAGEOUT	49
INDEXn	49
INDEXDD	51
INDEXOBJ	52
INDEXSTARTBY	52
INDEXEXIT	53
INPCCSID	53
INPEXIT	53
INPUTDD	54
INSERTIMM	55
MCF2REF	56
MSGDD	56
OBJCONLIB	56
OUTCCSID	58
OUTEXIT	58
OUTPUTDD	59
OVLYLIB	59
PAGEDEF	61
PARMDD	63
PDEFLIB	63
PRMODE	64
PSEGLIB	65
RESEXIT	66
RESFILE	67
RESLIB	68
RESOBJDD	68
RESTYPE	69
TRACE	71
TRC	72
TRIGGERn	72
UNIQUEBNGS	74
USERLIB	74
USERPATH	75

Chapter 4. Enhanced Indexing Parameters.	77
Parameter Values	77
FIELDn	78
GROUPMAXPAGES	84
INDEXn	84
TRIGGERn	87
USERMASK	89

Chapter 5. Examples of Using ACIF	91
Examples of Using ACIF Processing Parameters	91
Transforming Line Data or XML Data into a MO:DCA-P Document	91
Retrieving Resources	92
Specifying AFP Coded Fonts	92
Specifying TrueType and OpenType Fonts	93
Identifying the Location of Resource Libraries	94
Drawing Graphics with Record Format Page Definitions	95
Example of Using ACIF to View and Index Documents	95
Examining the Input File	97
Specifying ACIF Processing Parameters	100
Indexing Data in the Input File	104
Identifying the Locations of the Resources	106
Determining the Form Definition and the Page Definition	106
Running the ACIF Job	106
Concatenating ACIF Output Files	107
Accessing the Document File for Viewing	108
Example of Using Enhanced Indexing with ACIF	109
Enhanced Indexing Telephone Bill	109
Enhanced Indexing Parameter File	111
 Chapter 6. User Exits and Input Print File Attributes	113
User Programming Exits	113
Input Record Exit	114
Index Record Exit	118
Output Record Exit	119
Resource Exit	121
User Exit Search Order	124
AIX and Windows	124
z/OS	124
VM	124
VSE	124
Attributes of the Input Print File	125
 Chapter 7. ACIF Messages	127
Message Identifiers	127
Multiple Message Scenarios	128
General Messages	128
 Appendix A. Helpful Hints	191
Working with Control Statements That Contain Numbered Lines	191
Placing TLEs in Named Groups to Avoid Storage Problems	191
Understanding How ANSI and Machine Carriage Controls Are Used	192
Transferring Files into AIX and Windows	193
Understanding Common Methods of Transferring Files into AIX or Windows	
from Other Systems	194
Physical Media	194
PC File Transfer Program	194
FTP	195
Download for z/OS	195
Other Considerations	195
Creating Invoke Medium Map (IMM) Structured Fields	195
Indexing Considerations	196
Concatenating the Resource File and the Document File	197
Processing Inline Resources	197
Specifying the IMAGEOUT Parameter	198

Creating MO:DCA-P Object Containers	198
Understanding Error Return Code 310.	198
Appendix B. Data Stream Information	201
Tag Logical Element (TLE) Structured Field	201
Begin Resource Group (BRG) Structured Field.	202
Begin Resource (BRS) Structured Field	203
End Resource (ERS) and End Resource Group (ERG) Structured Fields	203
No Operation (NOP) Structured Field	203
Format of the Resources File	203
Appendix C. Format of the Index Object File	205
Group-Level Index Element (IEL) Structured Field	205
Page-Level Index Element (IEL) Structured Field	206
Begin Document Index (BDI) Structured Field	206
Index Element (IEL) Structured Field	206
Tag Logical Element (TLE) Structured Field	207
End Document Index (EDI) Structured Field.	207
Appendix D. Format of the Output Document File	209
Page Groups	210
Begin Document (BDT) Structured Field	211
Begin Named Group (BNG) Structured Field	211
Tag Logical Element (TLE) Structured Field	212
Begin Page (BPG) Structured Field	212
End Named Group (ENG), End Document (EDT), and End Page (EPG) Structured Fields	212
Output MO:DCA-P Data Stream	212
Composed Text Control (CTC) Structured Field	212
Map Coded Font (MCF) Format 1 Structured Field	212
Map Coded Font (MCF) Format 2 Structured Field	213
Presentation Text Data Descriptor (PTD) Format 1 Structured Field	213
Inline Resources	213
Page Definitions	213
Appendix E. Accessibility	215
Using Assistive Technologies	215
Keyboard Navigation of the User Interface	215
z/OS Documentation	215
Notices	217
Programming Interfaces	218
Trademarks.	218
Glossary	221
Bibliography	227
Print Services Facility (PSF) for z/OS	227
Infoprint Server	227
Advanced Function Presentation (AFP)	228
Data Transforms	228
Text Processing	228
Infoprint Manager	229
Content Manager OnDemand	229
i-data	230
z/OS Version 1 Release 7	230

Index	233
------------------------	------------

Figures

1.	How ACIF Fits into Advanced Function Presentation	3
2.	AFP Document with Index Tags and the Index Object File	6
3.	Example Bank Statement Input File	7
4.	ACIF Processing Parameters to Index a Bank Statement	8
5.	Using ACIF to Prepare Files for Viewing	11
6.	Using ACIF to Prepare Files for Distributed Printing	13
7.	Using ACIF to Prepare Files for Archiving and Retrieving	14
8.	AFP Workbench Viewer	15
9.	Sample z/OS JCL to Invoke ACIF.	23
10.	Sample VM/CMS Commands to Invoke ACIF	25
11.	Sample VSE JCL to Invoke ACIF	26
12.	Example of a Customer's Printed Telephone Bill	96
13.	Example of the Line Data Telephone Bill	98
14.	Example of an AIX Parameter File for ASCII Input Data	99
15.	Example of an AIX Parameter File for EBCDIC Input Data	99
16.	Example of an AIX Parameter File	101
17.	Example of a z/OS Parameter File	102
18.	Example of a VM Parameter File	103
19.	Example of a VSE Parameter File	104
I 20.	Example of the Enhanced Indexing Telephone Bill	110
I 21.	Example of an Enhanced Indexing Parameter File	111
22.	AIX or Windows Sample Input Record Exit C Language Header	115
23.	z/OS, VM, or VSE Sample Input Record Exit DSECT	115
24.	AIX or Windows Sample Index Record Exit C Language Header	118
25.	z/OS, VM, or VSE Sample Index Record Exit DSECT	118
26.	AIX or Windows Sample Output Record Exit C Language Header	120
27.	z/OS, VM, or VSE Sample Output Record Exit DSECT	120
28.	AIX or Windows Sample Resource Exit C Language Header	122
29.	z/OS, VM, or VSE Sample Resource Exit DSECT	122
30.	AIX or Windows Sample Print File Attributes C Language Header	125
31.	z/OS, VM, or VSE Sample Print File Attributes DSECT	125
32.	Example of Code Containing Group-Level Indexing.	209
33.	Example of Code Containing Group- and Page-Level Indexing	210

Tables

1.	Term Definitions	xiv
2.	Search Order for AFP Resources	19
3.	Search Order of Resource File Extensions	20
4.	ACIF Parameters and Operating Systems	31
I 5.	Hexadecimal Strings for Encoding Values	44
I 6.	ACIF Enhanced Indexing Parameters	77
7.	Font Short Names to Use with CHARS Parameter	93
8.	Output Files ACIF Creates	107

About This Publication

This publication describes Advanced Function Presentation™ Conversion and Indexing Facility (ACIF), which is available for use with Print Services Facility™ (PSF) and Infoprint® Manager. PSF uses ACIF in the z/OS®, z/OS.e, VM, and VSE environments, while Infoprint Manager uses ACIF in the AIX® and Windows® environments. In this publication, the term “z/OS” refers to z/OS and z/OS.e, unless otherwise specified.

Note: ACIF is also used by Infoprint Server on OS/400® and i5/OS™; however, this publication only describes using ACIF with PSF and Infoprint Manager. For information about using ACIF with Infoprint Server on OS/400 and i5/OS, refer to *Infoprint Server for iSeries: User's Guide*, G544-5775.

This publication assumes that you are familiar with Advanced Function Presentation (AFP™) concepts as well as the parameters that you specify when printing with PSF, Infoprint Manager for AIX, and Infoprint Manager for Windows. 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 : *IBM Infoprint Manager for AIX: Getting Started*, *IBM Infoprint Manager for Windows: Getting Started*, *IBM Infoprint Manager: Reference*, or the PSF user's guide for your operating system, as listed in “Bibliography” on page 227.

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

Who Should Read This Publication

This publication contains information that application programmers can use to develop ACIF applications that:

- Convert line data and XML data 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.

Note: This publication provides ACIF messages that contain instructions for the system programmers responsible for maintaining the operating system and the PSF or Infoprint Manager program running on it. You might need to show these messages to your system programmer for assistance from time to time.

How This Publication Is Organized

This publication contains information pertaining to ACIF support for AIX, Windows, z/OS, z/OS.e, VM, and VSE operating environments used by Infoprint Manager and PSF:

- Chapter 1, “Understanding ACIF” presents an overview of tasks you can do with the ACIF product, describes several related products, and describes system considerations for using ACIF.

- Chapter 2, “Using ACIF” provides sample code for invoking ACIF.
- Chapter 3, “ACIF Parameters” describes the parameters used for ACIF processing, including syntax rules and parameter values.
- Chapter 4, “Enhanced Indexing Parameters” describes the parameters used for ACIF enhanced indexing with PSF 4.1.0 for z/OS.
- Chapter 5, “Examples of Using ACIF” shows examples of an ACIF application.
- Chapter 6, “User Exits and Input Print File Attributes” describes the exits available for customizing ACIF.
- Chapter 7, “ACIF Messages” provides the ACIF messages, with suggestions for responding to the errors.
- The appendixes contain more information about ACIF:
 - 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.
 - Appendix E, “Accessibility” describes the accessibility features available in z/OS.

A notices section, glossary, bibliography, and index are included. The bibliography lists the publications containing additional information about AFP, PSF, Infoprint Manager, and related products.

What Terms Are Used in This Publication

The terms document, file, and library are used throughout this publication. In all systems, document is a file that contains AFP structured fields in Mixed Object Document Content Architecture™ for Presentation (MO:DCA-P) format. The terms *file* and *library* have different meanings in different operating systems. Table 1 lists the meanings of *file* and *library* in AIX, Windows, z/OS, VM, and VSE systems.

Table 1. Term Definitions

	File	Library
AIX	A collection of related data	A directory in which AFP resources are stored
Windows	A collection of related data	• A directory • A list of files stored on a disk or diskette
z/OS	• A sequential data set • A member of a partitioned data set • The name of a DD card	• A partitioned data set • A series of concatenated data sets
VM	A CMS file (filename filetype filemode)	A collection of CMS files, generally with the same file type
VSE	A sequential (SAM) file	A library.sublibrary

Understanding the Notational Conventions Used in This Publication

This publication uses consistent conventions for the following:

- Highlighting
- Syntax notation

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.

Syntax Notation

This publication uses the following syntax notation:

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

CPGID=*codepageid*

means that you enter **CPGID=** as shown and then replace the variable *codepageid* 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:

Vertical 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 and Windows, 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 **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.

Related Information

Publications that are referred to in this document or that contain additional information about AFP, the z/OS operating system, PSF, Infoprint Manager, and related products are listed in the “Bibliography” on page 227.

For additional information about z/OS, PSF, and Infoprint Manager, refer to these Web pages:

<http://www.ibm.com/servers/eserver/zseries/zos/>

http://www.printers.ibm.com/internet/wwwsites.nsf/vwwebpublished/psfhome_z_ww

http://www.printers.ibm.com/internet/wwwsites.nsf/vwwebpublished/zseriessoftware_ww

<http://www.ibm.com/printers>

To obtain the latest documentation updates for z/OS base elements and optional features that result from DOC APARs and PTFs, refer to this DOC APARs and ++HOLD DOC Web page:

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS

To obtain the latest documentation updates for PSF for z/OS, refer to these members in SYS1.SAMPLIB:

Member	Publication
APSGCUS4	<i>PSF for z/OS: Customization</i> , S544-5622
APSGDGN4	<i>PSF for z/OS: Diagnosis</i> , G544-5623
APSGDLG4	<i>PSF for z/OS: Download for z/OS</i> , S544-5624
APSGMAC4	<i>PSF for z/OS: Messages and Codes</i> , G544-5627
APSGSECU	<i>PSF for z/OS: Security Guide</i> , S544-3291
APSGUSR4	<i>PSF for z/OS: User's Guide</i> , S544-5630

Using LookAt to View Message Explanations

LookAt is an online facility that lets you view explanations for most IBM® messages and some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation. To find a message explanation in LookAt, you enter the message identifier (for example, IAT1836 or IAT*).

You can access LookAt from these locations:

Internet

You can access the LookAt Web site at:

<http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/>

z/OS TSO/E host system

You can install code on your z/OS or z/OS.e system to access LookAt from a TSO/E command line, such as TSO/E prompt, ISPF, or z/OS UNIX[®] System Services.

Microsoft[®] Windows workstation

You can install code on your workstation to access LookAt from a Microsoft Windows command prompt (also known as the DOS command line).

Wireless handheld device

You can link to the LookAt Mobile Edition from the LookAt Web site to access LookAt from a handheld device that has wireless access and an Internet browser, such as Internet Explorer for Pocket PCs; Blazer or Eudora for Palm OS; or Opera for Linux[™] handheld devices.

To install LookAt on your TSO/E host system or Microsoft Windows workstation, you can obtain the code from:

- A CD-ROM in the *z/OS Collection*
- The *z/OS and Software Products DVD Collection*
- The LookAt Web site's **Download** link. More information is available in the LOOKAT.ME files available during the download process.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents that contain the message.

Summary of Changes

Summary of Changes for *AFP Conversion and Indexing Facility User's Guide*, S550-0436-00

This publication contains additions and changes to information previously presented in *AFP Conversion and Indexing Facility: User's Guide*, S544-5285-04. The technical additions and changes are marked with a revision bar (|) in the left margin.

These general changes have been made throughout the publication:

- This publication has been enabled for these z/OS Library Center advanced searches: concepts, references, tasks, commands, and examples.
- The term "platform" has been changed to "operating system" or "system" as appropriate.
- References to "OS/390®" have been removed.

This information has been deleted:

- The AFP Workbench Viewer attribute name size is no longer limited to 64 bytes, so that limitation has been removed. See "AFP Workbench Viewer" on page 14.
- In the AIX and Windows examples for the parameters "FONTPATH" on page 46 and "USERPATH" on page 75, the single quotes have been removed from around the colon (:).
- The restriction that the size of the AIX and Windows input file is limited to 4 gigabytes (GB) has been removed from the parameter "INPUTDD" on page 54.

The following information is new or updated:

- "System Prerequisites" on page 17 has been updated.
- Table 3 on page 20 has been updated with resources and file extensions.
- Several updates have been made to the parameter "EXTENSIONS" on page 39:
 - The description of SPCMPRS has been updated.
 - This note has been added to RESORDER: "When RESORDER is specified, TrueType and OpenType fonts are not processed."
 - MVSICNV value has been added.
- The parameter "FILEFORMAT" on page 43 has been updated with additional NEWLINE values.
- The INPCCSID and OUTCCSID parameters have been added. See "Invoking ACIF" on page 21, "INPCCSID" on page 53, and "OUTCCSID" on page 58.
- A note has been added to the parameter "RESTYPE" on page 69 indicating that when EXTENSIONS=RESORDER is specified with RESTYPE=FONT, TrueType and OpenType fonts are not processed.
- Chapter 4, "Enhanced Indexing Parameters," on page 77 has been added to describe the parameters required for enhanced indexing.
- In Figure 17 on page 102, the RESLIB line has been updated from SPACE=(12288,(150,15),RLSE),UNIT=SYSDA, to SPACE=(32760,(150,150),RLSE),UNIT=SYSDA, and the SYSPRINT line has been updated from DCB=(BLKSIZE=141,RECFM=VB,DSORG=PS) to DCB=(LRECL=137,BLKSIZE=1374,RECFM=VBA,DSORG=PS).
- "Example of Using Enhanced Indexing with ACIF" on page 109 has been added with a telephone bill and z/OS enhanced indexing parameter file.

- AIX and Windows sample C language headers for the input record exit and the resource exit have been updated. See Figure 22 on page 115 and Figure 28 on page 122.
- “Using ACIF User Input Record Exits in AIX and Windows” on page 116 has been updated with information for converting encoded data from one character set to another (for example, ASCII to EBCDIC with **uconv**).
- Chapter 7, “ACIF Messages,” on page 127 has been updated with these new or updated messages:
 - APK104S on page 128
 - APK264S on page 144
 - APK392I on page 160
 - APK410S on page 162
 - APK424I on page 165
 - APK430I on page 166
 - APK452S on page 168
 - APK532S on page 172
 - APK2080I on page 188
 - APK3506I on page 189
 - APK3507I on page 189
- “Transferring Files into AIX and Windows” on page 193 has been updated with FILEFORMAT parameter information for variable-length files.
- “Understanding Error Return Code 310” on page 198 has been updated with another reason for receiving the return code.
- ACIF now supports page-level indexing if you are using enhanced ACIF indexing. See “Tag Logical Element (TLE) Structured Field” on page 201, Appendix C, “Format of the Index Object File,” on page 205, “Page-Level Index Element (IEL) Structured Field” on page 206, and Appendix D, “Format of the Output Document File,” on page 209.
- Terms in the glossary have been updated so they are consistent with term definitions used across IBM.

Chapter 1. Understanding ACIF

AFP Conversion and Indexing Facility (ACIF) is a batch application development utility that lets you create documents by formatting line data (record format and traditional), XML data, MO:DCA-P print files, and unformatted ASCII files and then print them with IBM Infoprint Manager or IBM Print Services Facility (PSF). ACIF also provides indexing and resource retrieval capabilities that let you view, distribute, archive, and retrieve document files across systems and operating systems.

Infoprint Manager uses ACIF in the AIX and Windows environments. ACIF is also used in the z/OS, z/OS.e, VM, and VSE environments. Hereafter, the term “z/OS” refers to z/OS and z/OS.e, unless otherwise specified.

This chapter gives an overview of ACIF, explains the functions that ACIF can perform, describes different scenarios for processing your files, describes the IBM products you can use with ACIF, and lists the system limitations and prerequisites you must consider for ACIF.

Overview

With ACIF you can:

- Convert line data, XML data, or mixed data into MO:DCA-P data, which is an architected, device-independent data stream used for interchanging documents between different systems.
- 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.

ACIF accepts data from your application in these formats:

- AFP data
- MO:DCA-P data
- Record format or traditional line data
- Mixed-mode data
- XML data
- Unformatted ASCII data (AIX and Windows only)

ACIF can process application print data and AFP resources to produce these AFP files:

- Document file
- Resource file
- Index object file

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

- Use PSF or Infoprint Manager to print the AFP document file. If you have specified resources in the AFP document file, PSF or Infoprint Manager

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 AFP Workbench Viewer application to view the AFP document file. AFP 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 Content Manager OnDemand. 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.

Figure 1 on page 3 shows a high-level overview of how ACIF fits into an installation's AFP process for creating, indexing, viewing, and printing documents. This figure shows the resources and the text data, which can be provided and used by various AFP and AFP-compatible products, that can feed into ACIF for processing. The files that ACIF produces can then be sent to a customer-supplied archival and retrieval system, to the spool, or to the AFP Workbench Viewer for viewing.

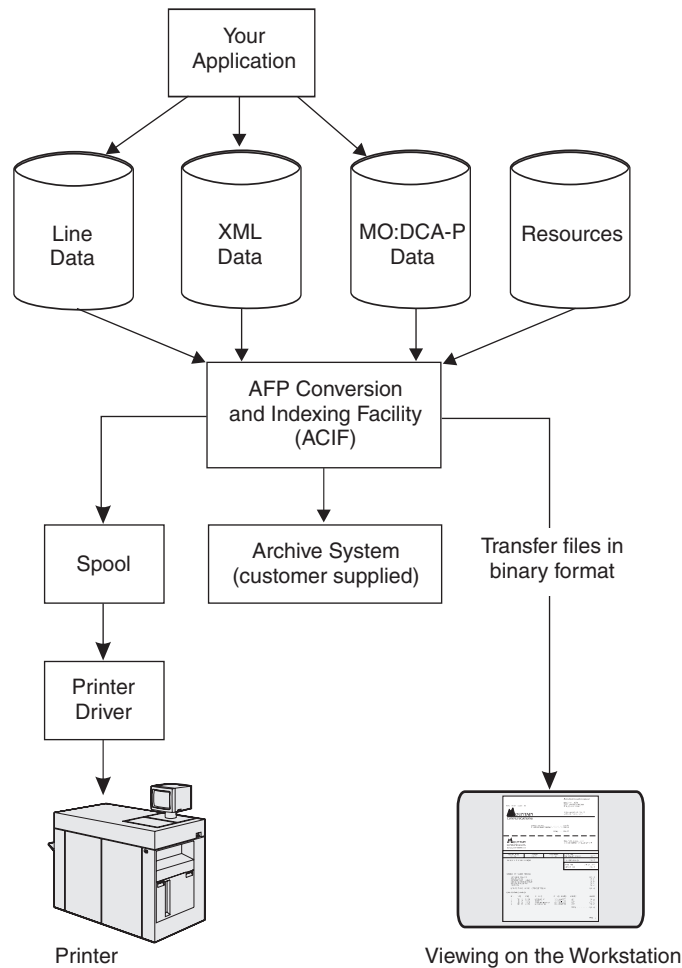


Figure 1. How ACIF Fits into Advanced Function Presentation

ACIF Functions

You can use ACIF to perform these functions:

- Convert data streams
- Index documents
- Retrieve resources

Converting Data Streams

ACIF processes these input data streams to create a MO:DCA-P document:

- AFP data
- MO:DCA-P data
- Record format or traditional line data
- Mixed-mode data
- XML data
- Unformatted ASCII (AIX and Windows 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 these 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. Fonts are either Font Object Content Architecture (FOCA) fonts or TrueType and OpenType fonts, which are not defined by FOCA. For more information about 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 these 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 most of 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 might contain multiple Begin Document (BDT) and End Document (EDT) structured fields, the ACIF output normally contains only one BDT/EDT structured-field pair. To pass all of the BDT/EDT pairs to the output data stream, the INDEXOBJ=BDTLY parameter is specified. See “Output MO:DCA-P Data Stream” on page 212 for information about the changes ACIF makes to support MO:DCA-P output format.

For more information about the MO:DCA-P data stream, refer to *Mixed Object Document Content Architecture Reference*.

Line Data

Line data is application data that is prepared for printing without any data placement or presentation information. Line data can be either traditional line data or record format line data. Traditional line data is data prepared for printing on a line printer. Record format line data is a form of line data where each record is preceded by a variable length identifier.

ACIF formats line data into pages by using a page definition (PAGEDEF) resource, in the same manner as 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 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*.

XML Data

Data that has been identified using Extensible Markup Language (XML) standards from the World Wide Web Consortium is called XML data. XML does not describe data placement or presentation information. For printing on page printers, a page

definition is required to provide the data placement and presentation information. The XML data processed by ACIF can be encoded in EBCDIC, ASCII, UTF-8 or UTF-16. For more information about XML data, refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* and *Extensible Markup Language (XML) 1.0 Specification*, at the World Wide Web Consortium <http://www.w3.org>.

Unformatted ASCII Data

Unformatted ASCII data is generated in AIX and Windows environments and is a type of line data that does not contain escape sequences. Unformatted ASCII data can have these characteristics:

- No embedded control characters except for newlines
- American National Standards Institute (ANSI) carriage control characters, table reference characters, or both
- Carriage returns and form feed controls

Unformatted ASCII data is formatted by ACIF using a page definition resource. ASCII data containing control characters (or escape sequences) for the IBM Proprinter and Quietwriter® does not need to be formatted by ACIF. Unformatted ASCII data can also be submitted for printing with Infoprint Manager 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 with 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 (PPFA) to create your own page definitions. PPFA is a separately orderable feature of Infoprint Manager for AIX and Infoprint Manager for Windows. For information about how to create page definitions using PPFA, refer to *Page Printer Formatting Aid: 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 a large print file into smaller, uniquely identifiable units, called *groups*, as defined by the MO:DCA-named group structured fields. For example, you can use ACIF to divide a large bank-statement application into individual groups by inserting structured fields that define group boundaries into the file. A group is a named collection of sequential pages, which, in this example, consists of the pages describing a single customer's account. For example, a bank-statement application probably produces a large printout consisting of thousands of individual customer statements. You can think of each of these statements as smaller, separate units, each uniquely identifying an account number, date, Social Security number, or other attributes.

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

- 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 AFP 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 2 shows the relationship between the group-level tags and the entries in the index object file.

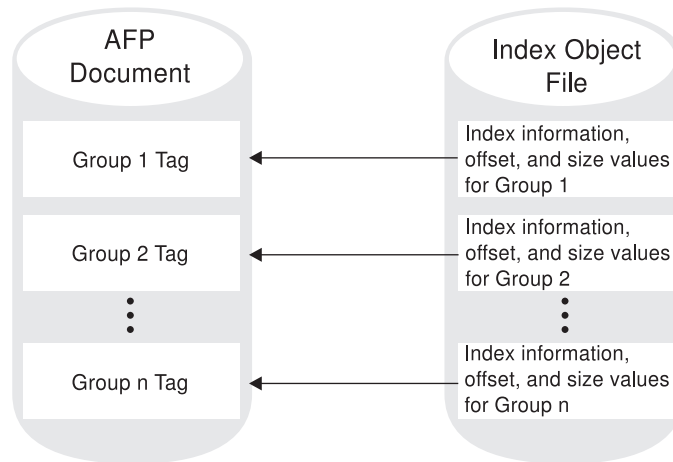


Figure 2. AFP Document with Index Tags and the Index Object File

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

- Line data, XML data, or mixed-mode data
- Unformatted ASCII data
- AFP data produced by the AFP Application Programming Interface (API), 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 these ways for you to generate the indexing tags placed in the print file:

- Use 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*.
- Use 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*.

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 might not contain the data you would like to use for an indexing tag. In these cases, you can

1. With ACIF, you can generate group-level tags and also page-level tags with enhanced indexing; with Document Composition Facility (DCF) and AFP Toolbox, you can generate both group-level tags and page-level tags. See “IBM Products Used with ACIF” on page 14 for more information about these products.

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.

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 might 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.

Example of Indexing with Data Values: This example shows how to use the ACIF parameters described in Chapter 3, "ACIF Parameters," on page 29. Figure 3 shows the print file for a typical bank statement.

```
1ACCOUNT NUMBER: 445-66-3821-5      PAGE 1
  CUSTOMER NAME: HENRY WALES
  DATE: 09/30/99
  CHECK# 001 - 455.00
  CHECK# 002 - 337.85
  ...
1ACCOUNT NUMBER: 333-56-4378-5      PAGE 1
  CUSTOMER NAME: KATHERINE CHARLES
  DATE: 09/30/99
  CHECK# 221 - 5.00
  CHECK# 222 - 1567.35
  ...
```

Figure 3. Example Bank Statement Input File

In Figure 3, the print file contains bank statements dated September 30, 1999 (09/30/99). Each statement has the same general format, although statements might 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 can 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 4.

```
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 4. ACIF Processing Parameters to Index a Bank Statement

The information in Figure 4 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 can 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 **FIELDn** definitions are relative to this anchor point.

In Figure 4, 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 before 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.

Indexing Limitations

For a line data or XML 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 "CCTYPE" on page 33 for a description of the parameter.)

Retrieving Resources

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

When you archive a document, ACIF also lets you archive the retrieved resources (such as fonts and page segments) 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, BCOCA, GOCA, IOCA** 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 z/OS and lets the resource file be referenced as a user library (USERLIB) when printing with PSF.
 - An AFP data stream resource group. The AFP resource-group format is useful when you are routing print output to remote AFP systems (for example, Infoprint Manager for Windows) or when storing a print file in an archive system (for example, Content Manager OnDemand).
- 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

Examples of specifying ACIF processing parameters for resource retrieval can be found in Chapter 5, "Examples of Using ACIF," on page 91.

Scenarios for Processing ACIF Files

ACIF can process your files for:

- Viewing with AFP Workbench Viewer
- Printing locally and on other systems
- Archiving and retrieving selectively

The following sections show scenarios for preparing files for viewing, printing, and archiving.

Preparing Files for Viewing

Figure 5 on page 11 shows the steps you take to prepare files for viewing with the AFP 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 or Infoprint Manager 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 AFP Workbench Viewer is installed, you concatenate the resource file (3c) to the AFP document file. The order of concatenation must be shown as in Figure 5 on page 11, with the document file concatenated last.
5. Transfer (5) the needed files in binary format to the workstation.
6. Using the AFP Workbench Viewer, view (6) your indexed document. You can also print the document from the AFP Workbench Viewer.

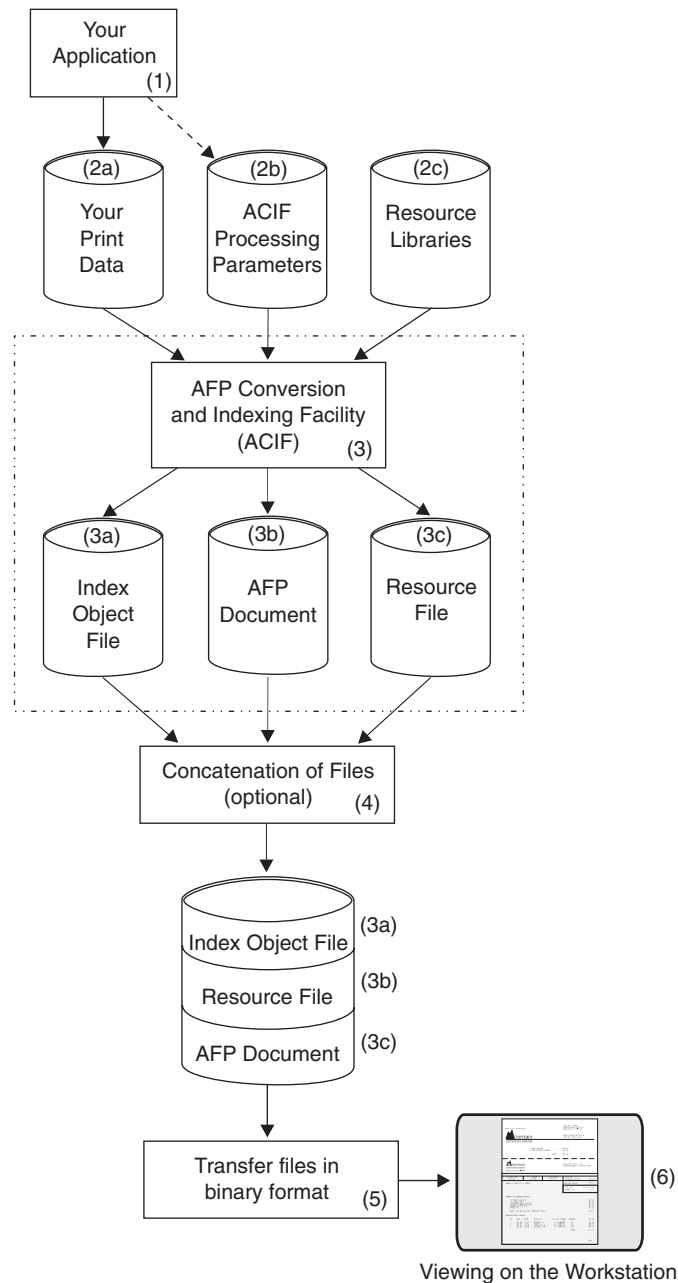


Figure 5. Using ACIF to Prepare Files for Viewing

Preparing Files for Printing

Figure 6 on page 13 shows the steps you take to prepare your files for printing:

1. Run ACIF (1), specifying that it create the AFP document file (1a) and the resource file (1b).

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

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

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

3. Submit (3) your MO:DCA-P print job to PSF or Infoprint Manager.

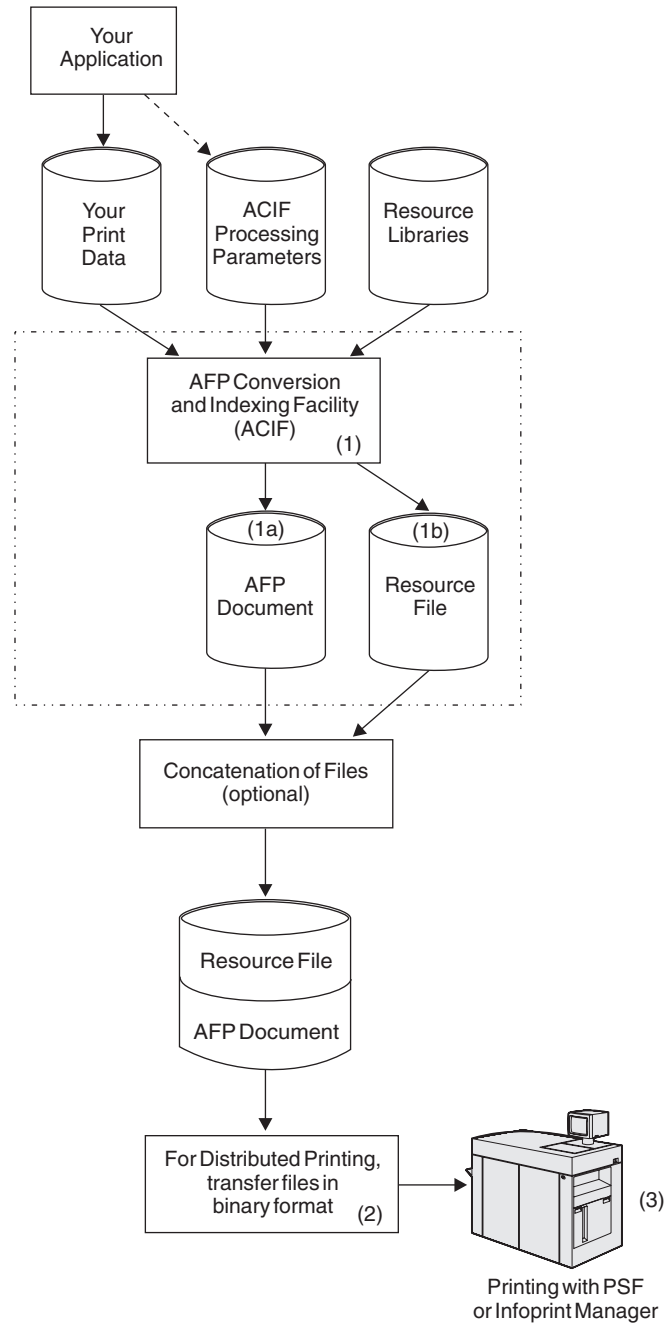


Figure 6. Using ACIF to Prepare Files for Distributed Printing

Preparing Files for Archiving and Retrieval

Figure 7 on page 14 shows the steps you can use to archive your files:

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

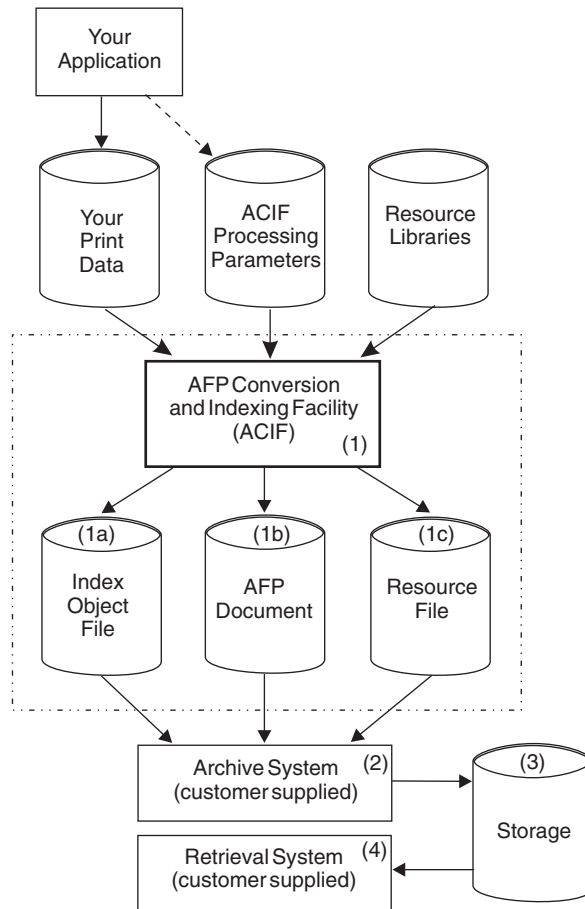


Figure 7. Using ACIF to Prepare Files for Archiving and Retrieving

IBM Products Used with ACIF

Although ACIF is a stand-alone utility, it has been designed for use with these IBM products:

- AFP Workbench Viewer
- AFP Toolbox
- Document Composition Facility (DCF)

AFP Workbench Viewer

Figure 8 on page 15 shows how AFP Workbench Viewer can display documents on a workstation running Microsoft Windows operating systems. These documents can contain an index object file and a resource group.

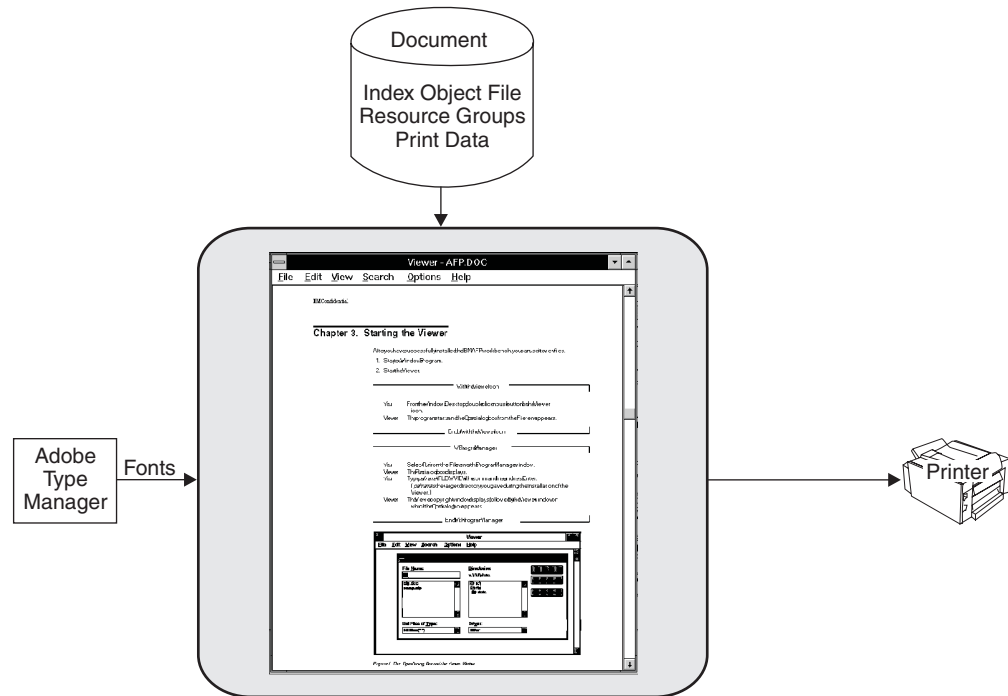


Figure 8. AFP Workbench Viewer

AFP Workbench Viewer uses Adobe Type 1 or TrueType and OpenType outline fonts when displaying documents. If the document references an AFP font for which no Type 1 font is available at the workstation, AFP Workbench Viewer can substitute an outline font for the requested font. AFP Workbench Viewer matches the requested point size and attempts to match the typeface as closely as possible. Font definition files are available with AFP Workbench Viewer to let you define which Type 1 fonts are to be substituted for your FOCA fonts.

Because AFP Workbench Viewer uses font substitution for AFP font resources instead of retrieving fonts from a resource file, you do not need to specify the **RESTYPE=FONT** or **RESTYPE=ALL** ACIF parameter when preparing a document to use with AFP Workbench Viewer. If you do not want to use font substitution, you should use TrueType and OpenType fonts when creating the AFP document, keeping in mind that not all Intelligent Printer Data Stream™ (IPDS™) printers support TrueType and OpenType fonts.

When using ACIF to index a file for viewing, specify **INDEXOBJ=ALL**. This setting provides AFP 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 AFP Workbench Viewer. (It is important that the document file comes last, at the end of the resulting concatenated file; otherwise, an error occurs.)

AFP Workbench Viewer supports a subset of MO:DCA-P data and might not display everything that PSF or Infoprint Manager can print. For example, AFP Workbench Viewer for Windows recognizes but ignores bar code (BCOCA) objects, whereas PSF and Infoprint Manager can print these objects.

AFP Toolbox

AFP Toolbox (Program Number 5655-A25) assists application programmers in formatting printed output. Without requiring knowledge of the AFP data stream, AFP Toolbox provides access to sophisticated AFP functions through a callable C, C++, or COBOL interface. With AFP Toolbox you can:

- Combine variable data with electronic forms, electronic signatures, and images.
- Define variable length paragraphs.
- Draw fixed or variable depth and width boxes.
- Generate bar code objects.
- Draw horizontal and vertical fixed or variable length lines.
- Include indexing tags for use in efficient viewing, archival, and retrieval.
- Accent printed output with color and shading.
- Dynamically control fonts, including user-defined fonts.
- Precisely position and align text anywhere on a page using a wide variety of fonts.
- Create graphical data objects such as pie charts and bar charts.

AFP Toolbox is available on the z/OS operating system.

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 z/OS, 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*. Note that DCF is not applicable to the AIX or Windows environment.

What Are the System Considerations for ACIF?

You must consider these when using ACIF:

- System limitations
- System prerequisites

System Limitations

You can use ACIF to build applications for PSF for z/OS, PSF/VM, or PSF/VSE; however, each PSF product provides different levels of MO:DCA-P support. For specific information about the MO:DCA-P function supported, refer to the documentation for the PSF product you are using.

PSF/VSE limitations include:

- 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 system.
- To use PSF/VSE 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, Windows, z/OS, VM, and VSE environments.

AIX Prerequisites

To see the AIX software requirements for using ACIF, refer to *IBM Infoprint Manager for AIX: Introduction and Planning Guide*.

Windows Prerequisites

To see the Windows software requirements for using ACIF, refer to *IBM Infoprint Manager for Windows: Introduction and Planning Guide*.

z/OS Prerequisites

One of these z/OS software products is required to use ACIF:

- z/OS 1.4.0 or higher
- z/OS.e 1.4.0 or higher

PSF 4.1.0 for z/OS to use the enhanced indexing parameters in ACIF

VM Prerequisites

One of these VM software products is required to use ACIF:

- VM/SP 5 or higher
- VM/SP HPO 5 or higher
- VM/XA 1.2.1 or higher
- VM/ESA[®] 1.1.0 or higher

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

VSE Prerequisites

One of these VSE software products is required to use ACIF:

- VSE/SP 4.1.2 or higher
- VSE/ESA[™] 1.1.0 or higher

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

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

Chapter 2. Using ACIF

This chapter describes how to invoke ACIF in AIX, Windows, z/OS, VM, and VSE environments.

Using ACIF in AIX and Windows

In AIX and Windows, ACIF transforms line data, XML 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:

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

Selecting Resources

Table 2 lists the order ACIF searches for AFP resources in AIX and Windows.

Table 2. Search Order for AFP Resources

Search Order		Location
Windows	AIX	
1	1	Paths specified by the USERLIB parameter (AFP resources, except TrueType and OpenType fonts) or the USERPATH parameter (TrueType and OpenType fonts only; see Note)
2	2	Paths specified by the FDEFLIB , FONTLIB , FONTPATH (see Note), 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/ipfonts In Infoprint Manager for AIX, AFP FOCA fonts are included in Infoprint Fonts. Refer to <i>IBM Infoprint Fonts: Font Summary</i> for more information.
	7	The directory /usr/lpp/afpfonts In Infoprint Manager for AIX, AFP FOCA fonts are included in the AFP Font Collection. Refer to <i>IBM AFP Fonts: Font Summary for AFP Font Collection</i> for more information.
	8	The directory /usr/lpp/psf/fontlib
3		Windows registry used to locate: 1. Default RESLIB (<i>\install_directory\reslib</i>) 2. Default FONTLIB (<i>\install_directory\fontlib</i>) 3. AFP Font Collection or Infoprint Fonts

Note: The paths specified by **USERPATH** and **FONTPATH** must be the same paths where the Font Installer for AFP Systems installed TrueType and OpenType fonts. Font Installer for AFP Systems is an optional feature of Infoprint Fonts for Multiplatforms (Program Number 5648–E77). For more information about TrueType and OpenType Fonts, refer to *Using OpenType Fonts in an AFP System*.

Note: AFP resource files that ACIF processes in all environments must contain a 'X'5A' carriage control character at the start of each structured field.

When ACIF 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. Table 3 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.

Table 3. Search Order of Resource File Extensions

Type of Resource	File Extensions Search Order (see Note)
AFP font objects: Coded fonts Code pages Outline fonts 240-pel resolution fonts 300-pel resolution fonts	1. No file extension 2. 240 3. 300 4. FONT300 5. FONT3820 6. FONT38PP 7. CDP 8. CFT 9. OLN 10. FONTOLN 11. FIL
BCOCA (bar code) objects	No File extension
Color mapping table	1. No file extension 2. SETUP 3. SET
Fonts, TrueType and OpenType	Entire file name, including extension, comes from the Resource Access Table of the Font Installer for AFP Systems.
Form definitions	1. No file extension 2. FDEF3820 3. FDEF38PP 4. FDE 5. FIL
GOCA (graphics) objects	No File extension
IOCA (IO image) objects	No File extension
MO:DCA™ objects	1. No File extension 2. OBJ 3. OBJECT
Overlays	1. No file extension 2. OVLY3820 3. OVLY38PP 4. OVL 5. OLY 6. OVR
Page definitions	1. No file extension 2. PDEF3820 3. PDEF38PP 4. PDE

Table 3. Search Order of Resource File Extensions (continued)

Type of Resource	File Extensions Search Order (see Note)
Page segments	1. No file extension 2. PSEG3820 3. PSEG38PP 4. PSG 5. PSE
Setup data	1. No file extension 2. SETUP 3. SET 4. COMSETUP
Note: All file extensions must be in uppercase format.	

Invoking ACIF

The **acif** command and the **line2afp** and **pdpr** commands of Infoprint Manager for AIX or Infoprint Manager for Windows are used to invoke ACIF.

To use ACIF to prepare line data, XML data, mixed-mode data, or unformatted ASCII files for printing with Infoprint Manager for AIX or Infoprint Manager for Windows, you can *automatically* invoke the **acif** command at print submission time by doing one of these:

- Use the **-odatatype=line** flag and keyword-value pair with one of the AIX print commands (**enq**, **lp**, or **qprt**).
- Use the **psfin** command to specify a job script with a setting of **-JsFiletype=line**.

The **line2afp** command is the same as the **acif** command and uses the **acif** command conversion parameters to produce output for printing. 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 can transform, print, view, archive, and retrieve files as in ACIF.

ACIF and the **line2afp** command use these parameters for *conversion*: **CC**, **CCTYPE**, **CHARS**, **EXTENSIONS**, **FDEFLIB**, **FILEFORMAT**, **FONTLIB**, **FORMDEF**, **IMAGEOUT**, **INPCCSID**, **INPEXIT**, **INPUTDD**, **INSERTIMM**, **MCF2REF**, **MSGDD**, **OUTCCSID**, **OUTEXIT**, **OUTPUTDD**, **OVLYLIB**, **PAGEDEF**, **PARMDD**, **PDEFLIB**, **PRMODE**, **PSEGLIB**, **RESEXIT**, **RESLIB**, **TRC**, and **USERLIB**.

ACIF uses these parameters for *resource retrieval*: **COMSETUP**, **FDEFLIB**, **FONTLIB**, **FONTPATH**, **FORMDEF**, **OBJCONLIB**, **OVLYLIB**, **PSEGLIB**, **RESEXIT**, **RESOBJDD**, **RESTYPE**, and **USERPATH**.

ACIF uses these parameters for its *indexing* functions: **CPGID**, **DCFPAGENAMES**, **FIELD_n**, **GROUPNAME**, **INDEX_n**, **INDEXDD**, **INDEXOBJ**, **INDEXSTARTBY**, **INDEXEXIT**, and **TRIGGER_n**.

The **pdpr** command calls **line2afp** to invoke ACIF. Parameters not allowed on the **pdpr** command can be passed to ACIF with the **-x "other-transform-options"** attribute.

The **line2afp** command and the **pdpr** command are described in *IBM Infoprint Manager: Reference*.

Files Provided with Infoprint Manager

The executable program (acif command)

- AIX: `/usr/lpp/psf/bin/acif`

Note: AIX maps the `line2afp` command to the `acif` command.

- Windows: `\install_directory\bin\acif.exe`

Windows also ships the executable, `line2afp.exe`, which is identical to `acif.exe`.

Sample ACIF user exits

- AIX: `/usr/lpp/psf/acif/apkinp.c`, `apkind.c`, `apkres.c`, `apkout.c`, `apka2e.c`, `asciinp.c`, `asciinpe.c`
- Windows: `\install_directory\exits\acif\apkinp.c`, `apkind.c`, `apkres.c`, `apkout.c`, `apka2e.c`, `asciinp.c`, `asciinpe.c`

Sample user exit executables

- AIX: `/usr/lpp/psf/bin/apka2e`, `apkinp`, `apkind`, `apkres`, `apkout`, `asciinp`, `asciinpe`
- Windows: use *.dsw files to build

Build rules for ACIF user exits: apkinp, apkind, apkres, apkout, apka2e, asciinp, asciinpe

- AIX: `/usr/lpp/psf/bin/Makefile`
- Windows: use *.dsw files to build

C language header file for ACIF user exits

- AIX: `/usr/lpp/psf/acif/apkexits.h`
- Windows: `\install_directory\exits\acif\apkexits.h`

Note: Infoprint Manager for AIX must be installed if you want to use the examples documented in this publication that contain path names indicating `/psf/`; for example:

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

Infoprint Manager for Windows must be installed if you want to use the examples documented in this publication that contain path names indicating `\exits\acif\`; for example:

```
inpexit=\install_directory\exits\acif\asciinpe.dll
```

NLS Messages

ACIF messages on AIX can be written in any one of these languages: Simplified Chinese, Traditional Chinese, English, French, French-Canadian, German, or Japanese. ACIF messages on Windows can be written in any one of these languages: French, German, Italian, Spanish, or Japanese.

In AIX, consult the description of the `NLSPATH` and `LANG` environment variables for information about setting these variables in an appropriate manner.

Suggested Reading

Refer to these publications for more information about printing with Infoprint Manager, form definitions, and page definitions:

- *IBM Infoprint Manager: Reference* for information about transforming line data for printing with Infoprint Manager and for information about form definitions and page definitions supplied with Infoprint Manager for AIX and Infoprint Manager for Windows.
- *IBM Infoprint Manager for AIX: Procedures* and *IBM Infoprint Manager for AIX: Getting Started*.
- *IBM Infoprint Manager for Windows: Procedures* and *IBM Infoprint Manager for Windows: Getting Started*.
- *Page Printer Formatting Aid: User's Guide* for information about how to create your own form definitions and page definitions.

Using ACIF in z/OS

Figure 9 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=print file
//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 9. Sample z/OS JCL to Invoke ACIF

z/OS JCL Statements for Invoking ACIF

The JCL statements in Figure 9 are explained in this section. For more information about programming JCL, refer to *z/OS MVS JCL Reference*.

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

Specifies 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 DD name for the data set containing the ACIF processing

parameters. If **PARMDD** is not specified, ACIF uses **SYSIN** as the default DD name and stops processing if **SYSIN** is not defined.

MSGDD

Defines the DD name 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 DD name and stops processing if **SYSPRINT** is not defined.

Although Figure 9 on page 23 shows a specified **REGION** size of 3 MB, this value can vary, depending on the complexity of the input data and the conversion and indexing options requested.

INPUT

Specifies the DD statement that defines the print data set to be processed by ACIF. In Figure 9 on page 23, this is the same data set as defined in the **PRINTOUT** DD statement.

OUTPUT

Specifies the DD statement that defines the name of the print data set that ACIF creates as a result of processing the application's print data set. Figure 9 on page 23 shows the DCB requirements.

RESOBJ

Specifies the DD statement that defines the name of the resource data set that ACIF creates as a result of processing the print data set. The RESOBJ file must be allocated with variable blocked records. This statement is not required if **RESTYPE=NONE** is specified in the processing parameter data set. See "RESTYPE" on page 69 for more information about the **RESTYPE** parameter.

INDEX

Specifies the DD statement that 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 is created.
- If **INDEXOBJ=NONE** is specified in the processing parameter data set and no indexing keywords are specified (**FIELD**, **INDEX**, or **TRIGGER**).

SYSPRINT

Specifies the DD statement that defines the system output data set. 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,DSORG=PS**.

SYSIN

Specifies the DD statement that defines the data set containing the ACIF processing parameters. This is the default DD name 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 DD names.

Using ACIF in VM

Figure 10 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
APKACIF (PARMDD ddname MSGDD ddname
```

Figure 10. Sample VM/CMS Commands to Invoke ACIF

VM/CMS Commands for Invoking ACIF

The CMS commands in Figure 10 are explained as follows.

USERAPPL

Invokes the application that produces the actual print output.

INPUT

Defines the DD name for the print file to be processed by ACIF. In Figure 10, this is the same print file that is created by **USERAPPL**.

OUTPUT

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

RESOBJ

Defines the DD name 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 DD name 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 is created.
- If **INDEXOBJ=NONE** is specified in the processing parameter data set and no indexing keywords are specified (**FIELD**, **INDEX**, or **TRIGGER**).

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 DD name for the file containing the ACIF processing parameters. If **PARMDD** is not specified, ACIF uses **SYSIN** as the default DD name and stops processing if **SYSIN** is not defined.

MSGDD

Defines the DD name 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 DD name and stops 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, $(137*10)+4 = 1374$).

Note: The ACIF naming convention for the DD name is the same as that used in z/OS.

ACIF requires about 3 MB 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 VSE

Figure 11 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 11. Sample VSE JCL to Invoke ACIF

VSE JCL Statements for Invoking ACIF

The statements in Figure 11 are explained in this section. 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,... // EXTENT ,volser // LIBDEF PHASE,SEARCH=...

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. The control statement is required or processing stops.

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

Defines the file to be processed by ACIF. In Figure 11 on page 26, 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 59 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. See "RESTYPE" on page 69 for more information about the **RESTYPE** parameter.

// 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 is created.
- If **INDEXOBJ=NONE** is specified in the processing parameter data set and no indexing keywords are specified (**FIELD**, **INDEX**, or **TRIGGER**).

//EXEC PGM=APKACIF

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

Chapter 3. ACIF Parameters

This chapter describes the ACIF parameters, including the syntax rules and values for parameters in AIX, Windows, z/OS, VM, and VSE operating systems.

Many of the parameters specified to ACIF are the same as the parameters specified to PSF or Infoprint Manager when you print a job. For those parameters that are common to both PSF or Infoprint Manager and ACIF, such as **OBJCONLIB**, **FONTLIB**, and **PSEGLIB**, you should specify the same parameter value to ACIF as specified to PSF or Infoprint Manager.

Notes:

1. For AIX or Windows, you might need to consult with your system support group for information about resource directories and other printing defaults contained in the Infoprint Manager for AIX or Infoprint Manager for Windows printer profiles used in your installation.
2. For z/OS and VSE, you might need to consult with your system programmer for information about resource library names and other printing defaults contained in the PSF startup procedures used in your installation.
3. For VM/CMS, you might need to link to the appropriate disks containing the resource files used to convert and print your job.

Syntax Rules

The following are general syntax rules for ACIF parameter files:

- Blank characters inserted between parameters, values, and symbols are allowed, but 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 ends 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.
```

Syntax Rules for AIX and Windows

In AIX and Windows, you can enter ACIF parameters with the **acif** command, in a parameter file, or both. If both are used, the value specified in the parameter file overrides the value specified with the **acif** command. To use a parameter file in AIX or Windows, specify the parameter file name with the **acif** command and the **PARMDD** parameter. For example, to use a parameter file named PARMFILE, specify:

```
acif parmdd=PARMFILE
```

The syntax for entering parameters with the **acif** command is:

```
acif [cc=value] [cctype=value] [chars=fontnames] [comsetup=name]
[cpgid=value] [dcfpagenames=value] [extensions=value] [fdeflib=pathlist]
[fieldn={record,column,length} | {'literalvalue' | X'literalvalue'}] [fileformat=value]
[fontlib=pathlist] [fontpath=pathlist] [formdef=fdefname] [groupname=value]
[imageout=value] [indexn={'attributename' | X'attributename'}{,fielddefinition}]
[indexdd=filename] [indexobj=value] [indexstartby=value]
[indexexit=programname] [inpccsid=ccsid] [inpexit=programname]
[inputdd=filename] [insertimm=value] [mcf2ref=value] [msgdd=filename]
[objconlib=pathlist] [outccsid=ccsid] [outexit=programname] [outputdd=filename]
[ovlylib=pathlist] [pagedef=pdefname] [parmdd=filename] [pdeflib=pathlist]
[prmode=value] [pseplib=pathlist] [resexit=programname] [reslib=pathlist]
[resobjdd=filename] [restype=value] [trc=value] [triggern={record | *}{,column |
*}{, 'trigger value' | X'trigger value'}] [userlib=pathlist] [userpath=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*.

Though the parameters themselves are not case-sensitive, associated values, such as file names, attribute names, and directory names in AIX and attribute names in Windows, *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).

Syntax Rules for z/OS, VM, and VSE

In z/OS, VM, and VSE, you enter ACIF parameters in a parameter file. 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:

z/OS

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)

Parameter Values

Table 4 lists the ACIF parameters and the values for the AIX, Windows, z/OS, VM, and VSE operating systems. Underscored values are the default and are used by ACIF if no other value is specified. Not all parameters are valid in every environment; parameter values are only listed for those operating systems to which they apply. The Windows operating system is referred to as “WIN” in the table.

Table 4. ACIF Parameters and Operating Systems

ACIF Parameters	Operating System	See Page...
CC = <u>{YES NO}</u>	AIX, WIN, z/OS, VM, VSE	33
CCTYPE = <u>{Z A M}</u>	AIX, WIN	34
CCTYPE = <u>{Z A M}</u>	z/OS, VM, VSE	34
CHARS = <i>fontname1</i> [, <i>fontname2</i>][, <i>fontname3</i>][, <i>fontname4</i>]	AIX, WIN, z/OS, VM, VSE	34
COMSETUP = <i>name</i>	AIX, WIN, z/OS, VM	36
CPGID = <u>{850 codepageid}</u>	AIX, WIN	38
CPGID = <u>{500 codepageid}</u>	z/OS, VM, VSE	38
DCFPAGENAMES = <u>{YES NO}</u>	AIX, WIN, z/OS, VM, VSE	38
EXTENSIONS = <u>{NONE ALL </u> [<u>PRCOLOR</u>][, <u>BOX</u>][, <u>FRACLINE</u>][, <u>CELLED</u>][, <u>SPCMPRS</u>][, <u>RESORDER</u>] [, <u>PASSOID</u>][, <u>MVSICNV</u>]	AIX, WIN, z/OS, VM, VSE	39
FDEFLIB = <i>pathlist</i>	AIX, WIN	40
FDEFLIB = <i>dsname1</i> [, <i>dsname2</i>][, <i>dsname3</i> ...]	z/OS	40
FDEFLIB = <i>filetype1</i> [, <i>filetype2</i>][, <i>filetype3</i> ...]	VM	41
FIELDn = <u>{record,column,length}</u> { ' <i>literalvalue</i> ' X ' <i>literalvalue</i> ' }	AIX, WIN, z/OS, VM, VSE	41
FILEFORMAT = <u>{RECORD RECORD,n STREAM</u> [(<u>NEWLINE</u> = <u>{value </u> <u>X'nnnn'</u>][, <u>encoding</u>])]	AIX, WIN	43
FONTTECH =UNBOUNDED	z/OS, VM, VSE	44
FONTLIB = <i>pathlist</i>	AIX, WIN	45
FONTLIB = <i>dsname1</i> [, <i>dsname2</i>][, <i>dsname3</i> ...]	z/OS	45
FONTLIB = <i>filetype1</i> [, <i>filetype2</i>][, <i>filetype3</i> ...]	VM	46
FONTPATH = <i>pathlist</i>	AIX, WIN, z/OS	46
FORMDEF = <i>fdefname</i>	AIX, WIN, z/OS, VM, VSE	47
GROUPNAME = <u>{INDEX1 INDEXn}</u>	AIX, WIN, z/OS, VM, VSE	48

Table 4. ACIF Parameters and Operating Systems (continued)

ACIF Parameters	Operating System	See Page...
IMAGEOUT={ <u>ASIS</u> <u>IOCA</u> }	AIX, WIN, z/OS, VM, VSE	49
INDEX n ={ ' <u>attributename</u> ' <u>X</u> ' <u>attributename</u> ' }, { <u>FIELD</u> n [, <u>FIELD</u> n ...] }	AIX, WIN, z/OS, VM, VSE	49
INDEXDD={ <u>INDEX</u> <u>filename</u> }	AIX, WIN	51
INDEXDD={ <u>INDEX</u> <u>ddname</u> }	z/OS, VM	51
INDEXDD={ <u>INDEX</u> <u>filename</u> (<u>DEVT</u> =TAPE <u>DISK</u>)}	VSE	51
INDEXOBJ={ <u>GROUP</u> <u>ALL</u> <u>NONE</u> <u>BDTLY</u> }	AIX, WIN, z/OS, VM, VSE	52
INDEXSTARTBY={ <u>1</u> <u>nn</u> }	AIX, WIN, z/OS, VM, VSE	52
INDEXEXIT= <u>programname</u>	AIX, WIN	53
INDEXEXIT= <u>modulename</u>	z/OS, VM, VSE	53
INPCCSID= <u>ccsid</u>	AIX, WIN	53
INPEXIT= <u>programname</u>	AIX, WIN	53
INPEXIT= <u>modulename</u>	z/OS, VM, VSE	54
INPUTDD={ <u>STDIN</u> <u>filename</u> }	AIX, WIN	54
INPUTDD={ <u>INPUT</u> <u>ddname</u> }	z/OS, VM	54
INPUTDD={ <u>INPUT</u> <u>filename</u> (<u>LRECL</u> = <u>nnnn</u> , <u>BLKSIZE</u> = <u>nnnn</u> , <u>RECFM</u> = <u>F</u> <u>FBI</u> <u>V</u> <u>VB</u> , <u>DEVT</u> =TAPE <u>DISK</u>)}	VSE	55
INSERTIMM={ <u>YES</u> <u>NO</u> }	AIX, WIN, z/OS, VM, VSE	55
MCF2REF={ <u>CPCS</u> <u>CF</u> }	AIX, WIN, z/OS, VM, VSE	56
MSGDD={ <u>STDERR</u> <u>filename</u> }	AIX, WIN	56
MSGDD={ <u>SYSPRINT</u> <u>ddname</u> }	z/OS, VM	56
OBJCONLIB= <u>pathlist</u>	AIX, WIN	56
OBJCONLIB= <u>dsname</u> 1[, <u>dsname</u> 2][, <u>dsname</u> 3...]	z/OS	57
OBJCONLIB= <u>filetype</u> 1[, <u>filetype</u> 2][, <u>filetype</u> 3...]	VM	57
OUTCCSID= <u>ccsid</u>	AIX, WIN	58
OUTEXIT= <u>programname</u>	AIX, WIN	58
OUTEXIT= <u>modulename</u>	z/OS, VM, VSE	58
OUTPUTDD={ <u>STDOUT</u> <u>filename</u> }	AIX, WIN	59
OUTPUTDD={ <u>OUTPUT</u> <u>ddname</u> }	z/OS, VM	59
OUTPUTDD={ <u>OUTPUT</u> <u>filename</u> (<u>DEVT</u> =TAPE <u>DISK</u>)}	VSE	59
OVLYLIB= <u>pathname</u>	AIX, WIN	59
OVLYLIB= <u>dsname</u> 1[, <u>dsname</u> 2][, <u>dsname</u> 3...]	z/OS	60
OVLYLIB= <u>filetype</u> 1[, <u>filetype</u> 2][, <u>filetype</u> 3...]	VM	60
PAGEDEF= <u>pdefname</u>	AIX, WIN, z/OS, VM, VSE	61
PARMDD= <u>filename</u>	AIX, WIN	63
PARMDD={ <u>SYSIN</u> <u>ddname</u> }	z/OS, VM	63
PDEFLIB= <u>pathlist</u>	AIX, WIN	63
PDEFLIB= <u>dsname</u> 1[, <u>dsname</u> 2][, <u>dsname</u> 3...]	z/OS	63
PDEFLIB= <u>filetype</u> 1[, <u>filetype</u> 2][, <u>filetype</u> 3...]	VM	64
PRMODE={ <u>SOSI</u> 1 <u>SOSI</u> 2 <u>SOSI</u> 3 <u>SOSI</u> 4 <u>aaaaaaaa</u> }	AIX, WIN, z/OS, VM, VSE	64
PSEGLIB= <u>pathlist</u>	AIX, WIN	65

Table 4. ACIF Parameters and Operating Systems (continued)

ACIF Parameters	Operating System	See Page...
PSEGLIB = <i>dsname1</i> [, <i>dsname2</i>][, <i>dsname3</i> ...]	z/OS	65
PSEGLIB = <i>filetype1</i> [, <i>filetype2</i>][, <i>filetype3</i> ...]	VM	66
RESEXIT = <i>programname</i>	AIX, WIN	66
RESEXIT = <i>modulename</i>	z/OS, VM, VSE	67
RESFILE ={ <u>SEQ</u> <u>PDS</u> }	z/OS	67
RESLIB = <i>pathlist</i>	AIX, WIN	68
RESOBJDD ={ <u>RESOBJ</u> <i>filename</i> }	AIX, WIN	68
RESOBJDD ={ <u>RESOBJ</u> <i>ddname</i> }	z/OS, VM	68
RESOBJDD ={ <u>RESOBJ</u> <i>filename</i> (<u>DEVT</u> =TAPE <u>DISK</u>)}	VSE	69
RESTYPE ={ <u>NONE</u> <u>ALL</u> [<u>FDEF</u>][, <u>PSEG</u>][, <u>OVLY</u>][, <u>FONT</u>][, <u>OBJCON</u>][, <u>BCOCA</u>][, <u>GOCA</u>][, <u>IOCA</u>][, <u>INLINE</u>]}	AIX, WIN, z/OS, VM, VSE	69
TRACE ={ <u>YES</u> <u>NO</u> }	z/OS	71
TRC ={ <u>YES</u> <u>NO</u> }	AIX, WIN, z/OS, VM, VSE	72
TRIGGER <i>n</i> ={ <i>record</i> *},{ <i>column</i> *},{ <i>'triggervalue'</i> X' <i>triggervalue</i> '}	AIX, WIN, z/OS, VM, VSE	72
UNIQUEBNGS ={ <u>YES</u> <u>NO</u> }	z/OS, VM, VSE	74
USERLIB = <i>pathlist</i>	AIX, WIN	74
USERLIB = <i>dsname1</i> [, <i>dsname2</i>][, <i>dsname3</i> ...]	z/OS	75
USERLIB = <i>filetype1</i> [, <i>filetype2</i>][, <i>filetype3</i> ...]	VM	75
USERPATH = <i>pathlist</i>	AIX, WIN, z/OS	75

The following sections describe the ACIF parameters. The format and usage is the same in all environments (AIX, Windows, z/OS, VM, and VSE) unless otherwise specified.

CC

Specifies whether the input file has 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 is formatted (single space, double space, triple space, and so forth). In addition, other carriage control characters can be used to position the line anywhere on the page. If there are no carriage control characters, single spacing is assumed.

CC={YES | NO}

The values are:

YES

The file contains carriage control characters.

NO

The file does not contain carriage control characters.

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

CCTYPE

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. The 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 occurs *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 occurs *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 is printed and the double spacing occurs *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 is moved down two lines but the line that contains the X'13' carriage control character is not printed. The next line in the data is printed at the current print position and the action for the associated carriage control character is performed *after* the line is printed.

If you are not sure which type of carriage control characters are in your input file, consult your system support group. For more information, see “Understanding How ANSI and Machine Carriage Controls Are Used” on page 192.

AIX and Windows

CCTYPE={Z | A | M}

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

z/OS, VM, and VSE

CCTYPE=Z | A | M

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

CHARS

Specifies the file name (in AIX, Windows, or VM) or the member name (in z/OS or VSE) of from one to four coded fonts that you want ACIF to use to process a file. A coded font specifies a character set and code page pair.

Note: The **CHARS** parameter is ignored if you specify the **FONTPATH** or **USERPATH** parameter for TrueType and OpenType fonts.

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

The value is:

fontname

The name of the coded font. The name is limited to four characters, consisting of any combination of alphanumeric characters (a-z, A-Z, 0–9) and special characters (# \$ @). It does not include the two-character prefix of the coded-font name (X0 through XG). In AIX and Windows, the font name is case-sensitive.

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

When ACIF is used to convert traditional line data, mixed-mode data, or unformatted ASCII data, you must specify a page definition with the **PAGEDEF** parameter. You can then specify the fonts either in the page definition or with the **CHARS** parameter, but not both. You cannot mix fonts specified in a page definition with fonts specified with **CHARS** for a single file. 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 the **CHARS** parameter.

Select fonts with table-reference characters (TRCs), with AFP structured fields, or in a page definition. If the page definition does not name any fonts, and you want to specify more than one font with the **CHARS** parameter, you must specify table reference characters (TRCs) in the input file to select the fonts. For example, if you want the file to print with these two fonts, X0GT10 (Gothic 10 pitch) and X0GT12 (Gothic 12 pitch), do the following:

- Specify **TRC=YES**.
- Use **CHARS** to associate the fonts with each TRC:

```
CHARS=GT10,GT12
```

where, **GT10** is associated with TRC 0 and **GT12** is associated with TRC 1.

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

- Specify **TRC=NO**.
- Use **CHARS** to indicate the single font in which the file should be printed. For example:

```
CHARS=GT10
```

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.

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.

AIX and Windows

The font name in AIX and Windows is case-sensitive.

If you use the ASCII fonts that are supplied with Infoprint Manager for AIX or Infoprint Manager for Windows, use the four-character short names (see Table 7 on page 93

page 93 for examples). In AIX, 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 these:

- 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
```

If the input file is unformatted ASCII, you can do one of these:

- 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.
- 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, specify the **INPEXIT** parameter. For example:

In AIX, use one of these:

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

In Windows, use one of these:

- `inpexit=\install_directory\exits\acif\apka2e.dll`
- `inpexit=\install_directory\exits\acif\asciinpe.dll`

See the **INPEXIT** parameter on page 54 for a description of **apka2e** and **asciinpe** functions.

You can also convert encoded data to another coded character set identifier (CCSID). See the **INPCCSID** parameter on page “INPCCSID” on page 53 and the **OUTCCSID** parameter on page “OUTCCSID” on page 58.

z/OS and VM

In z/OS and VM, fonts you specify must reside in a library specified with the **FONTLIB** parameter or reside in a user library specified with the **USERLIB** parameter.

VSE

In VSE, you must specify fonts in the `// LIBDEF PHASE, SEARCH=(...) JCL` statement.

COMSETUP

AIX, Windows, z/OS, and VM

Specifies the name of a COM setup file in AIX, Windows, z/OS, and VM. A COM setup file is an AFP resource that contains instructions required when printing on a microfilm device (*microfilm* can mean either microfiche or 16 mm film).

COMSETUP=*name*

The value is:

name

Any valid COM setup file name (in AIX, Windows, or VM) or member name (in z/OS). The *name* can be one to eight alphanumeric characters (a-z, A-Z, 0–9) and special characters (# \$ @), including the two-character prefix, if there is one. In AIX and Windows, *name* is case-sensitive.

Note: If the 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 **COMSETUP=MYSETUP**.

The COM setup file you use can be located:

- In a z/OS or VM library.
- In an AIX or Windows directory.
- Inline in the file (that is, within the file itself).

If the COM setup file is in an AIX or Windows directory or a z/OS or VM library, use the **USERLIB** or **OBJCONLIB** parameter to specify the path to the file or the data set. For example:

- In AIX , use one of these:
 - `comsetup=mysetup`
`userlib=/usr/afp/resources`
 - `comsetup=mysetup`
`objconlib=/usr/lib/setups`
- In Windows, use this:
 - `comsetup=mysetup`
`userlib=\install_directory\resources`
- In z/OS or VM, use one of these:
 - `COMSETUP=MYSETUP`
`USERLIB=USER.RESOURCES`
 - `COMSETUP=MYSETUP`
`OBJCONLIB=USER.SETUPS`

A COM setup file can be an inline resource in all data formats except XML. (XML data cannot have carriage control characters, which are used to identify inline resources.) If the COM setup file is an inline resource, you must do one of these:

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

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.

- 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**.

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

Specifies the three- or four-digit identifier that defines an IBM-registered code page used when the index values and attribute names are specified on the **INDEX_n** and **FIELD_n** parameters.

ACIF uses the 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 about this triplet, refer to *Mixed Object Document Content Architecture Reference*.

The code page identifier is used by programs, such as AFP Workbench Viewer, that must display indexing information. These programs use this identifier with code page translation tables to represent the index attribute and value data. 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 ends processing. For more information about code pages, refer to *IBM AFP Fonts: Technical Reference for Code Pages*, S544–3802.

AIX and Windows

CPGID={850 | *codepageid*}

The values are:

850

IBM code page 850

codepageid

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

If this parameter is not specified, ACIF uses code page 850 as the default.

z/OS, VM, and VSE

CPGID={500 | *codepageid*}

The values are:

500

IBM code page 500

codepageid

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

If this parameter is not specified, ACIF uses code page 500 as the default.

DCFPAGENAMES

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 Begin Page (BPG) structured fields with fully qualified names (FQNs), ACIF does not generate page names.

DCFPAGENAMES={YES | NO}

The values are:

YES

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

NO

ACIF generates page names using an 8-byte counter.

If this parameter is not specified, ACIF generates page names using an 8-byte counter.

EXTENSIONS

Specifies the extended options that ACIF uses. Extensions are MO:DCA-P data stream advanced features that might not be supported for all presentation devices. You should use care when choosing these options to ensure that they are supported by your print server, viewer, or printer.

EXTENSIONS={NONE | ALL | [PRCOLOR][,BOX][,FRACLINE][,CELLED][,SPCMPRS][,RESORDER][,PASSOID][,MVSICNV]}

The values are:

NONE

ACIF does not use any extended options.

ALL

ACIF uses all extended options.

Note: Be very careful when specifying ALL. More options might be added in the future that might not be supported by your presentation device.

PRCOLOR

Specifies that GOCA process color drawing orders are supported. This option is required when using the DRAWGRAPHIC command in a record format page definition. See “Drawing Graphics with Record Format Page Definitions” on page 95 for an example of using this option.

BOX

Specifies that GOCA box drawing orders are supported. This option is required when using the DRAWGRAPHIC command in a record formatting page definition. See “Drawing Graphics with Record Format Page Definitions” on page 95 for an example of using this option.

FRACLINE

Specifies that GOCA fractional line width drawing orders are supported. This option is required when using the DRAWGRAPHIC command in a record formatting page definition. See “Drawing Graphics with Record Format Page Definitions” on page 95 for an example of using this option.

CELLED

Specifies the IOCA Replicate and Trim function when converting IM1 celled images. This option might reduce the number of bytes needed for a raster image, and it might display or print faster. It requires that **IMAGEOUT=IOCA** is specified (the default).

SPCMPRS

Specifies the repeat string PTOCA order to compress embedded blanks.

RESORDER

Specifies that inline resources do not need to appear in any particular order in the input file; only before the Begin Document (BDT) structured field. When RESORDER is not specified, inline resources must appear in the input file in the order in which they are used. For example, if a coded font is inline, the character set and code page that the coded font points to must occur inline first. When RESORDER is specified, ACIF reads into memory only the inline resources actually needed to print the job and uses them when they are requested; inline resources that are not needed are not saved in the resource library.

Keep in mind:

1. When RESORDER is specified, TrueType and OpenType fonts that were originally inline in the input file are not saved in the resource library.
2. Specifying RESORDER impacts performance and storage use.

PASSOID

Specifies that ACIF passes OID information from the Resource Access Table (RAT) to the Begin Resource (BRS or BR) structured field when saving TrueType and OpenType fonts.

MVSICNV

Specifies that ACIF should call the ICONV translation services on MVS, even if TrueType and OpenType fonts are not being used. This parameter is necessary if the page definition specifies QR Code bar codes and the line data contains DBCS characters.

FDEFLIB

AIX or Windows

FDEFLIB=*pathlist*

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

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths. ACIF searches the paths in the order in which they are specified.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

FDEFLIB=*dsname1*[,*dsname2*][,*dsname3*...]

Specifies the data sets that compose the form definition library. You can specify a maximum of eight data sets. For example:

```
FDEFLIB=SYS1.FDEFLIB,USER.FDEFLIB
```

This parameter also specifies the concatenation sequence when ACIF searches for a particular form definition. ACIF first looks for the resource in *dsname1*. If it cannot find the resource in *dsname1*, it continues the search with *dsname2*, and so on, until it locates the requested resource or exhausts the list of specified data sets.

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 might differ. For information about how PSF selects resources, refer to *PSF for z/OS: Customization*.
3. For systems before MVS/DFP™ Version 2.3, data sets must be concatenated with the largest block size first.

4. **FDEFLIB** is a required parameter if **USERLIB** is not specified. If **FDEFLIB** is not specified, ACIF reports an error condition and ends processing.

VM

FDEFLIB=*filetype1*[,*filetype2*][,*filetype3*...]

Specifies the file types that define the form definition libraries. You can specify a maximum of eight file types. For example:

```
FDEFLIB=FDEF38PP,TEMPFDEF
```

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.

Notes:

1. File type values must conform to CMS naming conventions.
2. **FDEFLIB** is a required parameter if **USERLIB** is not specified. If **FDEFLIB** is not specified, ACIF reports an error condition and ends processing.

VSE

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 about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

FIELDn

Specifies the data fields to be used to construct the indexing information. These data fields can be specified as literal values (constants) or ACIF can retrieve the data from the input records of the file. You can define a maximum of 16 fields (**FIELD1** through **FIELD16**).

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

The values are:

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** parameter to determine a page-group boundary. When all of 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 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 **TRIGGERn** 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 ends 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 (X'40') to fill out the record. If the field begins outside the maximum length of the record, ACIF reports an error condition and ends processing.

literal value | X'literal value'

Specifies the literal (constant) value of the **FIELD n** parameter. 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.

Note: The literal value can be specified as ASCII character data in AIX or Windows, EBCDIC character data in z/OS, VM or VSE, or hexadecimal data. However, if the input data file is *anything other than* ASCII in AIX or Windows or EBCDIC in z/OS, VM, or VSE, the value *must* be specified as hexadecimal data (otherwise, the comparisons between the input data file and what is coded in the **FIELD n** parameter do not yield a match).

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

- FIELD1=0,2,20
- FIELD2=5,5,10
- FIELD3=-15,30,5
- FIELD4='444663821'
- FIELD5=X'0001'

In the example, the fields have these values:

- The first field 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 as character data, while the fifth field is specified as hexadecimal data.

For more information about using literal values or data values for indexing, see “Indexing with Literal Values” on page 6 and “Indexing with Data Values” on page 7.

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 lets ACIF 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.

See Chapter 4, “Enhanced Indexing Parameters,” on page 77 for information about using the **FIELD n** parameter with enhanced ACIF indexing.

FILEFORMAT

AIX and Windows

Specifies the format of the input file in AIX and Windows. If you do not specify the **FILEFORMAT** parameter, ACIF uses **STREAM** as the default.

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

FILEFORMAT=**{RECORD | RECORD,*n* | STREAM[(NEWLINE=**{CR | LF | CRLF | X'*nnnn*'}**)[*,encoding*])}**

The values are:

RECORD

The input file is formatted in S/390®, zSeries®, or System z9™ record format, where the first two bytes of each line, called the record descriptor word (RDW), specify the length of the line. Files with RECORD format typically are z/OS or VM files with a variable record format. These files are either NFS-mounted to AIX or Windows or sent using Download for z/OS.

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, and specifies the fixed length of the entire record. 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/390, zSeries, or System z9 host system, such as z/OS or VM.

STREAM

The input file has no length information; it is a stream of data separated by a new-line 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, such as AIX, Windows, or DOS.

ACIF 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 new-line character (X'0A') to delimit the end of a record. If ACIF determines that the input file is EBCDIC, ACIF looks for the EBCDIC new-line character (X'25') to delimit the end of a record. If the input record is MO:DCA-P, no new-line character is required. ACIF does not include new-line characters in the MO:DCA-P data stream that it produces.

Note: The default new-line characters might be incorrect; therefore, to ensure correct formatting results, specify NEWLINE with the STREAM parameter.

NEWLINE=**{CR | LF | CRLF | X'*nnnn*'}**[*,encoding*])

NEWLINE is an optional value of **FILEFORMAT** that is used only if STREAM is specified. You use NEWLINE to specify the characters and optional encoding for determining line breaks in the input data file. The new-line character values are:

CR Carriage returns determine line breaks.

LF Line feeds determine line breaks.

CRLF Carriage returns followed by line feeds determine line breaks.

X'nnnn'

One-, two-, or four-byte hexadecimal characters determine line breaks.

encoding

One of these values, **ASCII**, **EBCDIC**, **UTF8**, or **UTF16**, indicates which hexadecimal strings ACIF uses to determine line breaks when CR, LF, or CRLF are specified (see Table 5). If the encoding value is UTF8 or UTF16 and the UDTYPE parameter is specified as UTF16 with the PPFA PAGEDEF command, ACIF checks for a Byte Order Mark (BOM) character and, if present, reverses the bytes in the delimiter characters for UTF-16 little endian data. If the encoding value is different than the value on the UDTYPE parameter, ACIF issues a warning message.

Table 5. Hexadecimal Strings for Encoding Values

Encoding Value	CR	LF	CRLF
ASCII	X'0D'	X'0A'	X'0D0A'
EBCDIC	X'0D'	X'25'	X'0D25'
UTF8	X'0D'	X'0A'	X'0D0A'
UTF16 (big endian data)	X'000D'	X'000A'	X'000D 000A'
UTF16 (little endian data)	X'0D00'	X'0A00'	X'0D00 0A00'

You can use NEWLINE when ACIF's algorithm cannot determine the correct new-line character (if blanks are at the beginning of the file, for instance). Or you can use NEWLINE if you want to specify a new-line character that is not the standard default. If NEWLINE is not specified, ACIF uses the algorithm specified under **FILEFORMAT=STREAM**.

These examples show how to use NEWLINE:

```
FILEFORMAT=STREAM, (NEWLINE=X'0D0A')
FILEFORMAT=STREAM, (NEWLINE=X'000D000A')
FILEFORMAT=STREAM, (NEWLINE=CRLF, UTF16)
```

FONTECH

z/OS, VM, and VSE

Indicates that ACIF should process 3800 (unbounded box) fonts in z/OS, VM, and VSE.

Note: The **FONTECH** parameter must be used with caution. Unbounded fonts are supported only by the IBM 3800 printer. They are not supported by any other printer or the AFP Workbench Viewer. Any resource object file archived has very limited use. Unbounded box fonts cannot be used by Infoprint Manager for AIX or Infoprint Manager for Windows.

FONTECH=UNBOUNDED

Indicates that ACIF should process 3800 (unbounded box) fonts. Any value other than UNBOUNDED causes ACIF to issue an error message and end processing.

If you specify **FONTECH=UNBOUNDED** and **RESTYPE=FONT** or **RESTYPE=ALL**, ACIF reads unbounded box fonts and saves them in the

resource object data set. However, the unbounded box fonts are not syntax checked. If there are errors in the AFP data stream making up the font, ACIF does not issue an error message.

You cannot mix unbounded box fonts and TrueType or OpenType fonts in the same document. Therefore, if you specify **FONTPATH** and **USERPATH**, ACIF ignores them when **FONTECH=UNBOUNDED** is specified.

FONTLIB

Specifies the location of FOCA raster or outline fonts. **FONTLIB** is not used to specify directories for TrueType and OpenType fonts. Instead, use the **FONTPATH** or **USERPATH** parameter.

AIX and Windows

FONTLIB=*pathlist*

Specifies the paths where FOCA raster or outline fonts are installed.

The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths. ACIF searches the paths in the order in which they are specified.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

FONTLIB=*dsname1[,dsname2][,dsname3...]*

Specifies the data sets that contain the FOCA raster or outline fonts. You can specify a maximum of eight data sets. For example:

```
FONTLIB=SYS1.FONTLIB,USER.FONTLIB
```

This parameter also specifies the concatenation sequence when ACIF searches for a particular font resource. ACIF first looks for the resource in *dsname1*. If it cannot find the resource in *dsname1*, it continues the search with *dsname2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

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 might differ. For information about how PSF selects resources, refer to *PSF for z/OS: Customization*.
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 and **USERLIB** is not specified, or if **MCF2REF=CPCS** and any coded fonts are referenced in the input file or in an overlay. The **RESTYPE** parameter 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 ends processing.

VM

FONTLIB=*filetype1*[,*filetype2*][,*filetype3*...]

Specifies the file types that define the FOCA raster or outline font libraries. If your page definition or AFP input file refers to outline fonts on the Map Coded Font (MCF) structured fields, include an outline font library in the search order.

You can specify a maximum of eight file types. For example:

```
FONTLIB=FONT3820,TESTFONT
```

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.

Notes:

1. File type values must conform to CMS naming conventions.
2. This is a required parameter if font retrieval is requested and **USERLIB** is not specified, or if **MCF2REF=CPCS** and any coded fonts are referenced in the print file or in an overlay. The **RESTYPE** parameter 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 ends processing.

VSE

This parameter is not used for VSE. Font resources are located in the library defined by the **// LIBDEF PHASE,SEARCH=(...)** JCL statement. For information about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

FONTPATH

This parameter is not supported for VM and VSE; if specified, you see an error message.

AIX, Windows, and z/OS

FONTPATH=*pathlist*

Specifies the system font path library directories in which TrueType and OpenType fonts are stored. TrueType and OpenType fonts are Unicode-enabled AFP fonts that are not defined by the IBM Font Object Content Architecture (FOCA). The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX, a semicolon (;) in Windows, or a comma (,) in z/OS to separate multiple paths. For example:

AIX or Windows

```
acif inputdd=INFILE outputdd=OUTFILE pagedef=PAGTRUE formdef=F1A10110 \
fontpath=/u/fonts/truetype:/u/fonts/truetype/local
```

Note: The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

z/OS

```
INPUTDD=INFILE
OUTPUTDD=OUTFILE
PAGEDEF=PAGTRUE
FORMDEF=F1A10110
FONTPATH='/u/fonts/truetype','/u/fonts/truetype/local/'
```

ACIF searches the paths in the order in which they are specified.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

FORMDEF

Specifies the file name (in AIX, Windows, or VM) or the member name (in z/OS or VSE) 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. ACIF uses a form definition only at print time to retrieve resources; it does not use a form definition at transform time to convert data streams.

FORMDEF=*fdefname*

The value is:

fdefname

Any valid form definition name. The *fdefname* can be one to eight alphanumeric characters (a-z, A-Z, 0–9) and special characters (# \$ @), including the two-character prefix, if there is one. Unlike PSF for z/OS, PSF/VM, and PSF/VSE, ACIF does **not** require the name to begin with an F1 prefix; however, if the name does begin with F1, you cannot omit it. For example:

```
FORMDEF=F1USER10
```

Notes:

1. In AIX, the *fdefname* is case-sensitive.
2. 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**.
3. ACIF requires a form definition to process the input file (even though the form definition actually gets used at print time). If you do not specify the **FORMDEF** parameter or you specify **FORMDEF** without a form definition file name, ACIF reports an error condition and ends processing.

The form definition you use can be located:

- Inline in the file (that is, within the file itself)
- In an AIX or Windows directory
- In a z/OS or VM user library referenced in the **USERLIB** parameter
- In an z/OS or VM library referenced in the **FDEFLIB** parameter
- In a VSE library referenced in the **// LIBDEF PHASE,SEARCH=(...) DLBL** JCL statement

If the form definition file is in an AIX or Windows directory or a z/OS or VM library, use the **USERLIB** or **FDEFLIB** parameter to specify the path to the file or the data sets. For example:

- In AIX, use one of these:
 - `formdef=memo`
`userlib=/usr/afp/resources`
 - `formdef=memo`
`fdeflib=/usr/lib/formdefns`
- In Windows, use this:
 - `formdef=memo`
`userlib=\install_directory\resources`
- In z/OS or VM, use one of these:
 - `FORMDEF=MEMO`
`USERLIB=USER.RESOURCES`
 - `FORMDEF=MEMO`
`FDEFLIB=USER.FORMDEFNS`

A form definition can be an inline resource in all data formats except XML. (XML data cannot have carriage control characters, which are used to identify inline resources.) If the form definition is an inline resource, you must do these:

- Include an inline form definition in the file.
- Specify **CC=YES** to indicate that the file contains 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:
 - Specify variable length records in z/OS or VSE for the record format (variable blocked with ANSI carriage control characters [VBA] or variable blocked with machine carriage control characters [VBM]).
 - Specify variable length records in VM for the record format.
- Specify **FORMDEF** with one of these values:
 - *fdefname*, where *fdefname* is the name of the inline form definition.
If the name specified in the **FORMDEF** parameter does not match the name of an inline form definition, ACIF looks for the form definition in the **FORMDEF** search path.
 - **DUMMY**
If you specify **FORMDEF=DUMMY** but the file does not include an inline form definition, ACIF looks for the form definition named DUMMY. If ACIF cannot find a form definition named DUMMY, it reports an error and ends processing.

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. By changing the form definition name in the **FORMDEF** parameter on different printing jobs, you can test different form definitions.

GROUPNAME

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. The AFP 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.

GROUPNAME={INDEX1 | INDEX n }

The values are:

INDEX1

ACIF uses the value of **INDEX1** for the group name.

INDEX n

ACIF uses the value of the specified **INDEX** (**INDEX1**, **INDEX2**, **INDEX3**,...**INDEX8**) for the group name.

If **GROUPNAME** is not specified, ACIF uses **INDEX1** as the default.

See Chapter 4, “Enhanced Indexing Parameters,” on page 77 for information about using group indexes and triggers with enhanced ACIF indexing.

IMAGEOUT

Specifies the format of the image data produced by ACIF in the output document.

IMAGEOUT={ASIS | IOCA}

The values are:

ASIS

Specifies that ACIF produce all image data in the same format as in the input file.

IOCA

Specifies that ACIF produce all image data in uncompressed Image Object Content Architecture (IOCA) format.

If **IMAGEOUT** is not specified, ACIF uses **IOCA** as the default.

INDEX 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.

INDEX n ={ 'attributename' | X'attributename' }{,FIELD n [,FIELD n ...]}

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

'attributename' | X'attributename'

Specifies a user-defined attribute name to be associated with the actual index value. The attribute name is a label for 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). 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.

Note: The attribute name can be specified as ASCII character data in AIX or Windows, EBCDIC character data in z/OS, VM or VSE, or hexadecimal data. However, if the input data file is *anything other*

than ASCII in AIX or Windows or EBCDIC in z/OS, VM, or VSE, the value *must* be specified as hexadecimal data.

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 ends processing if this occurs.

If literal values (constants) 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 ends processing if literal values are specified for all **INDEX n** parameters and if any **TRIGGER n** parameters are also specified.

For **FIELD n** 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 n** parameters.

A valid set of index parameters comprises either of these:

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

You can also specify the same **FIELD n** parameters in more than one **INDEX n** parameter.

Note: If one or more **TRIGGER n** parameters are specified (that is, ACIF indexes 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 ends processing if this rule is not satisfied.

The following example specifies that the first index tag for the patent number is made up of the literal character string '1234567', while the second index tag for the employee name is made up of fields within the file records.

```
FIELD1='1234567'  
FIELD2=0,10,20  
FIELD3=0,25,20  
INDEX1='Patent Number',FIELD1  
INDEX2='Employee Name',FIELD2,FIELD3
```

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

```
FIELD1='123456'  
FIELD2='444556677'  
INDEX1='Account Number',FIELD1  
INDEX2='Social Security Number',FIELD2
```

Note: The preceding examples are based on character input data. If the input data was *not* ASCII in AIX or Windows or EBCDIC in z/OS, VM, or VSE, the literal values used in these examples would be expressed in hexadecimal strings. For an AIX example using hexadecimal strings, see Figure 15 on page 99.

See Chapter 4, “Enhanced Indexing Parameters,” on page 77 for information about using the **INDEX***n* parameter with enhanced ACIF indexing.

INDEXDD

AIX and Windows

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. The 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 and Windows file names.

If you specify the file name without a path, ACIF puts the index object file into your current directory. If **INDEXDD** is not specified, ACIF uses **INDEX** as the default file name.

z/OS and VM

INDEXDD=**{INDEX | ddname}**

Specifies the DD name for the index object file. The DD name is a 1- to 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 DD name. These are suggested DCB characteristics for the file:

- A block size of 32760
- A maximum record length of 32756

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the INDEX DD statement. If that happens, ACIF ends processing abnormally.

- Variable blocked format
- Physical sequential format

If **INDEXDD** is not specified, ACIF uses **INDEX** as the default DD name.

VSE

INDEXDD=**{INDEX | filename(DEVT=TAPE | DISK)}**

Specifies the file name and device type that appears on the DLBL or TLBL JCL statement. The file name is a 1- to 7-byte character string containing only those alphanumeric characters supported in VSE. The device type is either **TAPE** or **DISK**. These are the DTF characteristics for the file:

- A block size of 32760
- A maximum record length of 32756

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the DLBL or TLBL JCL statement. If that happens, ACIF ends processing abnormally.

- Variable blocked format
- Assigned to programmer logical unit 009

If **INDEXDD** is not specified, ACIF uses **INDEX** as the default file name and **DISK** as the default device type.

INDEXOBJ

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

INDEXOBJ={GROUP | ALL | NONE | BDTLY}

The values are:

GROUP

ACIF places only group-level entries into the index object file, which saves space.

ALL

ACIF 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 AFP Workbench Viewer application.

NONE

ACIF suppresses the collection of all index-level information. Select **NONE** if you do not require an external index file. Selecting **NONE** also reduces ACIF storage requirements.

BDTLY

ACIF normally removes any Begin Document (BDT) and End Document (EDT) structured fields from the input file and generates a single BDT/EDT for the entire output because MO:DCA-P indexes are relative to the BDT structured field. However, after the ACIF output goes to PSF for printing, the printer stapling function uses the BDT/EDT to indicate document boundaries for stapling. Specifying **BDTLY** (BDT only) tells ACIF to pass all BDT/EDT structured fields found in the document into the output data stream in the same order they are found, and not create any additional BDT/EDT pairs. This file is suitable for printing, but should not be used with indexing because the resulting index is not MO:DCA-P compliant and might not be processed correctly by programs that use the index.

If this parameter is not specified, ACIF uses **GROUP** as the default.

INDEXSTARTBY

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

INDEXSTARTBY={_ | nn}

The values are:

_ Specifies that ACIF must find an index on the first page.

nn Specifies the output page number (0–99) by which ACIF must find the index criteria specified. Zero (0) indicates that there is no limit to the page where ACIF must find an indexing field.

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.

INDEXEXIT

Specifies the 1- to 8-byte character name of the index record exit program.

AIX and Windows

INDEXEXIT=*programname*

This is the program ACIF calls for every record (structured field) it writes in the index object file (specified with the **INDEXDD** parameter). If you specify the program 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 does 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.

z/OS, VM, and VSE

INDEXEXIT=*modulename*

This is the name of the module ACIF loads during initialization and subsequently calls for every record (structured field) it writes to the index object file (specified with the **INDEXDD** parameter). If this parameter is not specified, no index record exit is used. See “Index Record Exit” on page 118 for more detailed information.

INPCCSID

AIX and Windows

Specifies a valid coded character set identifier (CCSID) for the input code page you want to convert to another CCSID. This parameter can be used by an input record exit program, such as **apka2e** or **asciinpe** to translate input data streams (see “Using ACIF User Input Record Exits in AIX and Windows” on page 116 for more information).

INPCCSID=*ccsid*

The value is:

ccsid

Any valid CCSID, which is a three to five-character decimal value in the range 00000 to 65535 that is registered by the Character Data Representation Architecture (CDRA). You can replace leading zeros with spaces.

For information about CCSIDs, see *CDRA Reference and Registry*, SC09-2190.

INPEXIT

Specifies the 1- to 8-byte character name of the input record exit program.

AIX and Windows

INPEXIT=*programname*

ACIF calls this program for every record (every line) it reads from the input file (specified with the **INPUTDD** parameter). 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 you do not specify this parameter, ACIF does 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 these exit programs supplied with Infoprint Manager for AIX or Infoprint Manager for Windows:

/usr/lpp/psf/bin/apka2e (AIX) or \install_directory\exits\aciflapka2e.dll (Windows)

Converts ASCII stream data to EBCDIC stream data. You can also convert encoded data to another coded character set identifier (CCSID) if you specify the **INPCCSID** and **OUTCCSID** parameters.

/usr/lpp/psf/bin/asciinp (AIX) or \install_directory\exits\acifasciinp.dll (Windows)

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

/usr/lpp/psf/bin/asciinpe (AIX) or \install_directory\exits\acifasciinpe.dll (Windows)

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. You can also convert encoded data to another coded character set identifier (CCSID) if you specify the **INPCCSID** and **OUTCCSID** parameters.

If your input file uses fonts that have ASCII code points (such as **CHARS=H292**), you should *not* use the **apka2e** or **asciinpe** exit programs. However, if your unformatted ASCII file contains carriage returns and form feeds, you might want to specify the **asciinp** exit program that is supplied with Infoprint Manager for AIX or Infoprint Manager for Windows.

z/OS, VM, and VSE

INPEXIT=*modulename*

This is the name of the module ACIF loads during initialization and subsequently calls for every input record it reads from the input file (specified with the **INPUTDD** parameter). If this parameter is not specified, no input record exit is used. See “Input Record Exit” on page 114 for more detailed information.

INPUTDD

AIX and Windows

INPUTDD={STDIN | filename}

Specifies the full path name of the input file that ACIF processes. If you do not specify **INPUTDD**, ACIF uses **STDIN** as the default.

z/OS and VM

INPUTDD={INPUT | ddname}

Specifies the DD name for the file ACIF processes. The DD name is a 1- to 8-byte character string containing only those alphanumeric characters supported in the operating environment. When ACIF processes a file, it reads from this DD name.

If **INPUTDD** is not specified, ACIF uses **INPUT** as the default DD name.

VSE

INPUTDD={INPUT | *filename* (LRECL=nnnn,BLKSIZE=nnnn,RECFM=FIFBIVVB,DEVT=TAPE | DISK)}

Specifies the file name and file characteristics that appear on the DLBL or TLBL JCL statement. The file name is a 1- to 7-byte character string containing only those alphanumeric characters supported in the operating environment.

The values are:

INPUT

ACIF uses **INPUT** as the name for the input file.

filename

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

LRECL=nnnn

Specifies the record length of the input data set.

BLKSIZE=nnnn

Specifies the block size of the input data set.

RECFM=FIFBIVVB

Specifies the record format of the input data set.

F Fixed

FB

Fixed Block

V Variable

VB

Variable Block

DEVT=TAPE | DISK

Specifies the device type, either **TAPE** or **DISK**.

Note: ACIF supports SAM or VSAM-managed SAM. It does not support VSAM ISDS, ESDS, or RRDS.

If **INPUTDD** is not specified, ACIF uses these default values:

- **INDEX** for the file name
- 133 bytes for the record length
- Unblocked records
- **F** for the record format
- **DISK** as the default device type
- Assigned to programmer logical unit 006

INSERTIMM

Specifies whether ACIF is to insert an Invoke Medium Map (IMM) structured field before the first Begin Page (BPG) structured field of every named page group.

INSERTIMM={YES | NO}

The values are:

YES

Specifies that ACIF inserts an IMM before the first BPG structured field in the named page group if no IMM was encountered within the named page group.

NO

Specifies that an IMM is not inserted before the first BPG structured field.

If this parameter is not specified, ACIF uses **NO** as the default.

MCF2REF

Specifies the way ACIF builds the Map Coded Font Format 2 (MCF-2) structured field in the OUTPUT file and the RESOBJ file.

MCF2REF={CPCS | CF}

The values are:

CPCS

ACIF uses the names of the code page and character set to build the MCF-2 structured field. ACIF opens and reads the contents of all coded fonts specified in MCFs in the input file or input resources.

CF

ACIF uses the name of the coded font to build the MCF-2 structured field. This is recommended when processing double-byte character set (DBCS) fonts. Specifying **CF** improves ACIF performance because, if **RESTYPE=FONT** or **RESTYPE=ALL** is not specified, ACIF does not have to read the coded fonts from the font library.

If this parameter is not specified, ACIF uses **CPCS** as the default.

MSGDD

AIX and Windows

MSGDD={STDERR | *filename*}

Specifies the name or the full path name of the file where ACIF writes error messages. If you specify the file name without a path, ACIF puts the error file into your current directory.

If **MSGDD** is not specified, ACIF uses **STDERR** as the default for its message output.

z/OS and VM

MSGDD={SYSPRINT | *ddname*}

Specifies the DD name for the file where ACIF writes error messages. The DD name is a 1- to 8-byte character string containing only those alphanumeric characters supported in the operating environment. When ACIF processes a file, it writes to the DD name.

If **MSGDD** is not specified, ACIF uses **SYSPRINT** as the default DD name.

OBJCONLIB

AIX and Windows

OBJCONLIB=*pathlist*

Specifies the directories in which object container setup files and non-OCA objects are stored. ACIF supports these object container resources: color

mapping table, encapsulated PostScript, and microfilm setup. A COM setup file consists of a MO:DCA structure called an object container. (For more information about the COM setup file, see “COMSETUP” on page 36.) Non-OCA objects are resources other than BCOCA, GOCA, IOCA, and page segment.

The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths. ACIF searches the paths in the order in which they are specified.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

OBJCONLIB=*dsname1*[,*dsname2*][,*dsname3*...]

Specifies the data sets that compose the setup file library. You can specify a maximum of eight data sets. For example:

```
OBJCONLIB=SYS1.OBJCONLIB,USER.OBJCONLIB
```

This parameter also specifies the concatenation sequence when ACIF searches for a particular setup file. ACIF first looks for a setup file in *dsname1*. If it cannot find the setup file in *dsname1*, it continues the search with *dsname2*, 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**.

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 might differ. For information about how PSF selects resources, refer to *PSF for z/OS: Customization*.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.

VM

OBJCONLIB=*filetype1*[,*filetype2*][,*filetype3*...]

Specifies the file types that define the setup file library. You can specify a maximum of eight file types. For example:

```
OBJCONLIB=OBJ3820,TEMPOBJ
```

This parameter also specifies the search order in which ACIF searches for a particular setup file. 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.

Note: File type values must conform to CMS naming conventions.

VSE

This parameter is not used for VSE. Object container resources are located in the library defined by the **// LIBDEF PHASE,SEARCH=(...)** JCL statement. For information about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

OUTCCSID

AIX and Windows

Specifies a valid coded character set identifier (CCSID) for the output code page you want to have converted. This parameter can be used by an input record exit program, such as **apka2e** or **asciinpe**, to specify the encoding of the output data (see “Using ACIF User Input Record Exits in AIX and Windows” on page 116 for more information).

OUTCCSID=ccsid

The value is:

ccsid

Any valid CCSID, which is a three to five-character decimal value in the range 00000 to 65535 that is registered by the Character Data Representation Architecture (CDRA). You can replace leading zeros with spaces.

For information about CCSIDs, see *CDRA Reference and Registry*, SC09-2190.

OUTEXIT

AIX and Windows

OUTEXIT=programname

Specifies the name or the full path name of the output record exit program. ACIF calls this program for every output record (every line) it writes to the output document file (specified with the **OUTPUTDD** parameter). If you specify the file name without a path, ACIF searches for the file name in the paths specified by the **PATH** environment variable. If you do not specify this parameter, ACIF does 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.

z/OS, VM, and VSE

OUTEXIT=modulename

Specifies the name of the output record exit program. This is a 1- to 8-byte character name of the module ACIF loads during initialization and subsequently calls for every output record it writes to the output document file (specified with the **OUTPUTDD** parameter). If this parameter is not specified, no output record exit is used. See “Output Record Exit” on page 119 for more detailed information.

OUTPUTDD

AIX and Windows

OUTPUTDD={STDOUT | *filename*}

Specifies the name or the full path name of the output document file. If you specify the file name without a path, ACIF puts the output file into your current directory.

If **OUTPUTDD** is not specified, ACIF uses **STDOUT** as the default.

z/OS and VM

OUTPUTDD={OUTPUT | *ddname*}

Specifies the DD name for the output document file ACIF produces when it processes a file. The DD name is a 1- to 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 DD name. Suggested DCB characteristics of the file are:

- Variable blocked format
- A maximum record length of 32756

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the OUTPUT DD statement. If this happens, ACIF ends processing abnormally.

- A block size of 32760
- Physical sequential format

If **OUTPUTDD** is not specified, ACIF uses **OUTPUT** as the default DD name.

VSE

OUTPUTDD={OUTPUT| *filename* (DEVT=TAPE** | **DISK**)}**

Specifies the file name and file characteristics that appears on the DLBL or TLBL JCL statement. The file name is a 1- to 7-byte character string containing only those alphanumeric characters supported in the operating environment. Characteristics of the file are:

- A block size of 32760
- A maximum record length of 32756

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the DLBL or TLBL JCL statement. If that happens, ACIF ends processing abnormally.

- Variable blocked format
- Assigned to programmer logical unit 007

If **OUTPUTDD** is not specified, ACIF uses **OUTPUT** as the default file name and **DISK** as the default device type.

OVLYLIB

AIX and Windows

OVLYLIB=*pathlist*

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

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths. ACIF searches the paths in the order in which they are specified.

You should specify the same value for the **OVLYLIB** parameter to ACIF as specified to Infoprint Manager for AIX or Infoprint Manager for Windows. 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 about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

OVLYLIB=*dsname1*[,*dsname2*][,*dsname3*...]

Specifies the data sets that compose the overlay library. You can specify a maximum of eight data sets. For example:

```
OVLYLIB=SYS1.OVLYLIB,USER.OVLYLIB
```

The parameter also specifies the concatenation sequence when ACIF searches for a particular overlay resource. ACIF first looks for the resource in *dsname1*. If ACIF cannot find the resource in *dsname1*, it continues the search with *dsname2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

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 might differ. For information about how PSF selects resources, refer to *PSF for z/OS: Customization*.
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 and **USERLIB** is not specified. 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 ends processing.

VM

OVLYLIB=*filetype1*[,*filetype2*][,*filetype3*...]

Specifies the file types that define the overlay libraries. You can specify a maximum of eight file types. For example:

```
OVLYLIB=OVLY38PP,TEMPOVLY
```

This 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.

Notes:

1. File types must conform to CMS naming conventions.
2. This is a required parameter if overlay retrieval is requested and **USERLIB** is not specified. The **RESTYPE** parameter determines whether overlays are

to be retrieved for inclusion in the resource file. If **OVLYLIB** is not specified, and overlay retrieval is requested, ACIF reports an error condition and ends processing.

VSE

This parameter is not used for VSE. Overlay resources are located in the library defined by the **// LIBDEF PHASE,SEARCH=(...)** JCL statement. For information about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

PAGEDEF

Specifies the file name (in AIX, Windows, or VM) or the member name (in z/OS or VSE) of the page definition. A page definition defines the page format that ACIF uses to compose line data, XML data, mixed-mode data, and unformatted ASCII data into pages; it is not used with MO:DCA-P data. Page definitions are only used by ACIF at transform time to convert data streams; they are not used by PSF and Infoprint Manager to print the output produced by ACIF.

Note: Inline page definitions are removed from the output data, even if **RESTYPE=INLINE**. Page definitions are not saved in the output resource library.

PAGEDEF=*pdefname*

The value is:

pdefname

Any valid page definition name. The *pdefname* can be one to eight alphanumeric characters (a-z, A-Z, 0–9) and special characters (# \$ @), including the two-character prefix, if there is one. Unlike PSF for z/OS, PSF/VM, and PSF/VSE, ACIF does **not** require the name to begin with a P1 prefix; however, if the name does begin with P1, you cannot omit it. For example:

```
PAGEDEF=P1USER10
```

Notes:

1. In AIX, the *pdefname* is case-sensitive.
2. 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**.
3. ACIF 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 line data, XML 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 the **PAGEDEF** parameter or you specify **PAGEDEF** without a page definition file name, ACIF reports an error condition and ends processing.
4. If you use the **PAGEDEF** parameter to specify a page definition that names fonts, but you also use the **CHARS** parameter to specify fonts, the **CHARS** parameter is ignored. Therefore, if your page definition names fonts, you should not use the **CHARS** parameter.
5. 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 you use can be located:

- Inline in the file (that is, within the file itself)
- In an AIX or Windows directory
- In a z/OS or VM user library referenced in the **USERLIB** parameter
- In a library referenced in the **PDEFLIB** parameter
- In a VSE library referenced in the **// LIBDEF PHASE,SEARCH=(...) DLBL JCL** statement

If the page definition file is in an AIX or Windows directory or a z/OS or VM library, use the **USERLIB** or **PDEFLIB** parameter to specify the path to the file or the data sets. For example:

- In AIX, use one of these:
 - `pagedef=memo`
`userlib=/usr/afp/resources`
 - `pagedef=memo`
`pdeflib=/usr/lib/pagedefns`
- In Windows, use this:
 - `pagedef=memo`
`userlib=\install_directory\resources`
- In z/OS or VM, use one of these:
 - `PAGEDEF=MEMO`
`USERLIB=USER.RESOURCES`
 - `PAGEDEF=MEMO`
`PDEFLIB=USER.PAGEDEFNS`

A page definition can be an inline resource in all data formats except XML. (XML data cannot have carriage control characters, which are used to identify inline resources.) If the page definition is an inline resource, you must do these:

- Include an inline form definition in the file.
- Specify **CC=YES** to indicate that the file contains 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 variable length records in z/OS or VSE for the record format (variable blocked with ANSI carriage control characters [VBA] or variable blocked with machine carriage control characters [VBM]).
 - Specify variable length records in VM for the record format.
- Specify **PAGEDEF** with one of these values:
 - *pdefname*, where *pdefname* is the name of the inline page definition. If the name specified in the **PAGEDEF** parameter does not match the name of an inline page definition, ACIF looks for the page definition in the **PAGEDEF** search path or uses the page definition from the resource library.
 - **DUMMY**
If you specify **PAGEDEF=DUMMY** but the file does not include an inline page definition, ACIF looks for the page definition named DUMMY. If ACIF cannot find a form definition named DUMMY, it reports an error and ends processing.

An input file can contain multiple page definitions, but only one page definition can be used by ACIF. 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**, ACIF uses the first inline page definition in the input file. By changing the page definition name in the **PAGEDEF** parameter on different printing jobs, you can test different page definitions.

PARMDD

AIX and Windows

PARMDD=filename

Specifies the name or the full path name of the parameter file that contains ACIF parameters and values. This parameter is specified with the **acif** command. For example, to use a parameter file named **PARMFILE**, specify:

```
acif parmd=PARMFILE
```

If you specify the file name without a path, ACIF searches for the file name in your current directory.

z/OS and VM

PARMDD={SYSIN | ddname}

Specifies the DD name for the parameter file that contains ACIF parameters and values. The DD name is a 1- to 8-byte character string containing only those alphanumeric characters supported in the operating environment. This parameter is specified in an EXEC statement or on the command line.

If **PARMDD** is not specified, ACIF uses **SYSIN** as the default DD name.

PDEFLIB

AIX and Windows

PDEFLIB=pathlist

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

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths. ACIF searches the paths in the order in which they are specified.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

PDEFLIB=dsname1[,dsname2][,dsname3...]

Specifies the data sets that compose the page-definition library. You can specify a maximum of eight data sets. For example:

```
PDEFLIB=SYS1.PDEFLIB,USER.PDEFLIB
```

This parameter also specifies the concatenation sequence when ACIF searches for a particular page definition. ACIF first looks for the resource in *dsname1*. If ACIF cannot find the resource in *dsname1*, it continues the search with *dsname2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

If **USERLIB** is also specified, ACIF searches for the resource in the data sets specified in **USERLIB** before searching the data sets identified in **PDEFLIB**.

Notes:

1. Data sets 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 might differ. For information about how PSF selects resources, refer to *PSF for z/OS: Customization*.
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 input file contains any line data and **USERLIB** is not specified. If this parameter is not specified and the input file contains line data, ACIF reports an error condition and ends processing.

VM

PDEFLIB=*filetype1* [, *filetype2*] [, *filetype3* ...]

Specifies the file types that define the page-definition libraries. You can specify a maximum of eight file types. For example:

```
PDEFLIB=PDEF38PP,TESTPDEF
```

This 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.

Notes:

1. The file types must conform to CMS naming conventions.
2. This is a required parameter if the print file contains any line data and **USERLIB** is not specified. If this parameter is not specified, and the print file contains any line data, ACIF reports an error condition and ends processing.

VSE

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 about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

PRMODE

Specifies the type of data in the input file and whether ACIF must perform optional processing of that data.

PRMODE={**SOSI1** | **SOSI2** | **SOSI3** | **SOSI4** | *aaaaaaaa*}

The values are:

SOSI1

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

SOSI2

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

SOSI3

Specifies that each shift-out character be converted to a Set Coded Font Local text control. Each shift-in is converted to a Set Coded Font Local Text control and two blanks. This SOSI3 data conversion is the same as the one performed by PSF for z/OS.

SOSI4

Specifies that each shift-out, shift-in code be skipped and not counted when calculating offsets for the input file. SOSI4 is used when double-byte character set (DBCS) text is converted from ASCII to EBCDIC. When SOSI4 is specified, the page definition offsets are correct after conversion; therefore, the user does not need to account for SOSI characters when computing FIELD offsets. The processing of shift-out and shift-in codes for SOSI4 is the same as for SOSI2.

aaaaaaaa

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

For the SOSI processing 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

AIX and Windows

PSEGLIB=*pathlist*

Specifies the directories in which page segments and BCOCA, GOCA, and IOCA objects are stored. The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths. ACIF searches the paths in the order in which they are specified.

You should specify the same value for the **PSEGLIB** parameter to ACIF as specified to Infoprint Manager for AIX or Infoprint Manager for Windows. 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 about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

PSEGLIB=*dsname1*[,*dsname2*][,*dsname3*...]

Specifies the data sets that compose the page segment library. You can specify a maximum of eight data sets. For example:

PSEGLIB=SYS1.PSEGLIB,USER.PSEGLIB

This parameter also specifies the concatenation sequence when ACIF searches for a particular page segment or BCOCA, GOCA, or IOCA object. ACIF first looks for the resource in *dsname1*. If it cannot find the resource in *dsname1*, it continues the search with *dsname2*, and so on, until it either locates the requested resource or exhausts the list of specified data sets.

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. 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 might differ. For information about how PSF selects resources, refer to *PSF for z/OS: Customization*.
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 and **USERLIB** is not specified. 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 ends processing.

VM

PSEGLIB=*filetype1*[,*filetype2*][,*filetype3*...]

Specifies the file types that define the page segment libraries. You can specify a maximum of eight file types. For example:

```
PSEGLIB=PSEG38PP,PSEGTEST
```

This parameter 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.

Notes:

1. The file types must conform to CMS naming conventions.
2. This is a required parameter if page segment retrieval is requested and **USERLIB** is not specified. 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 ends processing.

VSE

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 about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

RESEXIT

AIX and Windows

RESEXIT=*programname*

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

programname

Any valid resource exit program name. The exit program name is case-sensitive.

z/OS, VM, and VSE

RESEXIT=*modulename*

Specifies the name of the resource exit program. This is a 1- to 8-byte character name of the 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 121 for more detailed information.

RESFILE

z/OS

RESFILE={SEQ | PDS}

Specifies the format of the resource file created by ACIF in z/OS. ACIF can create either a sequential data set or a partitioned data set (PDS) from the resources it retrieves from the PSF for z/OS resource libraries.

The values are:

SEQ

Creates a resource group that can be concatenated to the document file as inline resources.

The format of the resource file specified for TrueType and OpenType fonts must be **SEQ**.

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.

If this parameter is not specified, ACIF writes to the DD name specified in the **RESOBJDD** parameter, assuming a sequential format. See “Format of the Resources File” on page 203 for more information about the contents of the resource data set.

It is important that the parameters you use to allocate the **RESOBJDD** data set be compatible with the value of the **RESFILE** parameter. For example, if **RESFILE=PDS**, then **DSORG=PO** must be specified in the DD statement of the data set named by the **RESOBJDD** parameter. In addition, the **SPACE** parameter must include a value for directory blocks, such as **SPACE=(12288,(150,15,15))**, in the DD statement of the data set named by the **RESOBJDD** parameter.

If **RESFILE=SEQ** is specified, then **DSORG=PS** must be specified in the DD statement of the data set named by the **RESOBJDD** parameter. In addition, the **SPACE** parameter must not include a directory value, as in **SPACE=(12288,(150,15))**, in the DD statement of the data set named by the **RESOBJDD** parameter. Failure to allocate the data set named by the **RESOBJDD** parameter in a manner compatible with the specification of the **RESFILE** parameter might result in a **RESOBJDD** data set that is unusable.

If **RESFILE** is not specified, ACIF uses **SEQ** as the default.

RESLIB

AIX and Windows

RESLIB=*pathlist*

Specifies the paths for the system resource directories in Infoprint Manager for AIX and Infoprint Manager for Windows. The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths.

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). However, **RESLIB** is not used to specify directories for TrueType and OpenType fonts. Instead, use the **FONTPATH** or **USERPATH** parameter.

In most cases, you want ACIF to find the same resources that Infoprint Manager for AIX or Infoprint Manager for Windows uses when it prints the file. If so, the **RESLIB** paths should be the same as the paths specified with the **RESPATH** parameter to Infoprint Manager.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

RESOBJDD

AIX and Windows

RESOBJDD=**{RESOBJ | filename}**

Specifies the name or the full path name for the resource file that ACIF writes data to. 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. Values are:

RESOBJ

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

filename

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

If you specify the file name without a path, ACIF puts the resource file into your current directory. If **RESOBJDD** is not specified, ACIF uses **RESOBJ** as the default file name.

z/OS and VM

RESOBJDD=**{RESOBJ | ddname}**

Specifies the DD name 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 then writes the resource data to this DD name. The DD name is a 1- to 8-byte character string containing only those alphanumeric characters supported in the operating environment. Required DCB characteristics for the file are:

- Variable blocked format
- A maximum record length of 32756

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the **RESOBJDD** statement. If this happens, ACIF ends processing abnormally.

- A block size of 32760
- Physical, sequential format

If **RESOBJDD** is not specified, ACIF uses **RESOBJ** as the default DD name.

VSE

RESOBJDD={RESOBJ | *filename* (DEVT=TAPE | DISK)}

Specifies the file name and file characteristics that appears on the DLBL or TLBL JCL statement. The file name is a 1- to 7-byte 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

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the DLBL or TLBL JCL statement. If this happens, ACIF ends processing abnormally.

- A block size of 32760
- Assigned to programmer logical unit 008

If **RESOBJDD** is not specified, ACIF uses **RESOBJ** as the default file name and **DISK** as the default device type.

RESTYPE

Specifies the type of AFP print resources ACIF should retrieve from the resource directories or libraries for inclusion in the resource file (**RESOBJDD**). (See Table 2 on page 19 for the order that ACIF searches for AFP resources.)

RESTYPE={NONE | ALL | [FDEF][,PSEG][,OVLY][,FONT][,OBJCON][,BCOCA][,GOCA][,IOCA][,INLINE]}

The values are:

NONE

Specifies that no resource file be created.

ALL

Specifies that all resources required to print or view the output document file (specified with the **OUTPUTDD** parameter) be included in the resource file.

FDEF

Specifies that the form definition (specified with the **FORMDEF** parameter) 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.

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. Also used for TrueType and OpenType fonts and specifies that all base fonts, linked fonts, and font collections required to print the output file be included in the resource file. If

MCF2REF=CF is specified, ACIF also includes coded fonts in the resource file; otherwise, coded fonts are not included in the resource file.

Notes:

1. ACIF can only process 512 font members. If you exceed 512 font members in your document, an error message is issued. You can specify **MCF2REF=CF** to try to reduce the number of font entries required.
2. Specifying **RESTYPE=FONT** is not recommended with double-byte raster fonts because of the size and large number of library members needed to process double-byte raster fonts.
3. When specifying **RESTYPE=FONT** with TrueType and OpenType fonts, the embed flag must be set “on” to save the font in the resource file. Refer to *Using OpenType Fonts in an AFP System* for more information.
4. ACIF wraps TrueType and OpenType fonts in MO:DCA-P structured fields when it saves them in the resource file.
5. When **EXTENSIONS=RESORDER** is specified with **RESTYPE=FONT**, TrueType and OpenType fonts that were originally inline in the input file are not saved in the resource library.

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.

BCOCA

Specifies that all BCOCA objects included by an IOB structured field required to print or view the output document file be included in the resource file.

GOCA

Specifies that all GOCA objects included by an IOB structured field required to print or view the output document file be included in the resource file.

IOCA

Specifies that all IOCA objects included by an IOB structured field required to print or view the output document file be included in the resource file.

INLINE

Specifies that if there are inline resources contained in the ACIF input file (INPUTDD), those inline resources are written to the output file (OUTPUTDD) as well as to the resource file (RESOBJDD). Otherwise, if **INLINE** is not specified, inline resources in the input file are only written to the resource file. For example, **RESTYPE=FONT,PSEG,INLINE** causes any inline fonts and page segments to be written to the output file, in addition to writing all inline and library fonts and page segments to the resource file. The inline resources precede the document in the output file. See “Processing Inline Resources” on page 197 for more information.

Because multiple resource types are contained in the page segment and object container libraries, and ACIF does not enforce a prefix for the eight-character resource name, you should define a naming convention that identifies each type of resource in the library. IBM recommends these two-character prefixes:

- B1 for BCOCA objects
- E1 for encapsulated PostScript objects
- G1 for GOCA objects
- H1 for microfilm setup objects
- I1 for IOCA objects

IT for IOCA tile objects
M1 for color mapping table objects
PP for PDF single-page objects
PR for PDF resource objects
S1 for page segments

ACIF supports the specification of FDEF, FONT, OBJCON, OVLY, BCOCA, GOCA, INLINE, IOCA, and PSEG in any combination. For example, if you want to specify form definitions, page segments, and overlays as the resource types, you enter:

RESTYPE=FDEF,PSEG,OVLY

Notes:

1. When creating AFP files to view on the AFP Workbench Viewer, do not specify **RESTYPE=FONT** or **RESTYPE=ALL**. The AFP Workbench Viewer uses font definition files for font substitution instead of retrieving fonts from a resource file when displaying documents. Therefore, you do not need to download fonts to the resource file, which is time consuming and increases the number of bytes transmitted when the file is transferred to the workstation or archived.
2. 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 only page segments be saved in a resource file, and the page segments are included in overlays, the page segments are not saved in the resource file because the overlays are not searched. In this case, you need to request that both page segments and overlays be saved.
3. ACIF saves specified inline resources in the resource file, even if they are not needed to print the job. However, if you specify **EXTENSIONS=RESORDER**, ACIF saves only the inline resources actually needed to print the job. You can also use a resource exit to filter out any resources you do not want included in the resource file (see “Resource Exit” on page 121 for more information).

TRACE

z/OS

TRACE={YES | NO}

Specifies that ACIF should provide diagnostic trace information while processing the file in z/OS. **TRACE** is only valid on z/OS. The values are:

YES

ACIF uses the facilities of the z/OS and 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 needs to be started with the option **TRACE=USRP**, and subsequently modified with **USR=(314)**.

Notes:

1. If **YES** is specified and GTF is active, ACIF ends with a Return Code 4 (RC=4).
2. If **YES** is specified and GTF is not active, you receive an S0C4 abend. Therefore, do not specify **YES** unless GTF is running.

NO

ACIF does not produce diagnostic trace records

Tracing increases processor overhead and should be turned off unless you need to do problem determination.

TRC

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.

Note: TRC characters can be used to map fonts in documents that reference either TrueType and OpenType fonts or FOCA fonts, but not a combination of the two.

TRC={YES | NO}

The values are:

YES

The input file contains table reference characters.

NO

The input file does not contain table reference characters.

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 you specify **TRC=YES** but TRCs are not contained in the file, ACIF interprets the first character of each line (or second, if carriage control characters are used) as the font identifier. Consequently, the font used to process each line of the file might not be the one you expect and one byte of data is lost from each record.
3. If you specify **TRC=NO** or you do not specify **TRC** at all, but your line data contains a TRC as the first character of each line (or second if carriage control characters are used), ACIF processes the TRC as a text character in the output rather than using it as a font identifier.

TRIGGERn

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 **TRIGGERn** 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 **TRIGGERn** parameter causes ACIF to index the input file.

TRIGGERn={record | *}{,column | *}{,'triggervalue' | X'triggervalue'}

Each **TRIGGERn** parameter has three values:

*record | **

Specifies the relative record number from the indexing anchor record (**TRIGGER1**). A value of “*” *must* be specified for **TRIGGER1** and cannot be specified for any other **TRIGGERn** parameter; “*” 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 ends 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), as a '0', or as an '*'. Specifying '0' or '*' 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 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.

'triggervalue' | **X***'triggervalue'*

Specifies the actual alphanumeric 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 to 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.

Note: The trigger value can be specified as ASCII character data in AIX or Windows, EBCDIC character data in z/OS, VM or VSE, or hexadecimal data. However, if the input data file is *anything other than* ASCII in AIX or Windows or EBCDIC in z/OS, VM, or VSE, the value *must* be specified as hexadecimal data.

The following example shows how to use a carriage control character as a trigger:

```
TRIGGER1=*,1,'1'           /* Look for Skip-to-Channel 1
TRIGGER2=0,50,'ACCOUNT:'   /* Find account number
TRIGGER3=3,75,'Page 1'     /* Find page
```

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 '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 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', it begins scanning the first record after the farthest field specified. If neither 'ACCOUNT:' nor 'Page 1' is 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 **TRIGGER_n** or **FIELD_n** value appear within the page range specified by the **INDEXSTARTBY** parameter (unless **INDEXSTARTBY=0** is specified). If no **TRIGGER_n** or **FIELD_n** parameter is satisfied within the **INDEXSTARTBY** page range, ACIF stops processing and issues an error message.

2. At least one **TRIGGER n** or **FIELD n** value 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 might 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 n** parameters when the input file contains indexing tags.

See Chapter 4, “Enhanced Indexing Parameters,” on page 77 for information about using the **TRIGGER n** parameter with enhanced ACIF indexing.

UNIQUEBNGS

z/OS, VM, and VSE

Specifies whether ACIF creates a unique group name by generating an eight-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.

UNIQUEBNGS={YES | NO}

YES

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

NO

ACIF does not generate the string. Specify **NO** if you use an application such as AFP Toolbox, AFP API, or DCF to generate your own group names.

If **UNIQUEBNGS** is not specified, ACIF uses **YES** as the default unless you specify **DCFPAGENAMES=YES**, in which case ACIF uses **NO** as the default.

USERLIB

AIX and Windows

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. **USERLIB** is not used to specify directories for TrueType and OpenType fonts. Instead, use the **USERPATH** parameter.

The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

z/OS

USERLIB=*dsname1*[,*dsname2*][,*dsname3*...]

Specifies data sets containing AFP resources for processing the input data set. You can specify a maximum of eight data sets. For example:

```
USERLIB=USER.IMAGES,USER.AFP.RESOURCES
```

ACIF dynamically allocates these data sets and searches for AFP 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 ends processing.

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. Data sets 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.

VM

USERLIB=*filetype1*[,*filetype2*][,*filetype3*...]

Specifies the file types that define the libraries containing AFP resources for processing the input file. You can specify a maximum of eight file types. For example:

```
USERLIB=USER3820,TEMPUSER
```

ACIF searches for AFP resources in these file types 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 ends processing.

Note: File types must conform to CMS naming conventions.

VSE

This parameter is not used for VSE. AFP resources are located in the library defined by the **// LIBDEF PHASE,SEARCH=(...)** JCL statement. For information about how PSF/VSE selects resources, refer to *Print Services Facility/VSE: System Programming Guide*.

USERPATH

This parameter is not supported for VM and VSE; if specified, you see an error message.

AIX, Windows, and z/OS

USERPATH=*pathlist*

Specifies the names of user directories containing TrueType and OpenType fonts. TrueType and OpenType fonts are Unicode-enabled AFP fonts that are not defined by the IBM Font Object Content Architecture (FOCA).

By convention, these fonts are typically used by one user, as opposed to the system fonts (specified with the **FONTPATH** parameter) that are shared by many users. The value is:

pathlist

Any valid search path. You must use a colon (:) in AIX, a semicolon (;) in Windows, or a comma (,) in z/OS to separate multiple paths. For example:

AIX or Windows

```
acif inputdd=INFILE outputdd=OUTFILE pagedef=PAGTRUE formdef=F1A10110 \  
userpath=/jdoe/fonts/truetype:/jdoe/fonts/truetype/myfonts/
```

Note: The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

z/OS

```
INPUTDD=INFILE  
OUTPUTDD=OUTFILE  
PAGEDEF=PAGTRUE  
FORMDEF=F1A10110  
USERPATH='/jdoe/fonts/truetype','/jdoe/fonts/truetype/myfonts/'
```

ACIF searches the paths in the order in which they are specified.

For information about how Infoprint Manager for AIX or Infoprint Manager for Windows selects resources, see “Using ACIF in AIX and Windows” on page 19 or refer to *IBM Infoprint Manager: Reference*.

Chapter 4. Enhanced Indexing Parameters

Enhanced ACIF indexing is only supported for ACIF running with PSF 4.1.0 for z/OS. Enhanced indexing lets you:

- Generate page-level information so you can move to specific pages in a document.
- Define a transaction field and create indexes where ACIF extracts the first and last value from a group or page.
- Define a field that is based on where the trigger is found.
- Define a default value for a field, which is used if the record is not long enough to hold the field.
- Change to a new group when the maximum number of pages is reached.
- Define a floating trigger, which can appear multiple times in a group or not at all, or define a trigger that is found within a range of records.
- Match specific characters that might appear in a field column or match a field mask symbol.

This chapter describes the ACIF parameters that are used for enhanced indexing functions. To use enhanced indexing in ACIF, you specify the parameters in this chapter, in addition to those found in Chapter 3, “ACIF Parameters,” on page 29. The syntax rules for the enhanced indexing parameters are the same as those for z/OS in “Syntax Rules” on page 29.

Parameter Values

Table 6 lists the ACIF enhanced indexing parameters and values for PSF 4.1.0 for z/OS. Underscored values are the default and are used by ACIF if no other value is specified.

Table 6. ACIF Enhanced Indexing Parameters

Enhanced Indexing Parameters	See Page...
FIELD <i>n</i> ={ <i>record,column,length</i> ,([TRIGGER = <u>1</u> <i>n</i>] BASE =0 TRIGGER [DEFAULT = <i>value</i> X ' <i>value</i> ']}]}	78
FIELD <i>n</i> ={ <i>literalvalue</i> X ' <i>literalvalue</i> '}	79
FIELD <i>n</i> ={*,*, <i>length</i> ,(OFFSET =(<i>start1:end1</i> [, <i>start2:end2</i>][, <i>start3:end3</i> ...]), MASK ='@ # = ~ ^ %'[, ORDER = <u>BYROW</u> <u>BYCOL</u>])}]}	80
FIELD <i>n</i> ={ <i>record,column,length</i> ,(TRIGGER = <i>n</i> , BASE =0 TRIGGER , MASK ='@ # = ~ ^ %'[, DEFAULT = <i>value</i> X ' <i>value</i> ']}]}	82
GROUPMAXPAGES = <i>nnnn</i>	84
INDEX <i>n</i> ={' <i>attributename</i> ' X ' <i>attributename</i> '}{[, FIELD <i>nn</i> [, FIELD <i>nn</i> ...]][,(TYPE ={ <u>GROUP</u> [, BREAK ={ <u>YES</u> <u>NO</u>]} GROUPRANGE PAGE PAGERANGE)}]}	84
TRIGGER <i>n</i> ={ <i>record</i> *}{[, <i>column</i> *]{[, <i>triggervalue</i> ' X ' <i>triggervalue</i> '[,(TYPE ={ <u>GROUP</u> <u>GROUP</u> , RECORDRANGE =(<i>start,end</i>) <u>FLOAT</u>)}]}	87
USERMASK { <i>n</i> ,' <i>symbolvalue</i> '}{[, <i>'stringvalue'</i> ' X ' <i>stringvalue</i> '}	89

The following sections describe the enhanced indexing parameters.

FIELDn

Specifies the field that identifies the location of index data and provides default and literal (constant) index values. ACIF supports these types of fields for enhanced indexing:

Trigger field

Based on the location of a trigger string value

Constant field

Lets you provide the actual index value that is stored in the database

Transaction field

Lets you index input data that contains one or more columns of sorted data when it is not practical to store every value in the database (ACIF extracts the beginning and ending sorted values in each group)

Mask field

Must be based on a floating trigger and uses a mask to match data located in the field columns

You must define at least one field and you can define a maximum of 32 fields (**FIELD1** through **FIELD32**). When adding a field parameter, use the next available number, beginning with “1”.

Trigger Field

A trigger field is a field that is based on the location of a trigger string value.

FIELD*n*={*record,column,length*,([**TRIGGER**=1 | *n*,]**BASE**=0 | **TRIGGER**[,**DEFAULT**=*value* | X'*value*'])}

The values are:

n Specifies the field parameter identifier.

record

Specifies the relative record number from the trigger on which the field is based. This is the record number where ACIF begins to search for the field. The supported range of values are ± 0 to 255.

column

Specifies the relative column number from the BASE value. This is the column number where ACIF begins to search for the field. A value of “1” refers to the first byte in the record. For files containing carriage control characters, column one 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. If you specify **BASE=0**, the *column* value can be 1 to 32756. If you specify **BASE=TRIGGER**, *column* value can be -32756 to 32756.

Note: If the specified value exceeds the physical length of the record and you do not specify a DEFAULT value, ACIF reports an error condition and ends processing.

length

Specifies the number of contiguous bytes (characters) that compose this field. The supported range of values 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.

Note: If the field begins outside the maximum length of the record and you do not specify a DEFAULT value, ACIF reports an error condition and ends processing.

TRIGGER=n | *n*

Specifies the **TRIGGER n** parameter ACIF uses to locate the field. This is an optional parameter. Replace n with the number of a defined parameter, such as **TRIGGER2**.

BASE=0 | TRIGGER

Specifies whether ACIF uses the starting column number of the trigger string value to locate the field data. The values are:

- 0** ACIF adds zero to the field column offset. You should use **0** if the field data always starts in a specific column.

TRIGGER

ACIF adds the starting column number of the trigger string value to the field column offset. Use **TRIGGER** if the field data does not always start in a specific column, but is always offset a specific number of columns from the trigger string value. The trigger string value can begin in any column in the record. A field based on this trigger occurs in the trigger record.

The field parameter in the following example causes ACIF to locate field values that begin in column 83 of the same record that contains the **TRIGGER1** string value. The field length is eight bytes. **BASE=0** is specified because the field data always starts in the same column.

```
TRIGGER1=*,1,'1',(TYPE=GROUP)
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
```

The field parameter in the following example causes ACIF to locate field values that begin 10 columns offset from the trigger string value. Basing the field on **TRIGGER2** and specifying **BASE=TRIGGER** lets ACIF locate the field by adding 10 to the starting column offset of the trigger string value.

```
TRIGGER2=*,*, 'ACCOUNT:',(TYPE=FLOAT)
FIELD2=0,10,12,(TRIGGER=2,BASE=TRIGGER)
```

DEFAULT=*value* | X'*value*'

Specifies the default value for the index when a record is not long enough to contain the field data because the JES spool has removed trailing blanks from the end of the record. The default value can be 1 to 250 bytes in length.

Note: The value can be specified as an EBCDIC character string or hexadecimal data. However, if the input data file is *anything other than* EBCDIC, the value *must* be specified as hexadecimal data (otherwise, the comparisons between the input data file and what is coded in the **FIELD n** parameter do not yield a match).

DEFAULT is used with Download for z/OS, which can transmit data from the JES spool. JES, by default, truncates trailing blanks at the end of records. When processing the records that have been truncated, ACIF ends with an error unless you specify DEFAULT.

Constant Field

A constant field is a field for which you specify the actual index value that is stored in the database. It is possible to generate an index value by concatenating or

combining the literal value that you specify for a constant field with the value that ACIF extracts from a document by using a trigger field. However, the trigger field cannot be based on a floating trigger.

FIELD*n*={*literalvalue* | **X**'*literalvalue*'}

The values are:

n Specifies the field parameter identifier.

literalvalue | **X**'*literalvalue*'

The actual index value of the field that is stored in the database. The literal value can be 1 to 250 bytes in length. ACIF does not check the validity of the actual content of the supplied data.

Note: The value can be specified as an EBCDIC character string or hexadecimal data. However, if the input data file is *anything other than* single-byte EBCDIC, the value *must* be specified as hexadecimal data (otherwise, the comparisons between the input data file and what is coded in the **FIELD***n* parameter do not yield a match).

The field parameter in the following example causes ACIF to store the same string of characters in each INDEX3 it creates.

```
FIELD3='251658240'
INDEX3='Account Number',FIELD3,(TYPE=GROUP,BREAK=NO)
```

The field parameters in the following example cause ACIF to concatenate a literal value with the index value extracted from the data. ACIF concatenates the literal value specified in the **FIELD3** parameter to each index value located using the **FIELD4** parameter. The concatenated string value is stored in the database. In this example, the account number field in the data is 14 bytes in length. However, the account number in the database is 19 bytes in length. Use a constant field to concatenate a constant five-byte prefix (0000–) to all account numbers extracted from the data. The input data is encoded in EBCDIC.

```
FIELD3='00006'
FIELD4=0,66,14
INDEX3='Account Number',FIELD3,FIELD4,(TYPE=GROUP,BREAK=YES)
```

Transaction Field

A transaction field lets you index input data that contains one or more columns of sorted data when you cannot store every value in the database.

FIELD*n*={*,length,(**OFFSET**=(*start1:end1*[,*start2:end2*[,*start3:end3*...]),**MASK**='@# = ~ ^ %'[,**ORDER**=**BYROW** | **BYCOL**])}

The values are:

n Specifies the field parameter identifier.

* Specifies the record number where ACIF begins searching for the field. A transaction field must specify an asterisk (*), meaning ACIF searches every record in the group.

* Specifies the column number where ACIF begins searching for the field. A transaction field must specify an asterisk (*). The **OFFSET** specification determines the column or columns where ACIF locates the field.

Note: If you enter a value other than an asterisk, ACIF ignores the value. When you specify **OFFSET**, ACIF always uses the starting column numbers from **OFFSET** to determine the location of the field values.

length

Specifies the number of contiguous bytes (characters) that compose this field. The supported range of values 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.

Note: If the field begins outside the maximum length of the record, ACIF reports an error condition and ends processing.

OFFSET=(start1:end1[,start2:end2][,start3:end3...])

Specifies the location of the field value from the beginning of the record. The start is the column where the field begins. The end is the last column of field data. A maximum of eight pairs of beginning and ending offset values are allowed. Separate the pairs with a comma. When you specify OFFSET, you must also specify MASK. The implied length of OFFSET must be the same as the number of characters in MASK or ACIF will not detect a match.

MASK='* @ # = ~ ^ %'

Specifies the pattern of symbols that ACIF matches with data located in the field columns. When you specify MASK, you must also specify OFFSET. When you define a field that includes a mask, an **INDEX** parameter based on the field cannot reference any other fields. An **INDEX** parameter based on a field that includes a mask must create GROUPRANGE or PAGERANGE type indexes. Valid mask symbols include these:

- * Matches a user-defined mask; not a literal asterisk. Refer to "USERMASK" on page 89.
- @ Matches an alphabetic character.
- # Matches a numeric character.
- = Matches any character.
- ~ Matches any non-blank character.
- ^ Matches any non-blank character.
- % Matches a blank character or numeric character.

Code page 500 is the default code page for the symbols in MASK. If you specify a different code page (with the **CPGID** parameter), ACIF translates all characters in the MASK value, except the MASK symbols. ACIF then matches the input characters against the MASK value. The following example ACIF searches columns 10 through 17 for a hexadecimal C1 followed by four numeric characters (hexadecimal F0-F9), a hexadecimal 60, and two numeric characters (hexadecimal F0-F9):

```
CPGID=500  
FIELD3=*,*,8,(OFFSET=(10:17),MASK='A####-##',ORDER=BYROW)
```

ORDER=BYROW | BYCOL

Specifies where ACIF can locate the smallest value and the largest value of a group of sorted values arranged in either rows or columns on the page. For **ORDER=BYROW**, ACIF extracts the first value in the first row and the last value in the last row that match the MASK. Data with a row orientation might appear as:

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

For **ORDER=BYCOL**, ACIF extracts the first value in the first column and the last value in the last column that match the MASK. Data with a column orientation might appear as:

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

The field parameter in the following example causes ACIF to locate a 10-character numeric string that begins in column three of any record in the group. This format of the **FIELD** parameter is used to create indexes for the beginning and ending sorted values of each group.

```
FIELD4=*,*,10,(OFFSET=(3:12),MASK='#####',ORDER=BYROW)
```

Mask Field

A *mask field* is a field with a mask that is based on a floating trigger. An **INDEX** parameter that is based on the mask field cannot include any other fields and must not create GROUPRANGE or PAGERANGE type indexes.

FIELD*n*={*record,column,length,(TRIGGER=*n*,BASE=0 | TRIGGER,MASK='@ # = - ^ %'[,DEFAULT=*value* | X'*value*])*}

The values are:

n Specifies the field parameter identifier.

record

Specifies the relative record number from the trigger on which the field is based. This is the record number where ACIF begins to search for the field. The supported range of values are ± 0 to 255.

column

Specifies the relative column number from the BASE value. This is the column number where ACIF begins to search for the field. A value of "1" refers to the first byte in the record. For files containing carriage control characters, column one 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. If you specify **BASE=0**, the *column* value can be 1 to 32756. If you specify **BASE=TRIGGER**, *column* value can be -32756 to 32756.

Note: If the specified value exceeds the physical length of the record and you do not specify a DEFAULT value, ACIF reports an error condition and ends processing.

length

Specifies the number of contiguous bytes (characters) that compose this field. The supported range of values 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.

Note: If the field begins outside the maximum length of the record and you do not specify a DEFAULT value, ACIF reports an error condition and ends processing.

TRIGGER=1 | *n*

Specifies the **TRIGGER***n* parameter ACIF uses to locate the field. When using MASK, you must specify a trigger that is defined with **TYPE=FLOAT**.

BASE=0 | TRIGGER

Specifies whether ACIF uses the starting column number of the trigger string value to locate the field data. The values are:

- 0** ACIF adds zero to the field column offset. You should use **0** if the field data always starts in a specific column.

TRIGGER

ACIF adds the starting column number of the trigger string value to the field column offset. Use **TRIGGER** if the field data does not always start in a specific column, but is always offset a specific number of columns from the trigger string value. The trigger string value can begin in any column in the record. A field based on this trigger occurs in the trigger record.

The field parameter in the following example causes ACIF to locate field values that begin in column 83 of the same record that contains the **TRIGGER1** string value. The field length is eight bytes. **BASE=0** is specified because the field data always starts in the same column.

```
TRIGGER1=*,1,'1',(TYPE=GROUP)
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
```

The field parameter in the following example causes ACIF to locate field values that begin 10 columns offset from the trigger string value. Basing the field on **TRIGGER2** and specifying **BASE=TRIGGER** lets ACIF locate the field by adding 10 to the starting column offset of the trigger string value.

```
TRIGGER2=*,*, 'ACCOUNT:',(TYPE=FLOAT)
FIELD2=0,10,12,(TRIGGER=2,BASE=TRIGGER)
```

MASK='@ # = ~ ^ %'

Specifies the pattern of symbols that ACIF matches with data located in the field columns. If the data matches the MASK, ACIF selects the field. Valid mask symbols include these:

- @** Matches an alphabetic character.
- #** Matches a numeric character.
- =** Matches any character.
- ~** Matches any non-blank character.
- ^** Matches any non-blank character.
- %** Matches a blank character or numeric character.

In the following example, ACIF selects the field only if the data in the field columns contain numeric characters:

```
TRIGGER2=*,25,'SOURCE',(TYPE=FLOAT)
FIELD2=0,38,4,(TRIGGER=2,BASE=0,MASK='####',DEFAULT='4059099376')
```

DEFAULT=value | X'value'

Specifies the default value for the index when a record is not long enough to contain the field data. The default value can be 1 to 250 characters in length.

Note: The value can be specified as an EBCDIC character string or hexadecimal data. However, if the input data file is *anything other than* single-byte EBCDIC, the value *must* be specified as

hexadecimal data (otherwise, the comparisons between the input data file and what is coded in the **FIELD n** parameter do not yield a match).

DEFAULT is used with Download for z/OS, which can transmit data from the JES spool. JES, by default, truncates trailing blanks at the end of records. When processing the records that have been truncated, ACIF ends with an error unless you specify DEFAULT.

Note: The MASK is not applied to the default value.

GROUPMAXPAGES

Specifies the maximum number of pages that ACIF can put into a group.

GROUPMAXPAGES=*nnnn*

The value is:

nnnn

A one to four digit number (1–9999).

If the maximum number of pages is reached before a group index value has changed, ACIF forces a new group. If you do not specify **GROUPMAXPAGES**, ACIF does not end the current group and begin a new group until the value of one of the fields specified with the **BREAK=YES** option on the **INDEX** parameter changes.

When indexing transaction data with a GROUPRANGE index, you typically set the **GROUPMAXPAGES** parameter to control the maximum number of pages in a group. For more information about the TYPE and BREAK options, see the **INDEX** parameter in *IBM DB2 Content Manager OnDemand for Multiplatforms: Indexing Reference*.

INDEX n

Specifies the index name, the field or fields on which the index is based, and the type of index ACIF generates. You can define group indexes for AFP and line data. You can define page indexes for AFP data and line data that you convert to AFP. When you define a group index, IBM recommends that you name the index the same as the application group database field name.

Keep in mind: Group indexes are stored in the database and used to search for documents. Page indexes are stored with the document, not in the database. This means that you cannot use page indexes to search for documents. After retrieving a document, you can use the page indexes to move to a specific page in the document by using the Go To command in the client.

To generate page-level information in the output file so you can go to specific pages in a document, you must convert the input data to AFP by specifying **CONVERT=YES** and **INDEXOBJ=ALL**. You must also create an index field by specifying **TYPE=PAGE** on the **INDEX** parameter.

You must define at least one index and you can define a maximum of 32 indexes (**INDEX1** through **INDEX32**). Each index can be made up of one or more **FIELD** definitions. When adding an index parameter, use the next available number, beginning with “1”.

INDEX*n*={*attributename*' |
*X'**attributename*}[,**FIELD***nn*[,**FIELD***nn*...]][,(**TYPE**={**GROUP**[,**BREAK**={**YES** | **NO**}] |
GROUPRANGE | **PAGE** | **PAGERANGE**)}]

The values are:

n Specifies the index parameter identifier.

'attributename' | *X'**attributename'*

Specifies a user-defined label 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 actual account number (for example, 0001234567). The attribute name can be 1 to 250 bytes in length.

Note: The value can be specified as an EBCDIC character string or hexadecimal data. However, if the input data file is *anything other than* single-byte EBCDIC, the value *must* be specified as hexadecimal data (otherwise, the comparisons between the input data file and what is coded in the **INDEX***n* parameter do not yield a match).

FIELD*nn*[,**FIELD***nn*...]

Specifies one or more **FIELD***n* parameters that ACIF uses to locate the index. A maximum of 32 **FIELD***n* parameters can be specified for each index. 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 ends processing if this occurs.

GROUPRANGE and **PAGERANGE** indexes must name one and only one transaction field. **PAGE** indexes must name fields based on floating triggers. An index that names a field based on a floating trigger must be **TYPE=GROUP,BREAK=NO** or **TYPE=PAGE**.

TYPE={**GROUP**[,**BREAK**={**YES** | **NO**}] | **GROUPRANGE** | **PAGE** | **PAGERANGE**}

The type of index ACIF generates. You can define either group or page indexes for AFP and line data. The types are:

GROUP[,**BREAK**={**YES** | **NO**}]

The values are:

GROUP

Specifies a group index value. ACIF creates one index value for each group. A group index that names a field parameter based on a floating trigger must specify **BREAK=NO**.

BREAK={**YES** | **NO**}

Specifies whether ACIF includes the index when calculating a group break. The values are:

YES

ACIF breaks the group when the index value changes.

NO

ACIF does not break the group.

The following example indicates that ACIF generates group indexes for date index values. The input data is encoded in EBCDIC.

```
INDEX1='Date Due',FIELD1,(TYPE=GROUP,BREAK=YES)
```

The next example indicates that ACIF generates group indexes for customer name and account number index values. The input data is encoded in EBCDIC. ACIF closes the current group and begins a new group only when the customer name index value changes (the data is sorted by customer name). In this example, a customer might have one or more statements with different account numbers. The page numbers in each statement begin with the number one, giving the appearance of unique statements. The goal is to collect all of a customer's statements in a single group.

```
INDEX1='Customer Name',FIELD1,(TYPE=GROUP,BREAK=YES)
INDEX2='Account Number',FIELD2,(TYPE=GROUP,BREAK=NO)
```

GROUPRANGE

Specifies a GROUPRANGE index, which does not break the group. ACIF creates index values for the first and last sorted values in each group. ACIF creates indexes for the group by extracting the first and last values that match the MASK of the transaction field on which the index is based. ACIF assumes that the input values are sorted. You can define one GROUPRANGE index per report.

A GROUPRANGE index must name one and only one transaction field, cannot name a field parameter that is based on a floating trigger, and cannot break a group.

For a GROUPRANGE index, ACIF can use the value of the **GROUPMAXPAGES** parameter to determine the number of pages in a group and when to break a group. For example, you need to index a line data report that consists of thousands of pages of sorted transaction data. You define a GROUP index to hold the report date index values and a GROUPRANGE index to hold the transaction numbers for each group. Because every page in the report contains the same date, the GROUP index cannot be used to break the report into groups. (And a GROUPRANGE index cannot be used to break a group.) To break the report into groups, set the GROUPMAXPAGES parameter to the maximum number of pages you want in a group (for example, 100). When calculating group breaks, ACIF will use the value of the **GROUPMAXPAGES** parameter to determine when to close the current group and begin a new group.

The following example indicates that ACIF generates GROUPRANGE indexes for loan number index values. ACIF extracts the beginning and ending loan numbers in each group of pages. The input data is encoded in EBCDIC. Because a GROUPRANGE index cannot be used to break a report into groups of page, the **GROUPMAXPAGES** parameter can be used to determine the number of pages in a group. ACIF closes the current group and begins a new group when the number of pages in the group is equal to the value of the **GROUPMAXPAGES** parameter.

```
INDEX2='Loan Number',FIELD2,(TYPE=GROUPRANGE)
GROUPMAXPAGES=100
```

PAGE

Specifies a page index, which does not break a group. You can create

more than one page index per page. Page indexes must name fields based on floating triggers, and cannot break a group.

Page indexes are stored with the document, not in the database, and cannot be used to search for documents. After retrieving a document, you can use the page indexes to move to a specific page in the document by using the Go To command in the client.

To generate page-level information in the output file so you can go to specific pages in a document, you must create an index field by specifying a page index with **INDEXOBJ=ALL**; otherwise, ACIF does not write the page index data to the index object file.

The following example indicates that ACIF generates PAGE indexes for subtotal values (the attribute name that appears in the Go To dialog box is Subtotal). The input data is encoded in EBCDIC. ACIF extracts the index values from each page.

```
INDEX3='Subtotal',FIELD3,(TYPE=PAGE)
```

PAGERANGE

Specifies a PAGERANGE index, which does not break a group. ACIF creates index values for the first and last sorted values on each page. ACIF creates indexes for the page by extracting the first and last values that match the MASK of the transaction field on which the index is based. ACIF assumes that the input values are sorted. You can define one PAGERANGE index per report.

A PAGERANGE index must name one and only one transaction field, cannot name a field parameter that is based on a floating trigger, and cannot break a group.

PAGERANGE indexes are stored with the document, not in the database, and cannot be used to search for documents. After retrieving a document, you can use the page indexes to move to a specific page in the document with the Go To command in the client.

To generate page-level information in the output file so you can go to specific pages in a document, you must create an index field by specifying a PAGERANGE index with **INDEXOBJ=ALL**; otherwise, ACIF does not write the PAGERANGE index data to the index object file.

TRIGGERn

Specifies locations and values required to uniquely identify the beginning of a group and the locations and values of data fields used to define indexes. These data fields are referred to as “triggers” because their presence in the file triggers a processing action. You must specify at least one **TRIGGERn** parameter and you can specify a maximum of eight parameters. When adding a trigger parameter, use the next available number, beginning with “1”.

```
TRIGGERn={record | *}{,column | *}{,'triggervalue' |  
X'triggervalue'}[, (TYPE={GROUP | GROUP,RECORDRANGE=(start,end) |  
FLOAT})]
```

The values are:

n Specifies the trigger parameter identifier.

record | *

Specifies the input record where ACIF locates the trigger value. You *must* specify an asterisk value (*) for **TRIGGER1**, record range triggers and float

triggers so that ACIF searches every input record for the trigger value. For other group triggers, the input record is relative to the record that contains the **TRIGGER1** value. The supported range of record numbers is 0 to 255.

column | *

Specifies the byte offset from the beginning of the record where the trigger value is located. The supported range of column values is 0 to 32756. You can specify an asterisk (*) or zero (0) so that ACIF scans the record from left to right looking for the trigger value. A value of "1" refers to the first byte in the record.

'triggervalue' | *X'triggervalue'*

Specifies the actual alphanumeric or hexadecimal value of the trigger that ACIF uses to match the input data. The trigger value can be 1 to 250 bytes in length and is case sensitive.

Note: The trigger value can be specified as EBCDIC character data or hexadecimal data. However, if the input data file is *anything other than* EBCDIC, the value *must* be specified as hexadecimal data.

TYPE={GROUP | GROUP,RECORDRANGE=(start,end) | FLOAT

The trigger type. **TRIGGER1** must be a GROUP trigger. The types are:

GROUP

Specifies the beginning of a group. In ACIF, a group is a named collection of sequential pages that form a logical subset of an input file. You define only as many group triggers as needed to identify the beginning of a group. In many cases, you only need one group trigger.

A group must contain at least one page, and it can contain all of the pages in an input file. However, most customers define their group triggers so that ACIF can logically divide an input file into smaller parts, such as by statement, policy, bill, or, for transaction data, number of pages.

A group is determined when the value of an index changes (for example, account number) or when the maximum number of pages for a group is reached. ACIF generates indexes for each group in the input file. Because a group cannot be smaller than one page, a group trigger should not appear more than once on a page. See the **BREAK** option on the **INDEX** parameter (page 85) for more information about breaking groups.

The following example indicates that ACIF searches column one of every input record for the occurrence of a skip-to-channel 1 carriage control character. The record value for **TRIGGER1** must be an asterisk (*) and **TRIGGER1** must be a GROUP trigger. The input data is encoded in EBCDIC.

```
TRIGGER1=*,1,'1',(TYPE=GROUP)
```

The next example indicates that ACIF matches the PAGE 1 value beginning in column two of every input record. The record value for **TRIGGER1** must be an asterisk (*) and **TRIGGER1** must be a GROUP trigger. The input data is encoded in EBCDIC. .

```
TRIGGER1=*,2,'PAGE 1',(TYPE=GROUP)
```

The final example indicates that ACIF matches the "Account Number" value beginning in column fifty of the sixth input record following the **TRIGGER1** record. The input data is encoded in EBCDIC.

TRIGGER2=6,50,'Account Number',(TYPE=GROUP)

GROUP,RECORDRANGE=(start,end)

Specifies field data that is not always located in the same record relative to **TRIGGER1**. ACIF determines the location of the field by searching the specified range of records. The range can be from 0 to 255. ACIF stops searching after the first match in the specified range of records. For example, if the range is **5,7** and records six and seven contain the trigger value, ACIF stops searching after matching the value in record six.

The following example indicates that ACIF locates the "Account Number" value beginning in column fifty within a range of records (the trigger value can occur in records six, seven, or eight following **TRIGGER1**) in each group. You must specify an asterisk (*) for record number because ACIF uses the record range to determine which records to search for the trigger value. The input data is encoded in EBCDIC.

TRIGGER2=*,50,'Account Number',(TYPE=GROUP,RECORDRANGE=(6,8))

FLOAT

Specifies field data that does not necessarily occur in the same location on each page, the same page in each group, or in each group. ACIF determines the location of the field by searching every input record for the trigger value starting in the specified column (or every column, if an asterisk is specified). For example, you need to index statements by type of account. Possible types of accounts include savings, checking, loan, IRA, and so forth. Not all statements contain all types of accounts. This causes the number of pages in a statement to vary and the page number where a specific type of account occurs to vary. However, each type of account is preceded by the string "Account Type". Define a float trigger with a trigger string value of "Account Type". The same float trigger can be used to locate all of the accounts that occur in a statement.

The following example indicates that ACIF matches the "Type of Income" value, beginning in column five of every record in the group. You must specify an asterisk (*) for the record number. The input data is encoded in EBCDIC.

TRIGGER3=*,5,'Type of Income',(TYPE=FLOAT)

Notes:

1. ACIF requires that at least one **TRIGGER_n** or **FIELD_n** value appear within the page range specified by the **INDEXSTARTBY** parameter (unless **INDEXSTARTBY=0** is specified). If no **TRIGGER_n** or **FIELD_n** parameter is satisfied within the **INDEXSTARTBY** page range, ACIF stops processing and issues an error message.
2. At least one **TRIGGER_n** or **FIELD_n** value 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 might 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_n** parameters when the input file contains indexing tags.

USERMASK

Specifies a user mask that identifies a symbol and string for matching field data.

USERMASK={*n*, 'symbolvalue'[, 'stringvalue' | **X**'stringvalue']}

The values are:

n Indicates the number of the user mask. Valid values are 1 to 4.

'symbolvalue'

Indicates a character that represent the characters in the *stringvalue* or the field mask. All printable characters except # @ = ~ ^ % are valid. The *symbolvalue* does not match its literal value in the field data. For example, ACIF does not match an asterisk (*) symbol with an * in the field data.

'stringvalue' | **X**'stringvalue'

Indicates one or more characters you want to match to field data. When the input data file is *anything other than* single-byte EBCDIC, the character string must be specified in hexadecimal.

The following example shows how to use **USERMASK** to match specific characters that might appear in the field column:

```
USERMASK=1, '*', 'AaBbCc'
```

```
FIELD3=*,*,15,(OFFSET=(10:24),MASK='*@@@@@@@@@@@@@@@@',ORDER=BYROW)
```

In this example, **USERMASK** causes ACIF to match an uppercase or lowercase A, B, or C in the first position of a 15-character string, such as a name.

A user mask can also match one of the field mask symbols. ACIF reserves the symbols # @ = ~ ^ % for the field mask. If the field data contains one of the mask symbols, you must define a user mask so that ACIF can find a match. For example,

```
USERMASK=2, '*', '%'
```

```
FIELD4=*,*,3,(OFFSET=(10:12),MASK='###',ORDER=BYROW)
```

In this example, ACIF matches a three-character string that contains two numerics and the percent sign (%), such as 85%.

Chapter 5. Examples of Using ACIF

This chapter shows examples of how to use ACIF processing parameters for transforming data, retrieving resources, specifying fonts, identifying the location of resource libraries, and drawing graphics with record format page definitions. Detailed examples of how to use ACIF for viewing and indexing a document are also described.

For examples of using ACIF or the **line2afp** command to transform line data and unformatted ASCII files for printing with Infoprint Manager for AIX or Infoprint Manager for Windows, refer to *IBM Infoprint Manager: Reference*.

Examples of Using ACIF Processing Parameters

This section shows how you can use ACIF processing parameters to:

- Transform line data or XML data into a MO:DCA-P document.
- Retrieve resources.
- Specify AFP coded fonts.
- Specify TrueType and OpenType fonts.
- Identify the location of resource libraries.
- Draw graphics with record format page definitions.

Note: In the following AIX and Windows examples, ACIF is run by entering the **acif** command, parameters, and values on the command line. In AIX, when all of the parameters do not fit on a single line across the screen, the backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

Transforming Line Data or XML Data into a MO:DCA-P Document

You have an EBCDIC line data file named **OLDFILE.1403** or an XML data file named **OLDFILE.xml** that you want to transform into a MO:DCA-P document named **NEWFILE.afp**. To do this for line data or XML data, enter these parameters for your operating system:

AIX or Windows

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

Notes:

1. For XML data, use **OLDFILE.xml** for **inputdd**.
2. The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

z/OS, VM, or VSE

```
INPUTDD=OLDFILE  
OUTPUTDD=NEWFILE  
CCTYPE=A  
PAGEDEF=P1A06462  
FORMDEF=F1A10110
```

Note: For XML data, **CCTYPE** is not used.

ACIF converts the line data or XML data file, pointed to by the **INPUTDD** parameter, into a document file pointed to by the **OUTPUTDD** parameter.

For line data, you specify **CCTYPE=A** to indicate that the file contains EBCDIC ANSI carriage control characters. This particular input file is in variable length record format, so in AIX and Windows you indicate this by specifying **FILEFORMAT=RECORD**. The **PAGEDEF** and **FORMDEF** parameters are required with your line-data input file, so you specify the file names of the page definition and form definition you want ACIF to use in processing this file.

Retrieving Resources

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 these parameters for your operating system:

AIX or Windows

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

Note: The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

z/OS, VM, or VSE

```
INPUTDD=MYFILE  
OUTPUTDD=MYDATA  
RESOBJDD=MYRES  
RESTYPE=PSEG,OVLY,FDEF  
FORMDEF=F1H10110
```

From this job, ACIF produces an AFP document file and a resource file. The AFP document file (**MYDATA**) contains the AFP data from **MYFILE**. The resource file (**MYRES**) contains the resource data from **MYFILE**.

You specify **RESTYPE=PSEG,OVLY,FDEF** so that the page segments and overlays are 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 using ACIF's resource retrieval functions, see Retrieving Resources on page 9.

Specifying AFP Coded Fonts

You have an input file (**MYFILE.asc**) that contains unformatted ASCII data, and you want these three AFP coded fonts to be used in processing the file: Helvetica 10-point, Times New Roman 10-point, and Courier 10-point. (To use any other ASCII coded fonts, you must first create them.) You are using a page definition supplied with PSF or Infoprint Manager (P1A06462), and the page definition does not name any fonts.

You specify the font names with the **CHARS** parameter. To use fonts with the appropriate ASCII code points for your unformatted ASCII input, see Table 7 on page 93, which shows the IBM Core Interchange Font names and their corresponding short names for each of the fonts you want to use: Helvetica 10-point, Times New Roman 10-point, and Courier 10-point. Because the **CHARS** parameter limits the specification of a font name to four characters, you use the

corresponding short name from the table for each of the three fonts.

Table 7. Font Short Names to Use with CHARS Parameter

Font Type	Coded Font Name	Short Name
Helvetica 10-point	X0H23002	H350
Times New Roman 10-point	X0N23002	N350
Courier 10-point	X0423002	4350

Because table reference characters are required in the input file when you want the file to print with more than one font, you specify **TRC=YES**.

You specify your input and output file names with the **INPUTDD** and **OUTPUTDD** parameters. You specify the page definition and form definition you want ACIF to use when processing the file with the **PAGEDEF** and **FORMDEF** parameters.

To use the three fonts, enter these parameters for your operating system:

AIX or Windows

```
acif inputdd=MYFILE.asc outputdd=MYFILE.afp chars=H350,N350,4350 \  
trc=yes pagedef=P1A06462 formdef=F1A10110
```

Note: The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

z/OS, VM, or VSE

```
INPUTDD=MYFILE  
OUTPUTDD=MYFILE  
CHARS=H350,N350,4350  
TRC=YES  
PAGEDEF=P1A06462  
FORMDEF=F1A10110
```

Specifying TrueType and OpenType Fonts

You have an input file (**INFILE**) that contains EBCDIC line data. You also have Unicode-enabled TrueType and OpenType fonts that reside in user path libraries, such as **/jdoe/fonts/truetype**, or system path libraries, such as **/u/fonts/truetype**. TrueType and OpenType fonts are those fonts that are not defined by the IBM Font Object Content Architecture (FOCA).

You want these TrueType and OpenType fonts to be used in processing the file: Arial Black, Century Gothic, and Times New Roman. The page definition (**PAGTRUE**) you are using references the fonts in Map Data Resource (MDR) structured fields. Refer to *Page Printer Formatting Aid: User's Guide* for information about creating page definitions that use TrueType and OpenType fonts.

You specify your input and output file names with the **INPUTDD** and **OUTPUTDD** parameters. You specify the page definition and form definition you want ACIF to use when processing the file with the **PAGEDEF** and **FORMDEF** parameters. You specify font path libraries with either the **USERPATH** parameter or the **FONTPATH** parameter.

To specify the fonts, enter these parameters for your operating system:

AIX or Windows

```
acif inputdd=INFILE outputdd=OUTFILE pagedef=PAGTRUE formdef=F1A10110 \  
userpath=/jdoe/fonts/truetype:/jdoe/fonts/truetype/myfonts/ \  
fontpath=/u/fonts/truetype:/u/fonts/truetype/local
```

Note: The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line. You must use a colon (:) in AIX or a semicolon (;) in Windows to separate multiple paths.

z/OS

```
INPUTDD=INFILE  
OUTPUTDD=OUTFILE  
PAGEDEF=PAGTRUE  
FORMDEF=F1A10110  
USERPATH='/jdoe/fonts/truetype','/jdoe/fonts/truetype/myfonts/'  
FONTPATH='/u/fonts/truetype','/u/fonts/truetype/local/'
```

Identifying the Location of Resource Libraries

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 library at your location (**SYS1.PSEGLIB**, **/usr/site/resdir**, or **\directory\site\resdir**, where *directory* is the installation directory). To be sure that ACIF finds the resources you want to use, enter these parameters for your operating system:

AIX

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

Windows

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

Note: The command parameters must be on one continuous line.

z/OS or VM

```
INPUTDD=INFILE  
OUTPUTDD=OUTFILE  
PAGEDEF=PAGED6B  
FORMDEF=FORMD1A  
USERLIB=USER.ART1,USER.ART2  
PDEFLIB=USER.PDEFDIR3
```

VSE

```
INPUTDD=INFILE  
OUTPUTDD=OUTFILE  
PAGEDEF=PAGED6B  
FORMDEF=FORMD1A
```

Note: VSE resources are located in the library defined by the **// LIBDEF PHASE, SEARCH=(...)** JCL statement.

The page definition you want to use (**PAGED6B**) is stored in one of the several page definition libraries used by your department (**USER.PDEFDIR3**, **/usr/dept/pdefdir3**, or **\install_directory\dept\pdefdir3**). The page definition is a

copy of one with the same file name that is stored in the site's general resource library, with some modifications made for use by your department.

Your page segments are stored in two other libraries that you have set up for your own use (**USER.ART1**, **/usr/mystuff/art1**, or **\install_directory\mystuffart1** and **USER.ART2**, **/usr/mystuff/art2**, or **\install_directory\mystuffart2**).

Because ACIF always searches the path specified by the **USERLIB** parameter first, your page segments are found in your personal libraries. ACIF next searches the paths specified by the parameters for specific resource libraries (**PDEFLIB**, **FDEFLIB**, and so forth), so ACIF then finds the page definition you want to use from the department's library. In AIX or Windows, ACIF then searches the path specified with the **RESLIB** parameter, finding your form definition and your overlay. See "Selecting Resources" on page 19 for a complete list of the search order for AFP resources.

ACIF does *not* use the page definition named **PAGED6B** that is stored in **USER.RESDIR**, **/usr/site/resdir**, or **\install_directory\site\resdir**, because it already finds the modified **PAGED6B** in the department library specified with the **PDEFLIB** parameter.

Drawing Graphics with Record Format Page Definitions

You have a page definition that you are using to format record format line data. The page definition contains these DRAWGRAPHIC commands to draw colored lines and boxes:

```
DRAWGRAPHIC BOX BOXSIZE 2.6 IN .25 IN ROUNDED LARGE
  LINETYPE SOLID COLOR Green
  FILL ALL SOLID COLOR Blue
DRAWGRAPHIC LINE ACROSS 7.5 IN
  LINEWT BOLD
  LINETYPE SOLID COLOR Red
```

To draw the graphics, enter these options on the EXTENSIONS parameter for your operating system:

AIX or Windows

```
acif inputdd=INFILE outputdd=OUTFILE fileformat=record \
extensions=prcolor,box,fracline pagedef=PAGERFLD formdef=FORMRFLD
```

Note: The backslash (\) tells AIX to continue reading the command from the next line. In Windows, the backslash is not valid; therefore, the command parameters must be on one continuous line.

z/OS, VM, or VSE

```
INPUTDD=INFILE
OUTPUTDD=OUTFILE
EXTENSIONS=PRCOLOR,BOX,FRACLINE
PAGEDEF=PAGERFLD
FORMDEF=FORMRFLD
```

Example of Using ACIF to View and Index Documents

A communications company produces monthly telephone bills with a line data application. The company wants to make the billing application output available so that when a customer calls with a billing inquiry, the customer service representatives can view the bill in the same format on their workstations as the

customer's printed copy. An example of the customer's printed telephone bill is shown in Figure 12.

Make check payable to



1 BASIC SERVICE \$30.56
2 LONG DISTANCE CHARGES \$26.41

TOTAL \$56.97

Return this portion with your payment.

WILLIAM R. SMITH
5280 SUNSHINE CANYON DR
BOULDER CO 80000-0000

TOTAL AMOUNT DUE: \$56.97
DATE DUE: JAN 29, 2000



BOULDER CO 80000-0000

BILL DATE: JAN 11, 2000
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, 2000
			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 12. Example of a Customer's Printed Telephone Bill

To meet the communications company's needs, you can use ACIF to:

- Convert the output from the line data application into a document format that can be used with the AFP Workbench Viewer.

- Index the file to facilitate searching the file with AFP Workbench Viewer.
- Retrieve resources so that all resources used in the bills are available at the workstation.

The following sections describe the tasks you do to view and index a telephone bill with ACIF:

1. Examine the input file to determine what ACIF parameters are needed to view the telephone bill and whether literal values are expressed as character data strings or hexadecimal strings. See “Examining the Input File.”
2. Specify ACIF parameters in AIX, Windows, z/OS, VM, or VSE. See “Specifying ACIF Processing Parameters” on page 100.
3. Index the input data file for data retrieval. See “Indexing Data in the Input File” on page 104.
4. Identify the locations of the resources used when the bill is printed. See “Identifying the Locations of the Resources” on page 106.
5. Determine the form definition and page definition needed to format and print the bill. See “Determining the Form Definition and the Page Definition” on page 106.
6. Run the ACIF job to create the output files. See “Running the ACIF Job” on page 106.
7. Concatenate the output files. See “Concatenating ACIF Output Files” on page 107.
8. Access the document file from a workstation for viewing with AFP Workbench Viewer. See “Accessing the Document File for Viewing” on page 108.

Examining the Input File

Figure 13 on page 98 shows the line data file currently used to print the telephone bill shown in Figure 12 on page 96.

Note: The line-data input file provided is hypothetical; it is intended only to help you understand how ACIF can 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.

Figure 13. Example of the Line Data Telephone Bill

The way literal values in the input file are defined in ACIF parameters depends on whether the input file contains ASCII or EBCDIC data. If the input file is in ASCII for AIX or Windows or in EBCDIC for z/OS, VM, or VSE, the literal values in the **FIELD_n**, **INDEX_n**, and **TRIGGER_n** parameters can be expressed in character data strings. For example, Figure 14 on page 99 shows part of an AIX parameter file for ASCII input data. The **CCTYPE** parameter value matches the type of data in the input file, in this case ASCII. The **CPGID** parameter indicates a code page for the type of data in the input file. The **FIELD_n**, **INDEX_n**, and **TRIGGER_n** parameters are expressed in character data strings because the input file is ASCII and the operating system is AIX.

```

/* Example phone bill */
/* DATA CHARACTERISTICS*/
CC=yes
CCTYPE=z
CHARS=42B2
CPGID=850

/* FIELD AND INDEX DEFINITION*/
FIELD1=13,66,15
FIELD2=0,50,30
FIELD3=1,50,30
FIELD4=2,50,30
FIELD5='1'
INDEX1='Account Number',FIELD1
INDEX2='Name',FIELD2
INDEX3='Address',FIELD3
INDEX4='City, State, Zip',FIELD4
INDEX5='Date Due',FIELD5

/* EXIT AND TRIGGER INFORMATION*/
TRIGGER1=*,1,'1'
TRIGGER2=13,50,'ACCOUNT NUMBER'

/* Carriage control used */
/* ASCII ANSI carriage controls */
/* Coded font */
/* Code page identifier */
/* Account Number data field */
/* Name data field */
/* Address data field */
/* City, State, Zip data field */
/* Date Due data field */
/* 1st index attribute */
/* 2nd index attribute */
/* 3rd index attribute */
/* 4th index attribute */
/* 5th index attribute */
/* 1st trigger */
/* 2nd trigger */

```

Figure 14. Example of an AIX Parameter File for ASCII Input Data

If the input data file is *not* ASCII in AIX or Windows or *not* EBCDIC in z/OS, VM, or VSE, the literal values in the **FIELD_n**, **INDEX_n**, and **TRIGGER_n** parameters must be expressed in hexadecimal strings. For example, Figure 15 shows part of an AIX parameter file for EBCDIC input data. The **CCTYPE** parameter value matches the type of data in the input file, in this case EBCDIC. The **CPGID** parameter indicates a code page for the type of data in the input file. The **FIELD_n**, **INDEX_n**, and **TRIGGER_n** parameters are expressed in hexadecimal strings because the input file is EBCDIC and the operating system is AIX.

```

/* Example phone bill */
/* DATA CHARACTERISTICS*/
CC=yes
CCTYPE=a
CHARS=GT15
CPGID=037

/* FIELD AND INDEX DEFINITION*/
FIELD1=13,66,15
FIELD2=0,50,30
FIELD3=1,50,30
FIELD4=2,50,30
FIELD5=X'0001'
INDEX1=X'C1838396A495A340D5A494828599',FIELD1
INDEX2=X'D5819485',FIELD2
INDEX3=X'C184849985A2A2',FIELD3
INDEX4=X'C389A3A86B40E2A381A3856B40E98997',FIELD4
INDEX5=X'C481A38540C4A485',FIELD5

/* EXIT AND TRIGGER INFORMATION*/
TRIGGER1=*,1,X'F1'
TRIGGER2=13,50,X'C1C3C3D6E4D5E340D5E4D4C2C5D9'

/* Carriage control used */
/* EBCDIC ANSI carriage controls */
/* Coded font */
/* Code page identifier */
/* Account Number data field */
/* Name data field */
/* Address data field */
/* City, State, Zip data field */
/* Date Due data field */
/* 1st index attr (Account Number) */
/* 2nd index attr (Name) */
/* 3rd index attr (Address) */
/* 4th index attr (City, State, Zip) */
/* 5th index attr (Date Due) */
/* 1st trigger (1) */
/* 2nd trigger (ACCOUNT NUMBER) */

```

Figure 15. Example of an AIX Parameter File for EBCDIC Input Data

Using the Shell with EBCDIC Literal Values in AIX or Windows

In AIX and Windows, literal values used in the **FIELD_n**, **INDEX_n**, and **TRIGGER_n** parameters must be expressed in hexadecimal strings when the input data is anything other than ASCII. Because the input data in Figure 15 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 or Windows 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:
 - With the **axeb** command, enter:
`attr2=$(echo -n "Name" | axeb)`
 - With the **iconv** command, enter:
`attr2=$(echo -n "Name" | iconv -fIBM-850 -tIBM-037)`
2. 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 Commands Reference*, SBOF-1851.

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.

Specifying ACIF Processing Parameters

You can process the ACIF parameters that are needed to produce the telephone bill by using one of these methods:

- Create and specify a parameter file.
- In AIX and Windows, enter the **acif** command, parameters, and values on the command line or in a shell script.

The following sections show examples of AIX, Windows, z/OS, VM, and VSE parameter files for the telephone bill.

AIX and Windows Parameter File

An AIX parameter file is shown in Figure 16 on page 101. A Windows parameter file is the same as the AIX parameter file except for these directories:

```
FDEFLIB=\d:\res\fdeflib1;\d:\res\fdeflib2
FONTLIB=\d:\res\fontlib1;\d:\res\fontlib2
OBJCONLIB=\d:\res\objconlib1;\d:\res\objconlib2
OVLYLIB=\d:\res\ovlylib1;\d:\res\ovlylib2
PDEFLIB=\d:\res\pdeflib1;\d:\res\pdeflib2
PSEGLIB=\d:\res\pseglib1;\d:\res\pseglib2
INPUTDD=\d:\data\INFILE
```

Also note that in Windows you use a semicolon (;) instead of a colon (:) to separate libraries.

```

/* Example phone bill */
/* DATA CHARACTERISTICS */
CC=yes /* Carriage control used */
CCTYPE=z /* ASCII ANSI carriage controls */
CHARS=42B2 /* Coded font */
CPGID=850 /* Code page identifier */

/* FIELD AND INDEX DEFINITION */
FIELD1=13,66,15 /* Account Number data field */
FIELD2=0,50,30 /* Name data field */
FIELD3=1,50,30 /* Address data field */
FIELD4=2,50,30 /* City, State, Zip data field */
FIELD5=4,60,12 /* Date Due data field */
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 */

/* INDEXING INFORMATION */
INDEXOBJ=all /* Index object file entries */

/* RESOURCE INFORMATION */
FORMDEF=F1A10110 /* Formdef name */
PAGEDEF=P1A08682 /* Pagedef name */
FDEFLIB=/usr/res/fdeflib1:/usr/res/fdeflib2 /* Formdef directories */
FONTLIB=/usr/res/fontlib1:/usr/res/fontlib2 /* Font directories */
OBJCONLIB=/usr/res/objconlib1:/usr/res/objconlib2 /* Objcon directories */
OVLVLIB=/usr/res/ovlylib1:/usr/res/ovlylib2 /* Overlay directories */
PDEFLIB=/usr/res/pdeflib1:/usr/res/pdeflib2 /* Pagedef directories */
PSEGLIB=/usr/res/pseglib1:/usr/res/pseglib2 /* Pseg directories */
RESOBJDD=RESDATA /* Resource file name */
RESTYPE=fdef,pseg,ovly /* Resource type selection */

/* FILE INFORMATION */
INDEXDD=INDEXOBJ /* Index file name */
INPUTDD=/usr/data/INFILE /* Input path & file name */
OUTPUTDD=OUTDOC /* Output file name */
MSGDD=acif.msg /* Error message file name */

/* EXIT AND TRIGGER INFORMATION */
TRIGGER1=*,1,'1' /* 1st trigger */
TRIGGER2=13,50,'ACCOUNT NUMBER' /* 2nd trigger */

```

Figure 16. Example of an AIX Parameter File

z/OS Parameter File

The JCL for a z/OS parameter file is shown in Figure 17 on page 102. This example creates a sequential data set; however, if you need a partitioned data set, change the parameters as follows:

- Set **RESFILE=PDS**
- Set the **SPACE** and **DSORG** parameters in the DD statement of the data set named by the **RESOBJDD** parameter to
SPACE=(12288,(150,15,15)),DSORG=PO.

Failure to set these parameters as described might produce a **RESOBJDD** data set that is unusable.

```

//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 *

                                /* Example phone bill */
                                /* DATA CHARACTERISTICS */
CC = YES                        /* Carriage control used */
CCTYPE = A                      /* Carriage control type */
CHARS = GT15
CPGID = 500                      /* Code page identifier */
                                /* FIELD AND INDEX DEFINITION */
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 */
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 */
                                /* INDEXING INFORMATION */
INDEXOBJ = ALL

                                /* RESOURCE INFORMATION */
FORMDEF = F1A10110              /* Formdef name */
PAGEDEF = P1A08682              /* Pagedef name */
FDEFLIB = SYS1.FDEFLIB
FONTLIB = SYS1.FONTLIBB,SYS1.FONTLIBB.EXTRA
OBJCONLIB = USER.PSEGLIB
OVLVLIB = SYS1.OVERLIB
PDEFLIB = SYS1.PDEFLIB
PSEGLIB = SYS1.PSEGLIB
RESFILE = SEQ                   /* Resource file type */
RESTYPE = FDEF,PSEG,OVLV        /* Resource type selection */
                                /* FILE INFORMATION */
INDEXDD = INDEX                 /* Index file ddname */
INPUTDD = INPUT                 /* Input file ddname */
OUTPUTDD = OUTPUT               /* Output file ddname */
RESOBJDD = RESLIB               /* Resource file ddname */
                                /* EXIT AND TRIGGER INFORMATION */
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=(32760,(150,150),RLSE),UNIT=SYSDA,
//      DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VBM,DSORG=PS)
//SYSPRINT DD DSN=APKACIF.SYSPRINT,DISP=(NEW,CATLG),
//      SPACE=(9044,(15,15),RLSE),UNIT=SYSDA,
//      DCB=(LRECL=137,BLKSIZE=1374,RECFM=VBA,DSORG=PS)

```

Figure 17. Example of a z/OS Parameter File

VM Parameter File

The CMS commands for a VM parameter file are shown in Figure 18.

```
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:

/* Example phone bill */
/* DATA CHARACTERISTICS */
CC = YES /* Carriage control used */
CCTYPE = A /* Carriage control type */
CHARS = GT15
CPGID = 500 /* Code page identifier */
FDEFLIB = FDEF3820,FDEF38PP
/* INDEXING INFORMATION */
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 */
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 */
/* FILE INFORMATION */
INDEXDD = INDEX /* Index file ddname */
INPUTDD = INPUT /* Input file ddname */
OUTPUTDD = OUTPUT /* Output file ddname */
RESOBJDD = RESLIB /* Resource file ddname */
/* RESOURCE INFORMATION */
FORMDEF = F1A10110 /* Formdef name */
PAGEDEF = P1A08682 /* Pagedef name */
FONTLIB = FONT3820,Font38PP
OBJCONLIB = OBJC3820
OVLYLIB = OVLY3820,OVLY38PP
PDEFLIB = PDEF3820,PDEF38PP
PSEGLIB = PSEG3820,PSEG38PP
RESFILE = SEQ /* Resource file type */
RESTYPE = FDEF,PSEG,OVLY /* Resource type selection */
/* EXIT AND TRIGGER INFORMATION */
TRIGGER1 = *,1,'1' /* 1st TRIGGER */
TRIGGER2 = 13,50,'ACCOUNT NUMBER:' /* 2nd TRIGGER */
```

Figure 18. Example of a VM Parameter File

VSE Parameter File

The JCL for a VSE parameter file is shown in Figure 19 on page 104.

```

// 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

/* DATA CHARACTERISTICS */
CC = YES /* Carriage control used */
CCTYPE = A /* Carriage control type */
CHARS = GT15
CPGID = 500 /* Code page identifier */

/* FIELD AND INDEX DEFINITION */
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 */
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 */

/* FILE INFORMATION */
INDEXDD = INDEX /* Index file ddname */
INPUTDD = INPUT /* Input file ddname */
OUTPUTDD = OUTPUT /* Output file ddname */
RESOBJDD = RESOBJ /* Resource file ddname */

/* RESOURCE INFORMATION */
FORMDEF = F1A10110 /* Formdef name */
PAGEDEF = P1A08682 /* Pagedef name */
RESFILE = SEQ /* Resource file type */
RESTYPE = FDEF,PSEG,OVLY /* Resource type selection */

/* EXIT AND TRIGGER INFORMATION */
TRIGGER1 = *,1,'1' /* 1st TRIGGER */
TRIGGER2 = 13,50,'ACCOUNT NUMBER:' /* 2nd TRIGGER */
/*
/&

```

Figure 19. Example of a VSE Parameter File

Indexing Data in the Input File

The parameter file you create invokes the ACIF program to index the input file. The example in Figure 14 on page 99 uses these data values as the indexing attributes:

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

You must 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 the indexing attributes:

- Specify the **TRIGGER1** parameter because ACIF always scans for the data specified in **TRIGGER1** first. Because the data contains carriage control characters, include a carriage control character of '1' to indicate a new page. ACIF locates 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.

- Define subsequent **TRIGGER_n** parameters 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_n** parameters.

- 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
```

- Define 'Name' as the indexing attribute to create the indexing tag for the Name 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 contains indexing tags for each of the five indexing attributes. Using AFP Workbench Viewer, customer service representatives can locate a single customer bill in the ACIF document using any combination of the indexing attributes.

Identifying the Locations of the Resources

To build the resource file, you must specify the resource libraries in the parameter file so ACIF knows where to find the resources specified in the job. The parameter file examples for the telephone bill (Figure 16 on page 101, Figure 17 on page 102, Figure 18 on page 103, and Figure 19 on page 104) define these resource libraries:

FDEFLIB	Form definition library
FONTLIB	Font library
OBJCONLIB	Object container library
OVLYLIB	Overlay library
PDEFLIB	Page definition library
PSEGLIB	Page segment and BOCA, GOCA, and IOCA object library

Notes:

1. Resource files processed by ACIF must contain a X'5A' carriage control character at the start of each structured field.
2. See “Selecting Resources” on page 19 for a complete list of the search order for AFP resources in AIX or Windows.

Determining the Form Definition and the Page Definition

To format and print the job, specify form definition and page definition resources in the parameter file. The parameter file examples for the telephone bill (Figure 16 on page 101, Figure 17 on page 102, Figure 18 on page 103, and Figure 19 on page 104) define these resources:

FORMDEF

F1A10110, a standard form definition provided with PSF or Infoprint Manager.

PAGEDEF

P1A08682, a standard page definition provided with PSF or Infoprint Manager.

Running the ACIF Job

Run the ACIF job, depending on your operating system:

AIX and Windows

Use one of these methods:

- Use a parameter file that contains the parameters and values needed for the application (see Figure 16 on page 101). To use a parameter file,

enter the **acif** command, the **PARMDD** parameter, and the parameter file name. For example, to use a parameter file named PARMFILE, specify the following on the command line:

```
acif parmd=PARMFILE
```

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

```
acif cc=yes cctype=z chars=42B2 cpgid=850...
```

(continue entering all of the remaining parameters and values).

See “Examples of Using ACIF Processing Parameters” on page 91 for examples of running ACIF from the command line. For information about creating and running shell scripts, refer to *IBM Infoprint Manager: Reference*.

z/OS and VSE

Run a batch job using the JCL you created for the parameter file (see Figure 17 on page 102 and Figure 19 on page 104).

- VM** Run the CMS command for the parameter file you created (see Figure 18 on page 103).

ACIF processes the parameters that you have specified in the parameter file, on the command line, or in the shell script and creates output files. Table 8 shows the output files that ACIF creates for AIX, Windows, z/OS, VM, and VSE. The Windows operating system is referred to as “WIN” in the table.

Table 8. Output Files ACIF Creates

Type of File	AIX and WIN	z/OS	VM	VSE
Document file, including indexing structured fields	OUTDOC	APKACIF.OUTPUT	APKACIF OUTPUT	APKACIF.OUTPUT
Index object file	INDXOBJ	APKACIF.INDEX	APKACIF INDEX	APKACIF.INDEX
Resource file	RESDATA	APKACIF.RESLIB	APKACIF RESLIB	APKACIF.RESLIB
Message file listing: • ACIF parameters used • Resources used • Return code	acif.msg	APKACIF.SYSPRINT	APKACIF SYSPRINT	APKACIF.SYSPRINT

Concatenating ACIF Output Files

To view the document file on a workstation using AFP Workbench Viewer, you must first concatenate the index object file, the resource file, and the document file. The order of the files in the concatenated file must be the index object file first, the resource file next, and the document file last.

AIX Files

Use one of the methods described in these shell command examples to perform the concatenation of the AIX output files:

- `cat INDXOBJ RESDATA OUTDOC > NEWFILE`

The index object file, the resource file, and the document file are combined to create a new file that contains all three files.

- `cat RESDATA OUTDOC >> INDXOBJ`

The resource file and the document file are added on to the end of the existing index object file.

Windows Files

Use one of the methods described in these shell command examples to perform the concatenation of the Windows output files:

- `copy /b INDXOBJ + RESDATA + OUTDOC NEWFILE`

The index object file, the resource file, and the document file are combined to create a new file that contains all three files.

- `copy /b INDXOBJ + RESDATA + OUTDOC`

The resource file and the document file are added on to the end of the existing index object file.

z/OS Files

The following is an example of z/OS 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 Files

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
```

VSE Files

VSE does not supply a utility for concatenating ACIF output files; you must write your own program.

Accessing the Document File for Viewing

To view the concatenated document file with AFP Workbench Viewer, you must access the file from a workstation running Microsoft Windows. You can use one of these methods to access the file:

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

Notes:

1. You must have TCP/IP installed on both the AIX, Windows, z/OS, VM, or VSE system and the workstation system where AFP Workbench Viewer is installed.
2. To *mount* your AIX or Windows directory on the workstation where AFP Workbench Viewer is installed, you must have TCP/IP with Network File System (NFS) installed on both the AIX or Windows system and on the workstation system where AFP Workbench Viewer is installed.

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

Transferring the Document File to the Workstation

You can use the File Transfer Protocol (FTP) program to *transfer* the concatenated document file to the workstation where Microsoft Windows and AFP Workbench Viewer are installed:

1. 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, Windows, z/OS, VM, or VSE system:
`ftp systemname`
2. Enter your system user name.
3. Enter the password for your system user name.
4. To access the directory where the concatenated document file currently resides, enter:
`cd directoryname`
5. To transfer the file in binary format, enter:
`bin`
6. To transfer a concatenated document file named NEWFILE, enter:
`get NEWFILE`

The file is copied to the workstation, where you can open it for viewing with AFP Workbench Viewer.

7. To exit the FTP program, enter:
`ftp bye`

Mounting the AIX or Windows Directory on the Workstation

You can *mount* your AIX or Windows directory on the workstation where Microsoft Windows and AFP Workbench Viewer are installed by using the NFS **mount** command and the procedures documented in the NFS manuals or your own installation file mounting procedures.

Example of Using Enhanced Indexing with ACIF

You can use enhanced indexing in ACIF to view and index a report, such as a telephone bill. The tasks you do are the same as those described in “Example of Using ACIF to View and Index Documents” on page 95. This section shows a telephone bill and the ACIF enhanced indexing parameter file needed to view and index the bill.

Enhanced Indexing Telephone Bill

Figure 20 on page 110 shows an example of a customer’s telephone bill.

1DETAIL REPORT

TELECOMMUNICATIONS STATEMENT FOR PERIOD OF JUNE 2005
FOR: APPLE, ANNIE

PAGE 0001

SUBGROUP SEQ 4020
ACCOUNTDIVISION SALES
GROUP SEQ 006
DEPARTMENT 3517

APPLE, ANNIE

CALLS VIA PHONE NETWORK

DATE	TIME	ORIGINATION	DESTINATION	DAC	NUMBER	MIN	CHARGES
06-02	08:03	DXSD PLT5	WILLOW GRV PA		215 555-4100	1	.28
06-02	08:24	DXSD PLT5	MEMPHIS TN		901 555-3293	2	.55
06-02	08:46	DXSD PLT5	NASHVILLE TN		615 555-2000	3	.82
06-02	09:23	DXSD PLT5	PHILA PA		215 555-3573	12	3.22
06-02	09:35	DXSD PLT5	COLUMBIA SC		803 555-6781	4	1.08
06-02	09:46	DXSD PLT5	COLUMBIA SC		803 555-6781	2	.55
06-02	09:48	DXSD PLT5	BOUND BROOK NJ		201 555-2939	13	3.49
06-02	10:01	DXSD PLT5	NEW YORK NY		212 555-2470	6	1.62
06-02	10:06	DXSD PLT5	NEW YORK NY		212 555-2470	5	1.35
06-02	10:12	DXSD PLT5	INDIANAPLS IN		317 555-2533	5	1.35
06-02	10:40	DXSD PLT5	CHARLESTON WV		204 555-7000	8	2.15
06-02	10:47	DXSD PLT5	CLEVELAND OH		216 555-1744	4	1.08
06-02	11:01	DXSD PLT5	CLEVELAND OH		216 555-1744	2	.55
06-02	13:09	DXSD PLT5	DETROIT MI		313 555-8889	5	1.35
06-02	13:28	DXSD PLT5	COLUMBIA SC		803 555-3580	5	1.35
06-02	14:51	DXSD PLT5	COLUMBIA SC		803 555-3743	2	.55
06-02	14:57	DXSD PLT5	COLUMBIA SC		803 555-3580	17	4.56
06-02	18:47	DXSD PLT5	BIRMINGHAM MI		313 555-9143	1	.21
06-03	10:00	DXSD PLT5	COLUMBIA SC		803 555-3580	5	1.35
06-03	10:10	DXSD PLT5	COLUMBIA SC		803 555-3580	5	1.35
06-03	10:33	DXSD PLT5	PICORIVERA CA		213 555-7426	1	.29
06-03	14:48	DXSD PLT5	PICORIVERA CA		213 555-7426	6	1.43
06-03	15:22	DXSD PLT5	PITTSBURGH PA		412 555-2536	2	.55
06-03	16:10	DXSD PLT5	IRVINE CA		714 555-4933	1	.27
06-03	16:15	DXSD PLT5	HNTGNTNBCH CA		714 555-6488	14	3.05
06-03	16:29	DXSD PLT5	IRVINE CA		714 555-4933	4	.92
06-03	16:34	DXSD PLT5	KALAMAZOO MI		616 555-0478	3	.82
06-03	16:36	DXSD PLT5	IRVINE CA		714 555-4933	3	.70
06-03	16:39	DXSD PLT5	PICORIVERA CA		213 555-7426	3	.75
06-03	16:41	DXSD PLT5	WLOSANGELS CA		213 555-5000	7	1.76
06-03	16:47	DXSD PLT5	IRVINE CA		714 555-0170	3	.70
06-04	08:03	DXSD PLT5	WILLOW GRV PA		215 555-4133	1	.28
06-04	08:24	DXSD PLT5	MEMPHIS TN		901 555-3293	2	.55
06-04	08:46	DXSD PLT5	NASHVILLE TN		615 555-2000	3	.82
06-04	09:23	DXSD PLT5	PHILA PA		215 555-3573	12	3.22
06-04	09:35	DXSD PLT5	COLUMBIA SC		803 555-6781	4	1.08
06-04	09:46	DXSD PLT5	COLUMBIA SC		803 555-6781	2	.55
06-04	09:48	DXSD PLT5	BOUND BROOK NJ		201 555-2939	13	3.49
06-04	10:01	DXSD PLT5	NEW YORK NY		212 555-2470	6	1.62
06-04	10:06	DXSD PLT5	NEW YORK NY		212 555-2470	5	1.35
06-04	10:12	DXSD PLT5	INDIANAPLS IN		317 555-2533	5	1.35

1DETAIL REPORT

TELECOMMUNICATIONS STATEMENT FOR PERIOD OF JUNE 2005
FOR: APPLE, ANNIE

PAGE 0002

SUBGROUP SEQ 4020
ACCOUNTDIVISION SALES
GROUP SEQ 006
DEPARTMENT 3517

APPLE, ANNIE

(CONTINUED)
CALLS VIA PHONE NETWORK

DATE	TIME	ORIGINATION	DESTINATION	DAC	NUMBER	MIN	CHARGES
06-04	10:22	DXSD PLT5	LOMBARD IL		312 555-1717	9	2.42
06-04	11:08	DXSD PLT5	LOSANGELES CA		213 555-4732	21	5.12
06-04	11:53	DXSD PLT5	KANSASCITY KS		913 555-1400	4	1.08
06-04	12:18	DXSD PLT5	PICORIVERA CA		213 555-7426	15	4.09
06-04	14:38	DXSD PLT5	INDIANAPLS IN		317 555-2533	15	4.15
06-04	15:12	DXSD PLT5	SAN DIEGO CA		619 555-3133	5	.17
06-04	15:35	DXSD PLT5	BOULDER CO		303 555-1442	8	2.15
06-05	10:42	DXSD PLT5	PITTSBURGH PA		412 555-2536	11	2.95
06-05	12:04	DXSD PLT5	PONTIAC MI		313 555-5184	17	4.56
06-05	13:06	DXSD PLT5	BIRMINGHAM MI		313 555-9143	13	3.49
06-06	08:46	DXSD PLT5	MINNEAPOLS MN		612 555-2984	4	1.08
06-06	12:51	DXSD PLT5	PITTSBURGH PA		412 555-3090	3	.82
06-06	12:57	DXSD PLT5	MINNEAPOLS MN		612 555-2984	1	.28
06-06	13:13	DXSD PLT5	MINNEAPOLS MN		612 555-2984	5	1.35
06-06	13:19	DXSD PLT5	PITTSBURGH PA		412 555-3090	10	2.69
06-06	13:28	DXSD PLT5	PITTSBURGH PA		412 555-3090	7	1.88
06-06	14:28	DXSD PLT5	PITTSBURGH PA		412 555-2536	8	2.15
06-06	10:12	DXSD PLT5	INDIANAPLS IN		317 555-2533	5	1.35
06-09	08:03	DXSD PLT5	WILLOW GRV PA		215 555-4133	1	.28
06-09	08:24	DXSD PLT5	MEMPHIS TN		901 555-3293	2	.55
06-09	08:46	DXSD PLT5	NASHVILLE TN		615 555-2000	3	.82
06-09	09:23	DXSD PLT5	PHILA PA		215 555-3573	12	3.22
06-09	09:35	DXSD PLT5	COLUMBIA SC		803 555-6781	4	1.08
06-09	09:46	DXSD PLT5	COLUMBIA SC		803 555-6781	2	.55
06-09	09:48	DXSD PLT5	BOUND BROOK NJ		201 555-2939	13	3.49
06-09	10:01	DXSD PLT5	NEW YORK NY		212 555-2470	6	1.62
06-09	10:06	DXSD PLT5	NEW YORK NY		212 555-2470	5	1.35
06-09	10:12	DXSD PLT5	INDIANAPLS IN		317 555-2533	5	1.35
06-09	10:40	DXSD PLT5	CHARLESTON WV		204 555-7000	8	2.15
06-09	10:47	DXSD PLT5	CLEVELAND OH		216 555-1744	4	1.08
06-09	11:01	DXSD PLT5	CLEVELAND OH		216 555-1744	2	.55
06-09	13:09	DXSD PLT5	DETROIT MI		313 555-8889	5	1.35
06-09	13:28	DXSD PLT5	COLUMBIA SC		803 555-3580	5	1.35
06-09	14:51	DXSD PLT5	COLUMBIA SC		803 555-3743	2	.55
06-09	14:57	DXSD PLT5	COLUMBIA SC		803 555-3580	17	4.56
06-09	18:47	DXSD PLT5	BIRMINGHAM MI		313 555-9143	1	.21

Figure 20. Example of the Enhanced Indexing Telephone Bill

Enhanced Indexing Parameter File

The JCL for a z/OS enhanced indexing parameter file is shown in Figure 21. This example creates a sequential data set; however, if you need a partitioned data set, set these parameters:

- **RESFILE=PDS**
- **SPACE** and **DSORG** parameters in the DD statement of the data set named by the **RESOBJDD** parameter to **SPACE=(12288,(150,15,15)),DSORG=PO**.

Failure to set these parameters as described might produce a **RESOBJDD** data set that is unusable.

```
CC=YES
CCTYPE=A
CPGID=500
INPUTDD = INPUT
INDEXDD = INDEX
MSGDD = SYSPRINT
OUTPUTDD = OUTPUT
/*TRACE = YES
RESOBJDD = RESOURCE
INDEXSTARTBY=7
INDEXOBJ=ALL
FORMDEF = F1IBM
TRIGGER1=*,1,X'F1'
TRIGGER2=0,125,X'F0F0F0F1' /* Anchor point top-of-page
TRIGGER3=0,2,X'C4C5E3' /* Anchor point for DET
TRIGGER4=*,60,X'C6D6C9', (TYPE=FLOAT)
TRIGGER5=*,60,X'C6D6D9', (TYPE=FLOAT)
FIELD1=6,2,20 /* Pick up name
FIELD2=5,116,4 /* Pick up department number
FIELD3=4,116,3 /* Pick up group name
FIELD4=3,19,4 /* Pick subgroup number
FIELD5=0,92,10, (TRIGGER=4) /* Pick up total using trigger2
FIELD6=0,92,10, (TRIGGER=5) /* Pick up total using trigger2
/* INDEX1='NAME', FIELD1 /* Put literal in index
INDEX1=X'D5C1D4C5', FIELD1 /* Put literal in index
/* INDEX2='Dept', FIELD2 /* Put literal in index
INDEX2=X'C4C5D7E3', FIELD2 /* Put literal in index
/*INDEX3='SubGroup',FIELD3 /* Put literal in index
INDEX3=X'E2E4C2C7D9D6E4D7',FIELD3 /* Put literal in index
/* INDEX4='GROUP',FIELD4 /* Put literal in index
INDEX4=X'C7D9D6E4D7',FIELD4 /* Put literal in index
/* INDEX5='TOTAL',FIELD5 /* Put literal in index
INDEX5=X'E3D6E3C1D3',FIELD5 /* Put literal in index
INDEX6=X'E3D6E3C1D3F6',FIELD6, (TYPE=PAGE) /* Put literal in index
PAGEDEF = P1C09182
TRC=NO
RESFILE=SEQ
RESTYPE=ALL
/* PAGEDEF = P1STD1
FDEFLIB = SYS1.FDEFLIB,ACIF.REGRESS.RESLIB
FONTLIB = FONTS.FONTLIB,FONTS.FONTLIBB,FONTS.FONTLIBB.EXTRA,
ACIF.REGRESS.RESLIB
OVLYLIB = SYS1.OVRLIB
PDEFLIB = SYS1.PDEFLIB,ACIF.REGRESS.RESLIB
PSEGLIB = SYS1.PSEGLIB
```

Figure 21. Example of an Enhanced Indexing Parameter File

Chapter 6. User Exits and Input Print File Attributes

A user exit is a point during ACIF processing that lets you 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 chapter contains programming interface information and describes the four user programming exits provided with ACIF. It also describes the information ACIF provides to the exits about the input print file attributes.

User Programming Exits

ACIF provides these sample programming exits so you can customize the program:

- Input record
- Index record
- Output record
- Resource

The exits are described in the following sections. Sample AIX or Windows C language headers and z/OS, VM, or VSE DSECTs are shown for each programming exit. The C exits are intended for use on AIX and Windows only.

Using the programming exits is optional. You specify the names of the exit programs with the **INPEXIT**, **INDEXEXIT**, **OUTEXIT**, and **RESEXIT** parameters. (These parameters are lowercase in AIX and Windows.) Each of these parameters is described in Chapter 3, “ACIF Parameters.”

Note: If ACIF receives a non-zero return code from any exit program, ACIF issues message APK412I and stops processing.

ACIF sample code for z/OS, VM, and VSE exits is available from the software downloads section of the IBM Printing Systems Division Web page, <http://www.ibm.com/printers>.

ACIF provides the sample code for AIX or Windows exits in the AIX directory `/usr/lpp/psf/acif/` or the Windows directory `install_directory\exits\acif\`. Microsoft Visual C++ project (*.dsp) and workspace (*.dsw) files are also provided in the Windows directory. The sample exits are:

apkinp.c
Input record exit

apkind.c
Index record exit

apkout.c
Output record exit

apkres.c
Resource exit

In addition, ACIF provides these AIX or Windows user input record exits to translate input data streams:

apka2e.c

Converts ASCII stream data to EBCDIC stream data. You can also convert encoded data to another coded character set identifier (CCSID) if you specify the **INPCCSID** and **OUTCCSID** parameters.

asciinp.c

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

asciipe.c

Converts unformatted ASCII data into a record format as does **asciinp.c**; then, converts the ASCII stream data to EBCDIC stream data. You can also convert encoded data to another coded character set identifier (CCSID) if you specify the **INPCCSID** and **OUTCCSID** parameters.

The C language header file for all AIX or Windows exit programs, **apkexits.h**, is also provided along with the build rules for the AIX user exits, **Makefile**.

For more information about compiling AIX or Windows user exit programs, refer to *IBM Infoprint Manager for AIX: Procedures* or *IBM Infoprint Manager for Windows: Procedures*.

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 **INPEXIT** parameter.

This exit is called after each record is read from the input file and before any further processing is performed on the input record. 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 libraries.

Note: ACIF issues message APK419S with Return Code 999 (RC=999) and stops processing if the input exit returns a zero length record.

In a MO:DCA-P document, indexing information can be passed in the form of a Tag Logical Element (TLE) structured field (see “Tag Logical Element (TLE) Structured Field” on page 201 for more information). The exit program can create these structured fields while ACIF is processing the print file. You can insert No Operation (NOP) structured fields into the input file in place of TLEs and use ACIF’s indexing parameters (**FIELD_n**, **INDEX_n**, and **TRIGGER_n**) to index the NOPs. 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 data, XML data, or mixed-mode data.

Figure 22 on page 115 contains a sample C language header that describes the control block that is passed to the AIX or Windows 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 */
    unsigned short in_CCSID;      /* Input CCSID for translating */
    unsigned short out_CCSID;     /* Output CCSID for translating */
    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 22. AIX or Windows Sample Input Record Exit C Language Header

Figure 23 contains a sample DSECT that describes the control block for z/OS, VM, or VSE exit programs.

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 23. z/OS, VM, or VSE Sample Input Record Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, VM, and VSE, the address is passed in a standard parameter list pointed to by register 1; for AIX and Windows, the address is passed by the first parameter.

Note: AIX and Windows parameters are specified in lowercase if they differ from the uppercase parameters listed; for example, **WORK@** | **work** means that the parameter for z/OS, VM, or VSE is **WORK@** while the parameter for AIX and Windows is **work**. Otherwise, unless specified, all uppercase parameters are assumed to be lowercase for AIX and Windows.

WORK@ | 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 before the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ | pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 125 for more information about the format of this data structure and the information it contains.

RECORD@ | 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.

in_CCSID (Bytes 13-14)

The value from the **INPCCSID** parameter in AIX or Windows.

| **out_CCSID (Bytes 15-16)**

| The value from the **OUTCCSID** parameter in AIX or Windows.

| **RESERVED1 (Bytes 13-16)**

| These bytes are reserved for future use in z/OS, VM, or VSE.

| **RECORDLN (Bytes 17–18)**

| 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.

| When using the **INPCCSID** and **OUTCCSID** parameters, the actual length of the field might differ from the input record length if you are converting to or from UTF-16 data.

| **Note:** ACIF issues message APK419S with Return Code 999 (RC=999) and stops processing if the input exit returns a zero length record.

| **RESERVED2 (Bytes 19–20)**

| These bytes are reserved for future use.

| **REQUEST (Byte 21)**

| An indication of 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 is to be processed by ACIF.

| **X'01'** Specifies that the record is not to be processed by ACIF.

| **X'02'** Specifies that the record is to be processed by ACIF and control returned to the exit program to let it 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=Y**), the last record has already been presented to the input exit, and the input file has been closed. The pointer **RECORD@ | record** is no longer valid. Records cannot 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 lets the exit program perform some additional processing at the end of the print file. The exit program cannot change this parameter.

| **Using ACIF User Input Record Exits in AIX and Windows**

| ACIF provides these AIX or Windows user input record exits to translate input data streams:

apka2e

The **apka2e** input record exit program uses the **uconv** command to convert ASCII stream data to EBCDIC stream data. You can also convert encoded data to another coded character set identifier (CCSID) if you specify the **INPCCSID** and **OUTCCSID** parameters. You use this exit when your print job requires fonts with code points that are different from your data file. For example, GT12 has only EBCDIC code points defined. If the **INPCCSID** and **OUTCCSID** parameters are not specified, the default translation is from ASCII (code set IBM-850) to EBCDIC (code set IBM-037). Otherwise, **INPCCSID** specifies the input code page and **OUTCCSID** specifies the output code page.

asciinp

You can use **asciinp** if your unformatted ASCII file contains carriage returns and form feeds. The **asciinp** input record exit program translates an unformatted ASCII data stream into a record format that contains an ANSI 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.

asciinpe

You use **asciinpe** if your unformatted ASCII file contains carriage returns and form feeds, and your print job requires fonts with code points that are different from your data file. The **asciinpe** input record exit program translates an unformatted ASCII data stream into a record format just as **asciinp** does. Then, **asciinpe** uses the **uconv** command to convert the ASCII stream data to EBCDIC stream data. You can also convert encoded data to another coded character set identifier (CCSID) just as **apka2e** does if you specify the **INPCCSID** and **OUTCCSID** parameters.

To use an input record exit program to translate input data streams:

1. Specify `inpexit=exitprogram` in ACIF, where *exitprogram* is **apka2e**, **asciinp**, or **asciinpe**. Ensure that the directory where the input record exit program resides is included in the **PATH** environment variable; otherwise, you must specify the *exitprogram* with the full path name, such as: `inpexit=/usr/lpp/psf/bin/apka2e` for AIX or `inpexit=c:\exits\acif\apka2e.dll` for Windows. See “INPEXIT” on page 53 for more information about this parameter.
2. Optionally, if you want to convert encoded data other than ASCII to EBCDIC, specify these parameters:
 - `inpccsid=ccsid`, where *ccsid* is a valid CCSID for the input code page
 - `outccsid=ccsid`, where *ccsid* is a valid CCSID for the output code page

See “INPCCSID” on page 53 and “OUTCCSID” on page 58 for more information about these parameters.

Notes:

1. If **asciinp** is to be used with ACIF to produce an index file, consideration must be made for the carriage control character inserted by **asciinp** into byte 0 when determining offsets for indexing parameters.
2. 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 about using and modifying these programs, refer to the prolog of the **asciinp.c** source file that is provided with Infoprint Manager for AIX in the **/usr/lpp/psf/acif** directory or with Infoprint Manager for Windows in the **\install_directory\exits\acif** directory.

Index Record Exit

ACIF provides an exit that lets you modify or ignore the records that ACIF writes in the index object file. The program invoked at this exit is defined by the **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 24 contains a sample C language header that describes the control block that is passed to the AIX or Windows exit program.

```
typedef struct _INDEXEXIT_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 */
} INDEXEXIT_PARMS;
```

Figure 24. AIX or Windows Sample Index Record Exit C Language Header

Figure 25 contains a sample DSECT that describes the control block that is passed to the z/OS, VM, or VSE 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 25. z/OS, VM, or VSE Sample Index Record Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, VM, and VSE, the address is passed in a standard parameter list pointed to by register 1; for AIX and Windows, the address is passed by the first parameter.

Note: AIX and Windows parameters are specified in lowercase if they differ from the uppercase parameters listed; for example, **WORK@** | **work** means that the parameter for z/OS, VM, or VSE is **WORK@** while the parameter for AIX or Windows is **work**. Otherwise, all uppercase parameters are assumed to be lowercase for AIX or Windows.

WORK@ | **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 before to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ | pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 125 for more information about the format of this data structure and the information it contains.

RECORD@ | record (Bytes 9–12)

A pointer to the first byte of the index record including the carriage control character. The record resides in a 32 KB 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)

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)

An indication of 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 is to be processed by ACIF.

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

A value of X'00' on entry to the exit program specifies that the record is to 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 processing the index object file.

When **EOF** is signaled (**EOF=Y**), the last record has already been presented to the index exit. The pointer **RECORD@ | record** is no longer valid. Records cannot 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, lets the exit program 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 **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 26 contains a sample C language header that describes the control block passed to the AIX or Windows 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 26. AIX or Windows Sample Output Record Exit C Language Header

Figure 27 contains a sample DSECT that describes the control block passed to the z/OS, VM, or VSE 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 27. z/OS, VM, or VSE Sample Output Record Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, VM, and VSE, the address is passed in a standard parameter list pointed to by register 1; for AIX and Windows, the address is passed by the first parameter.

Note: AIX and Windows parameters are specified in lowercase if they differ from the uppercase parameters listed; for example, **WORK@** | **work** means that the parameter for z/OS, VM, or VSE is **WORK@** while the parameter for AIX or Windows is **work**. Otherwise, all uppercase parameters are assumed to be lowercase for AIX or Windows.

WORK@ | 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 before to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ | pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 125 for more information about the format of this data structure and the information contained in it.

RECORD@ | record (Bytes 9–12)

A pointer to the first byte of the output record. The record resides in a 32 KB 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)

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)

An indication of 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 is to be processed by ACIF.

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

A value of X'00' on entry to the exit program specifies that the record is to 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 signaled (**EOF=Y**), the last record has already been presented to the output exit. The pointer **RECORD@** | **record** is no longer valid. Records cannot 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, lets the exit program return to ACIF. The exit program cannot change this parameter.

Resource Exit

ACIF provides an exit that lets you “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 are going to send ACIF output to PSF and you only wanted to send those fonts that were not included with the PSF product. You can code this exit program to contain a table of all fonts included with PSF and filter those 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 **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 to be ignored, ACIF automatically ignores any resources the overlay references (that is, fonts and page segments).

Figure 28 contains a sample C language header that describes the control block that is passed to the AIX or Windows 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 (8 byte) */
    char          restype;       /* Type of resource */
    char          request;       /* Ignore or process the resource */
    char          eof;          /* Last call indicator */
    unsigned short resnamel;     /* Length of resource name */
    char          pad1[3];       /* Padding byte */
    char          resnamf[250];  /* Resource name if more than 8 bytes */
} RESEXIT_PARMS;

```

Figure 28. AIX or Windows Sample Resource Exit C Language Header

Figure 29 contains a sample DSECT that describes the control block that is passed to the z/OS, VM, or VSE exit program.

PARMLIST	DSECT		Parameters for the output record exit
WORK@	DS	A	Address of 16-byte work area
PFATTR@	DS	A	Address of print file attributes
RESNAME	DS	CL8	Resource object name (long)
RESTYPE	DS	X	Resource type indicator
REQUEST	DS	X	Use or ignore action code
EOF	DS	X	Last call flag
RESNAMEL	DS	H	Length of resource name
RESERVED	DS	CL3	Reserved
RESNAMF	DS	CL250	Resource name if more than 8 bytes

Figure 29. z/OS, VM, or VSE Sample Resource Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, VM, and VSE, the address is passed in a standard parameter list pointed to by register 1; for AIX and Windows, the address is passed by the first parameter.

Note: AIX and Windows parameters are specified in lowercase if they differ from the uppercase parameters listed; for example, **WORK@** | **work** means that the parameter for z/OS, VM, or VSE is **WORK@** while the parameter for AIX or Windows is **work**. Otherwise, unless specified, all uppercase parameters are assumed to be lowercase for AIX or Windows.

WORK@ | 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 before to the first call. The user-written exit program must provide the code required to manage this work area.

PFATTR@ | pfattr (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 125 for more information about the format of this data structure and the information presented.

RESNAME (Bytes 9–16)

The name of the resource to be included. This can be a file or member name for AFP resources up to eight characters. For resources names of more than eight characters, use **RESNAMF**. The resource type field defines how the **RESNAME** should be interpreted.

RESTYPE (Byte 17)

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

- X'03'** Specifies a GOCA (graphics) object
- X'05'** Specifies a BCOCA (bar code) object
- X'06'** Specifies an IOCA (IO image) object
- X'40'** Specifies a font character set
- X'41'** Specifies a code page
- X'42'** Specifies a coded font
- X'92'** Specifies an object container (also applies to TrueType and OpenType font objects)
- 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 data, XML data, mixed-mode data, and unformatted ASCII data. 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

If **MCF2REF=CF** is specified, coded fonts are included in the resource file. Otherwise, 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.

COM setup files

A COM setup file is a required resource for processing microfilm files (*microfilm* can mean either microfiche or 16 mm film). 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)

An indication of 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 is to be processed by ACIF.
- X'01'** Specifies that the resource is not to be processed by ACIF.

A value of X'00' on entry to the exit program specifies that ACIF should process the resource. 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 signaled (**EOF=Y**), the last record has already been presented to the resource exit. The pointer **RECORD@** | **record** is no longer valid. Records cannot 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.

RESNAMEL (Bytes 20–21)

The actual length of the meaningful characters in **RESNAME** and **RESNAMF**.

RESERVED | pad1 (Bytes 22-24)

Reserved bytes used for padding or future use.

RESNAMF (Bytes 25–274)

The name of the resource to be included if more than eight characters. This can be a full font name for TrueType or OpenType fonts (up to 250 Unicode characters). The resource type field defines how the **RESNAMF** should be interpreted.

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.

AIX and Windows

ACIF searches for the exit program in the paths specified by the **PATH** environment variable or in the paths you specify with the exit program parameters. See “INDEXIT” on page 53, “INPEXIT” on page 53, “OUTEXIT” on page 58, and “RESEXIT” on page 66 for more information.

If the **INPCCSID** and **OUTCCSID** parameters are used with an exit, the **PATH** environment variable must include the directory that contains the ICU libraries shipped with Infoprint Manager for AIX or Infoprint Manager for Windows.

z/OS

Exit load modules can reside in a load library used as STEPLIB, JOBLIB, or in a system library. ACIF uses the standard z/OS 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.TEXT* or *name.TEXTLIB*.

VSE

Exit load modules are located in the library defined by the // LIBDEF PHASE,SEARCH=(...) JCL statement.

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 30 shows a sample C language header that describes the format of the AIX or Windows 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 30. AIX or Windows Sample Print File Attributes C Language Header

Figure 31 shows a sample DSECT that describes the format of the z/OS, VM, or VSE 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 31. z/OS, VM, or VSE Sample Print File Attributes DSECT

The address of the control block containing the following parameters is passed to the input record exit. For z/OS, VM, and VSE, the address is passed in a standard parameter list pointed to by register 1; for AIX and Windows, the address is passed by the first parameter.

Note: All uppercase parameters are assumed to be lowercase for AIX and Windows.

CC (Bytes 1–3)

The value of the **CC** parameter as specified on the **acif** command or 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 on the **acif** command or in the ACIF processing parameter file. 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 or in the ACIF processing parameter file, 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 or 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 on the **acif** command or 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 on the **acif** command or 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 on the **acif** command or in the ACIF processing parameter file. ACIF uses the default value if this parameter is not explicitly specified.

Notes:

1. Each of the previous character values is left-justified, with padding blanks added to the right-end of the string. For example, if **PAGEDEF=P1TEST** is specified on the **acif** command or 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 '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 7. ACIF Messages

ACIF prints a message list at the end of each compilation. A return code of 0 means that ACIF completed processing without any errors.

Notes:

1. ACIF messages contain instructions for the PSF or Infoprint Manager system programmer. Please show your system programmer these messages, because they might not be contained in the PSF or Infoprint Manager messages publications.
2. AIX and Windows users can invoke the PSF MSG command to view or print messages online.
3. z/OS users can use the LookAt online facility to view explanations for ACIF messages. See “Using LookAt to View Message Explanations” on page xvi for more information.

Message Identifiers

ACIF issues the same messages for AIX, Windows, z/OS, VM, and VSE users. However, the message identifiers in AIX and Windows differ from those in z/OS, VM, and VSE:

- In AIX and Windows, the format of the message identifier is **0425-*nnnn***, where:

0425- Identifies a message in AIX or Windows.

nnnn Specifies the three-digit or four-digit message number.

- In z/OS, VM, and VSE, the format of the message identifier is **APK*nnnnt***, where:

APK Identifies a message in z/OS, VM, or VSE.

nnnn Specifies the three-digit or four-digit message number.

t Specifies an error condition:

Error Type	Description
------------	-------------

S

Severe error that causes ACIF to stop processing the current print file. The exact method of stopping can 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 ends processing with the appropriate error messages written to the message file specified when you invoked ACIF. Most error conditions detected by ACIF fall into the severe category.

W

Warning error that ACIF issues when the fidelity of the document (assuming it is reprinted) might be in question.

I

Informational error that ACIF issues when it processes a print file to let the operator or application programmer determine if the correct processing parameters have been specified. These messages can assist in providing an audit trail.

The message number, *nnnn*, is the same in all environments (for example, 0425-**031** in AIX or Windows or **APK031I** in z/OS, VM, or VSE). In this publication, the ACIF messages and explanations are listed according to the z/OS, VM, and

VSE message identifiers (for example, APK031I) because AIX and Windows users are more likely to use the PSF MSG command to view the messages online.

The terms used in the messages and explanations are those used for z/OS, VM, and VSE, even though the messages and explanations also apply to AIX and Windows. The following list shows some of the terms used in the messages in this publication and what those terms refer to in AIX and Windows:

Term	AIX, Windows
print data set	input file
data set	input file
data stream	file
PSF	PSF or Infoprint Manager
record	data set

Multiple Message Scenarios

ACIF can 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 these message numbers:

- 105, 108, 109, 103
- 105, 108, 110, 103
- 106, 108, 109, 103
- 106, 108, 110, 103

Any subset of the listed message numbers 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 might not always be accurate.

Message number 101 can 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 can occur in any resource.

APK104S **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID:** *structuredfield*
STRUCTURED FIELD IS NOT ALLOWED OR FORMS AN INVALID SEQUENCE.

Explanation: The structured field identified in this message is either out of sequence or not valid in an object. The record might be line data. If inline resources are used with data-set header pages, multiple resource groups might be present.

| This message is issued if your AFP input file contains
| IM1 image with no Image Raster Data (IRD) structured
| field. This type of AFP file might be built by DCF when
| creating shaded text with a shading percentage of zero.

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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK105I THE ERROR REPORTED ABOVE OCCURRED IN LOGICAL RECORD NUMBER *recordnumber*, WHOSE SEQUENCE NUMBER IS *sequencenumber*, AND RESOURCE NAME IS *resourcename*.

Explanation: This message is given in addition to the message that describes the error. It identifies the specific input record that is not valid. The object (if any) that contains the not valid record is identified in either message APK108I or message APK109I.

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 might be inaccurate if the data set is using a page definition that performs conditional processing.

The sequence number might 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 "*tokenname*" IN *begintypestructuredfield* DOES NOT MATCH NAME "*tokenname*" IN *endtypestructuredfield*.

Explanation: The TOKEN NAME parameters in the Begin-type and End-type structured fields identified in this message do not match. Structured fields might 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 match.

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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK108I THE ERROR REPORTED ABOVE WAS DETECTED WITHIN OBJECT TYPE *objecttype* WITH TOKEN NAME *tokenname*.

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 *resourcename* 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. Error message APK108I identifies 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 OVLY 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 OR TRIPLET *identifier* OF THE *structuredfield* STRUCTURED FIELD IS INCORRECT.

Explanation: Insufficient data was present in the structured field for the length given in the self-defining parameter or triplet. If the self-defining parameter or triplet ID is 0, the length of the self-defining parameter or triplet might have been 0 or 1, which means that no ID was available for use in this message.

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. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

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 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 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 might 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 your system's diagnosis reference 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 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 print data set or resource, ensure that the RDW for the structured field that is not valid 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 your system's diagnosis reference 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 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 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 structured field that is not valid. 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 might 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 your system's diagnosis reference 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 *structuredfield* STRUCTURED FIELD.

Explanation: The length indicated by the structured-field introducer specifies an incorrect number of bytes for the structured field identified in this

message. This error is caused by one of these :

- The Extension or Padding Indicator flags in the structured-field introducer are set incorrectly.
- One or more of the parameters in the structured fieldthat is not valid 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 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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK118W UNSUPPORTED 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 incorrect 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 incorrect 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK120S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structuredfield1* STRUCTURED FIELD CONTAINS AN INCORRECT VALUE FOR THE SIZE OF THE *structuredfield2* REPEATING GROUP.

Explanation: *Structuredfield1* specifies the length of each repeating group found in *structuredfield2*. Either the value specified in *structuredfield1* 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 and issues a message identifying the position of the structured field in the data stream or resource.

User response: Use the accompanying messages to determine if the structured field causing the error is in the print data set or in a resource. Correct the process used to create the print data set or resource. If you used an IBM licensed program to create the data stream with the error, use local problem-reporting procedures to report this message.

APK130S DATA IN AN INPUT RECORD IS INVALID: *structuredfield* 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 might

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 your system's diagnosis reference 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 *structuredfield* 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 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 might 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 your system's diagnosis reference 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 *structuredfield* STRUCTURED FIELD.

Explanation: An incorrect 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 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 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 might 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 your system's diagnosis reference 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 not valid. The MSU 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 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 might 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 your system's diagnosis reference 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 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, 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 might 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 your system's diagnosis reference 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 incorrect 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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* 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 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 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 might 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 your system's diagnosis reference 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 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 might 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 your system's diagnosis reference 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 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, 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 your system's diagnosis reference 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 structuredfield 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 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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK157S MISMATCH BETWEEN PRINT DATA SET AND FORMDEF RESOURCE: MEDIUM MAP "mediummap" SPECIFIED IN IMM STRUCTURED FIELD WAS NOT FOUND IN FORMDEF "formdefinition".

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 and 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 embed 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.

APK158I 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 line data, XML data, or structured fields that do not form a MO:DCA-P page. This kind of data set cannot be printed without an active page definition. No PAGEDEF keyword was provided for this job.

This error can also occur if MO:DCA-P data in the print data set contains a record 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. Therefore, ACIF detects an error.

System action: ACIF stops processing the print data set.

User response: If you intended to print line data or XML data, you must specify the PAGEDEF keyword when invoking ACIF.

If you did not intend to print line data or XML data, and

you used a program to create the structured fields for the print data set, ensure that all MO:DCA-P data records begin with the X'5A' control character and then 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK162S MISMATCH BETWEEN PRINT DATA SET AND PAGEDEF RESOURCE: DATA MAP "datamap" SPECIFIED IN IDM STRUCTURED FIELD WAS NOT FOUND IN PAGEDEF "pagedefinition".

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 and 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 embed 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.

APK163I 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 not valid. The image block or image cell might be contained in an overlay, a page segment, or a composed-text print data set. It might also be embedded in a data set containing line data, using a Begin Image (BIM) 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 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 might 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 your system's diagnosis reference 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: A font in the Map Coded Font (MCF) structured field can be identified with a CODED FONT NAME parameter, with a combination of the FONT CHARACTER SET NAME parameter and the CODE PAGE NAME parameter, or with a CODED FONT parameter (also known as a GRID parameter). One of the repeating groups in an MCF structured field specified more than one of these ways to specify a font

or specified a CODED FONT (GRID) and a section number other than 0. The MCF structured field is in the MO:DCA-P data, an overlay, or a 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 print data set or the resource, 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 not valid. 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 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 might 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 your system's diagnosis reference 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 *structuredfield*
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 not valid. 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 not valid. 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 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 might 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 your system's diagnosis reference 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 not valid. 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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK210S DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A REQUIRED TRIPLET OR SELF-DEFINING PARAMETER WITH ID *identifier* WAS MISSING FROM A *structuredfield* STRUCTURED FIELD.

Explanation: The triplet or self-defining parameter specified in the message was not found in the structured field indicated. This is a required triplet or self-defining parameter.

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, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might be an ACIF logic error. If you used a program to create the structured fields, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the structured field with the

error, verify that the input to that program was valid.

APK212S DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE UNIT BASE PARAMETER IN THE *structuredfield* STRUCTURED FIELD IS NOT VALID.

Explanation: An incorrect Unit Base value was encountered in the structured field identified in this message.

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: Correct the process used to create the image object. If you used an IBM licensed program to create the image object with the error, use local problem-reporting procedures to report this message.

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 (BRS or BR) structured field is not valid. The BRS structured field is contained in the print data set.

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 placed the BRS 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 BRS structured field in the print data set, contact your system programmer.

System programmer response: If an IBM licensed program was used to place the BRS structured field in the print data set, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis 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 incorrect orientation value. The MDD 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK244I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE *structuredfield* STRUCTURED FIELD CONTAINS TOO MANY REPEATING GROUPS.

Explanation: The structured field contains more repeating groups than are allowed. The structured field in which the error appears can be in a resource environment group, a composed text page, an overlay, or a 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 print data set or the resource, 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK245I A COMPLEX IM IMAGE OBJECT CONTAINS INVALID OR INCORRECT DATA. THE COMPLEX IM IMAGE OBJECT CANNOT BE CONVERTED TO AN IO IMAGE OBJECT.

Explanation: This message is issued when ACIF converts a complex IM image object to an IO image object and the image size is not large enough to contain the image raster data from the IRD structured fields. This message is issued when the default IMAGEOUT=IOCA parameter is specified. This message is issued if either of these are true:

- The XCSize or YCSize parameter value of the ICP structured field is larger than the calculated image X size or Y size, respectively.
- The XCOset plus XFilSize parameter values or the YCOset plus YFilSize parameter values of the ICP structured field are larger than the calculated image X size or Y size, respectively.

When ACIF converts a complex IM image object to an IO image object, ACIF calculates the image size by subtracting the X and Y image origins from the X and Y page sizes. The X and Y image origins are from the Xoa0set and Yoa0set parameter values of the IOC structured field. The X and Y page sizes are from the XpgSize and YpgSize parameter values of the PGD structured field, if the image object is contained in a MO:DCA-P file or overlay, or is embedded in a file containing line data. For an image object in a page segment, the X and Y page sizes used by ACIF are 2040 and 2640 respectively. The IOC and ICP structured fields are contained in a MO:DCA-P file, overlay, or page segment, or are embedded in a file containing line data. The PGD structured field is contained in a MO:DCA-P file, overlay, or page definition.

System action: ACIF stops.

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* for more information about the structured field. If the structured field has no error, the error might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK250S DATA IN A PAGE OR RESOURCE IS MISSING: THE REQUIRED STRUCTURED FIELD *structuredfield* 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 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 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 might 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 your system's diagnosis reference 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference 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 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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK258I DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: structuredfield 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.

Note: If the structured field is "EOF", ACIF read the entire print data set without finding an expected 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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK259I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE X-DIRECTION AND Y-DIRECTION L-UNITS PER UNIT BASE VALUES SPECIFIED IN THE *structuredfield* 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 and issues a message identifying the position of the structured field in the data stream or resource.

User response: Use the accompanying messages to determine if the structured field causing the error is in the print data set or in a resource. Correct the process used to create the print data set or resource. If you used an IBM licensed program to create the data stream with the error, use local problem-reporting procedures to report this message.

APK261S DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: STRUCTURED FIELD *structuredfield* CONTAINED A 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 font local identifier for different fonts. Fonts can be mapped in a Map Coded Font (MCF) and a Map Data Resource (MDR) structured field. Each font mapped must have a unique font local identifier. The MCF and MDR structured fields can be in the MO:DCA print data set, an overlay, a graphics object, a bar code object, or a page definition.

User response: If you created the structured fields in the object containing the error, check the font local identifiers in the MCF and MDR structured field for duplicates. If the MCF and MDR structured fields have no error, the error might 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.

APK262S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD *structuredfield* CONTAINS AN INVALID ROTATION VALUE.

Explanation: The rotation value specified in the named structured field was not valid.

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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK263S OVERLAY *overlayname* 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 might be contained in the AEG of a composed-text page or a 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 the MPO indicates that this overlay is for annotation only, create another MPO structured field in the AEG that defines the page overlay. If you are using the input data to define the name of your page overlay and your input data is ASCII, this error can occur because the resource name in the MPO is EBCDIC. 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 more information about the structured field. If the structured field has no error, the error might be an ACIF logic error. If you use 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK264S DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A RESOURCE MAPPED BY A *structuredfield* STRUCTURED FIELD IN AN OBJECT ENVIRONMENT GROUP IS NOT NAMED IN THE ACTIVE ENVIRONMENT GROUP OF THE PAGE OR RESOURCE.

Explanation: A structured field in an object environment group names a resource. However, that resource is not defined in the 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 and issues a message identifying the position of the structured field in the data stream or resource.

User response: Make sure that any resources that are mapped in an included object are also mapped in the page environment group. If there is an MCF structured field inside the object, it also needs to be in the page environment group with the identical characteristics it has in the object, such as point size for an outline font. 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 might 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 your system's diagnosis reference 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 might be contained in an overlay, a page definition, or a composed-text print data set.

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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK268S WARNING: 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 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 might 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 your system's diagnosis reference 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 *structuredfield* 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), Image Input Descriptor (IID), Include Object (IOB), and Preprocess Presentation Object (PPO). The ACIF arithmetic equation and a value of zero can result in an abend (divide by zero error). The value of zero is not valid.

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: Use the accompanying messages to determine if the structured field causing the error is in the print data set or in a resource. Correct the process used to create the print data set or resource. If you used an IBM licensed program to create the data stream with the error, use local problem-reporting procedures to report this message.

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 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 might 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 your system's diagnosis reference 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 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 might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference 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 ID *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 might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK278S DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE MAPPING OPTION SPECIFIED IN THE *structuredfield* STRUCTURED FIELD IS INCORRECT OR UNSUPPORTED.

Explanation: The structured field in error contained an incorrect Mapping Option value or the printer does not support the Mapping Option value. The structured field could be contained in a bar code object, graphics object, image object, or object container object. Alternatively, it could be an IOB structured field or a Preprocess Presentation Object (PPO) structured field with an incorrect mapping option triplet.

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: Correct the process used to create the object. If you used an IBM licensed program to create the object with the error, use the local problem-reporting procedures to report this message.

APK299I 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, XFIlSize, or YFIlSize parameter values of the ICP structured field is zero.

System action: ACIF stops processing the print data set.

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 your system's diagnosis reference 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 *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 might 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 your system's diagnosis reference 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 *number* IS *parametervalue*. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF *parametervalue*.

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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK307S DATA IN A PAGEDEF RESOURCE IS INVALID: IN LND STRUCTURED FIELD NUMBER *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 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect 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 might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect 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 might 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 your system's diagnosis reference 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 *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 might 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 your system's diagnosis reference 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 *number* IS *parametervalue*. 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect 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 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK319I DATA IN A PAGEDEF RESOURCE IS NOT VALID: LND, RCD, OR XMD STRUCTURED FIELD NUMBER *number* HAS A NULL VALUE SPECIFIED IN THE SUPPRESSION TOKEN NAME PARAMETER. A NULL VALUE IS NOT VALID.

Explanation: The SUPPRESSION TOKEN NAME parameter in the Line Descriptor (LND), Record Descriptor (RCD), or XML Descriptor (XMD) 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.

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 might 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 your system's diagnosis reference 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 incorrect 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 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect value. The TIMING OF ACTION parameter in one of the repeating groups of the CCP structured field contains an incorrect 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect value. The MEDIUM MAP ACTION parameter in one of the repeating groups of the CCP structured field contains an incorrect 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect value. The DATA MAP ACTION parameter in one of the repeating groups of the CCP structured field contains an incorrect 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 might 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 your system's diagnosis reference 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 *ccpidentifier* IS INVALID.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect value. The COMPARISON parameter in one of the repeating groups of the CCP structured field contains an incorrect 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK326S DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA MAP "*dataname*" SPECIFIED IN THE DATA MAP NAME PARAMETER OF CCP STRUCTURED FIELD *ccpidentifier* WAS NOT FOUND.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect 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 *dataname* 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 might 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 your system's diagnosis reference 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 *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 might 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 your system's diagnosis reference 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 *number* IS *parametervalue1*. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF *parametervalue2*.

Explanation: The NEXT LINE DESCRIPTOR IF REUSING DATA parameter in the Line Descriptor (LND)

structured field identified in this message has an incorrect 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK330I DATA IN A PAGEDEF RESOURCE IS NOT VALID: WHEN THE DATA START POSITION VALUE IS ADDED TO THE DATA LENGTH VALUE IN LND, RCD, OR XMD STRUCTURED FIELD NUMBER *number*, IT EXCEEDS THE FDS STRUCTURED FIELD SIZE VALUE OF *parametervalue*.

Explanation: The Use Fixed Data flag in byte 0 in the Line Descriptor (LND) structured field, in byte 11 in the Record Descriptor (RCD) structured field, or in byte 1 in the XML Descriptor (XMD) 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, RCD, or XMD structured field. The FDX, XMD, RCD, and LND structured fields are in the page definition.

The DATA START POSITION parameter in the LND, RCD, or XMD structured field indicates the offset of the first byte of data. The DATA LENGTH parameter specifies how many bytes of FDX are to be placed within the page. This error was caused when these two parameters specified more data than the FDX structured fields contain. 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK335S DATA IN A PAGEDEF RESOURCE IS INVALID: THE MEDIUM MAP "*mediummapname*" SPECIFIED IN THE MEDIUM MAP NAME PARAMETER OF CCP STRUCTURED FIELD *ccpidentifier* WAS NOT FOUND.

Explanation: The Conditional Processing Control (CCP) structured field has an incorrect 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 *mediummapname* 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK337I DATA IN A PAGEDEF RESOURCE IS NOT VALID: IN LND, RCD, OR XMD STRUCTURED FIELD NUMBER *number*, THE CONDITIONAL PROCESSING FLAG WAS SET BUT THE CONDITIONAL PROCESSING CONTROL IDENTIFIER WAS ZERO.

Explanation: In the Line Descriptor (LND), Record Descriptor (RCD), or XML Descriptor (XMD) structured field, the Conditional Processing flag had a value of B'1', indicating that the line data to be processed by this LND, RCD, or XMD 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, RCD, or XMD structured field is used to find one of the CCP structured fields in the current page definition. This parameter was set to 0, which is not a valid value if the Conditional Processing flag is on. The LND, RCD, XMD, and CCP structured fields are 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in

determining the source of the problem.

APK339I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE IDENTIFIER *identifier* SPECIFIED IN THE CONDITIONAL PROCESSING CONTROL IDENTIFIER PARAMETER IN LND, RCD OR XMD STRUCTURED FIELD NUMBER *number* WAS NOT FOUND.

Explanation: In the Line Descriptor (LND), Record Descriptor (RCD), or XML Descriptor (XMD) structured field, the Conditional Processing flag had a value of B'1', indicating that the line data to be processed by this LND, RCD, or XMD 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, RCD, or XMD structured field is used to find one of the CCP structured fields in the current page definition. However, the identifier specified in the LND, RCD, or XMD structured field identified in this message 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, RCD, XMD, and CCP structured fields are 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK340I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING PARAMETER VALUE IN LND, RCD OR XMD STRUCTURED FIELD NUMBER *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), Record Format Descriptor (RCD), or XML Descriptor (XMD) structured field has an incorrect

value. The value is greater than the COUNT parameter in the Line Descriptor Count (LNC) structured field in the current data map. The LNC, LND, RCD, and XMD 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK342I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING PARAMETER VALUE IN LND, RCD OR XMD STRUCTURED FIELD NUMBER *number* WILL CAUSE AN INFINITE LOOP.

Explanation: The NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING parameter in the Line Descriptor (LND), Record Format Descriptor (RCD), or XML Descriptor (XMD) structured field caused an infinite-loop condition. The LND, RCD, and XMD structured fields are 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK343I DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS NOT VALID: RELATIVE POSITIONING PLACED DATA OUTSIDE THE LOGICAL PAGE IN THE NEGATIVE Y DIRECTION. THE PRIOR AND CURRENT LND, RCD OR XMD STRUCTURED FIELD NUMBERS ARE *priornumber* AND *currentnumber*.

Explanation: When relative positioning is being used on a Line Descriptor (LND), Record Descriptor (RCD), or XML Descriptor (XMD) structured field, the relative position specified for the Y direction can be a negative value. The current LND, RCD, or XMD position (*priornumber*) defines the baseline position from which the relative offset of the current LND, RCD, or XMD is measured.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK344S DATA IN A PAGEDEF RESOURCE IS INVALID: THE NUMBER OF LND OR RCD STRUCTURED FIELDS DOES NOT MATCH THE VALUE SPECIFIED IN THE LNC STRUCTURED FIELD.

Explanation: The number of Line Descriptor (LND) or Record Descriptor (RCD) 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, RCD, and LNC structured fields are 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference 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 = recordnumber 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 incorrect. 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 might 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.

APK350S DATA IN A PAGEDEF RESOURCE IS INVALID: IN LND STRUCTURED FIELD NUMBER *number*, THE SHIFT-OUT CODED FONT LOCAL IDENTIFIER WAS NON-ZERO BUT THE GENERATE FONT CHANGE FLAG WAS NOT SET.

Explanation: In the Line Descriptor (LND) or Record Descriptor (RCD) structured field identified in this message, the Shift-Out Coded Font Identifier was

non-zero. The Generate Font Change flag should be set to indicate that the Primary Coded Font Local Identifier should be used whenever a shift-in code is processed. However, the Generate Font Change flag had a value of B'0'. The LND or RCD 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference 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 *number* DOES NOT MATCH THE LENGTH OF COMPARISON STRING PARAMETER VALUE IN CCP STRUCTURED FIELD *ccpidentifier*.

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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK356S DATA IN A PAGEDEF RESOURCE IS INVALID: A PAGE SEGMENT OR OVERLAY WAS REQUESTED IN THE LND OR RCD STRUCTURED FIELD *structuredfield*, BUT THE INLINE OR BASELINE POSITION VALUES WERE SPECIFIED FOR THE LND OR RCD.

Explanation: If any resource object-include triplets are specified in the LND structured field, bits 2 and 3 of bytes 0–1 in the LND structured field must both be set. If any resource object-include triplets are specified in the RCD structured field, bits 2 and 3 of bytes 11–13 in the RCD structured field must both be set.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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: None.

APK359I 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 might 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.

APK364I THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD INDICATES DIFFERENT SELECT INPUT SOURCE VALUES FOR THE FRONT AND BACK SIDES OF A DUPLEX SHEET.

Explanation: The Medium Modification Control (MMC) structured field referenced by the Medium Copy Count (MCC) structured field repeating groups specify different input source or media type local ID values, along with either tumble or normal duplex. This is an attempt to print the front and back sides of a sheet from different input bins.

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 might 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: None.

APK366I DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS NOT VALID: THE ORIENTATION USED WITH RELATIVE POSITIONING IS DIFFERENT THAN THE LAST ORIENTATION USED FOR PRINTING. THE PRIOR AND CURRENT LND, RCD OR XMD STRUCTURED FIELD NUMBERS ARE *priornumber* AND *currentnumber*.

Explanation: When relative positioning is being used on a Line Descriptor (LND), Record Descriptor (RCD), or XML Descriptor (XMD) structured field, the text orientation field of the current LND, RCD, or XMD (*currentnumber*) must match the text orientation field of the LND, RCD, or XMD (*priornumber*) that was last used for positioning data. The prior LND, RCD, or XMD position defines the baseline position from which the relative offset of the current LND, RCD, or XMD is measured.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK367I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE MAPPING OPTION SPECIFIED IN AN IOB STRUCTURED FIELD WITH LOCAL ID *identifier* IS NOT VALID OR UNSUPPORTED. THE IOB IS INCLUDED WITH LND, RCD, OR XMD STRUCTURED FIELD NUMBER *number*.

Explanation: The Include Object (IOB) structured field in error contained an Output Option value that is not valid, or the printer does not support the Output Option value. The IOB is included using the Line Descriptor (LND), Record Descriptor (RCD), or XML Descriptor (XMD) structured field identified in this message. The IOB, LND, RCD, and XMD 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 resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 resource with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK368S DATA IN A PAGEDEF RESOURCE IS INVALID: THE RESOURCE LOCAL ID *identifier* SPECIFIED IN THE EXTENDED RESOURCE LOCAL ID TRIPLET ON LND OR RCD STRUCTURED FIELD NUMBER *number* WAS NOT FOUND.

Explanation: In the Line Descriptor (LND) or Record Descriptor (RCD) structured field, and Extended Resource Local Identifier triplet specifies a local ID (*identifier*) of an Include Object (IOB) structured field that is to be used to include an object when this LND or RCD is used for printing. The identifier specified on the

LND or RCD does not match any of the IOB 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 resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 resource with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK369S AN *structuredfield* STRUCTURED FIELD IS ATTEMPTING TO INCLUDE AN UNSUPPORTED OBJECT CLASS.

Explanation: The Object Classification triplet (X'10') on a structured field is requesting a non-presentation object as the object class. Only presentation objects are allowed to be included through this 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 resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 resource with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK380S THE REGISTRATION ID (*identifier*) OF AN OBJECT CONTAINER RESOURCE, NAME *resourcenam* OR OBJECT OID *objectoid*, DOES NOT MATCH THE CORRESPONDING REGISTRATION ID FOR THE INVOKING JCL KEYWORD OR STRUCTURED FIELD.

Explanation: An object container resource was requested through a JCL keyword or an IOB or MDR structured field, but the Object Classification triplet in the Begin Object Container structured field did not have the corresponding registration ID. For a list of registration IDs and their assumed functions, refer to *Mixed Object Document Content Architecture Reference*. A value of *** means that the resource name or object OID was not specified.

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 container resource, ensure that the registration ID corresponds to the keyword used to invoke the resource or the registration ID specified in the Object Classification triplet specified on the IOB or MDR structured field.

System programmer response: If an IBM licensed program was used to create the structured fields for the object container that contains the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK381S DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE REGISTERED OBJECT ID IN THE OBJECT CLASSIFICATION TRIPLET ON A *structuredfield* STRUCTURED FIELD IS NOT SPECIFIED.

Explanation: The registered object ID is 0 in the Object Classification triplet. Object containers require a registered ID to be specified.

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, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might be an ACIF logic error. If you used a program to create the structured fields, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the structured field with the error, verify that the input to that program was valid.

APK384S DATA IN A FORMDEF RESOURCE IS INVALID: THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD INCLUDES CONFLICTING PRESENTATION SYSTEM SETUP ID VALUES.

Explanation: Multiple MMC structured fields referenced by the MCC structured field do not use the exact same set of Presentation System Setup ID values.

System action: ACIF stops processing the print data set.

User response: If you created the structured fields for the form definition, correct the MCC structured field. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the MCC has no errors, the error might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK385S DATA IN A FORMDEF RESOURCE IS INVALID: THE MODIFICATIONS SPECIFIED IN THE MMC STRUCTURED FIELD *structuredfield* INCLUDE UNPAIRED *keyword1* AND *keyword2* KEYWORDS.

Explanation: The keywords must be paired in the Medium Modification Control (MMC) structured field. This form definition has one or the other keyword but not both, or the keyword pairs are not adjacent. 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK386I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A REQUIRED TRIPLET WITH ID *identifier* WAS MISSING FROM AN *structuredfield* STRUCTURED FIELD.

Explanation: When the structured field is an IOB or

PPO and the identifier is X'4B', the x- or y-axis origin for object content or an object area size (X'4C') triplet was specified on the IOB or PPO, but no measurement unit (X'4B') triplet was specified. The structured field is contained in a print data set if it is a PPO. The structured field is contained in a print data set, overlay, or page definition if it is an IOB. When the structured field is an IOB and the identifier is X'22', the Extended Resource Local Identifier (X'22') triplet is required when the IOB structured field is contained in a 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 placed the IOB structured field in the print data set or resource, 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 IOB structured field in the print data set or resource, contact your system programmer.

System programmer response: If an IBM licensed program was used to place the IOB or PPO structured field in the print data set or resource, verify that the input to that program was valid.

APK387S DATA IN AN INPUT RECORD IS NOT VALID: A PARAMETER IN AN *structuredfield* STRUCTURED FIELD CONTAINS UNACCEPTABLE DATA.

Explanation: One of the parameters in the structured field is not valid. If the structured field is IOB, one of these caused the problem:

- The object type specified is not supported or is not valid.
- The x or y offset of the object area or the rotation value are not explicitly specified when the reference coordinate system is set to X'00'.

If the structured field is PPO, one of these caused the problem:

- The object type specified is not supported or is not valid.
- The x or y offset of the object area is not valid.

System action: ACIF stops processing the input data set.

User response: If you placed the IOB structured field in the input data set or overlay, correct the error and resubmit the ACIF job. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If you used a program to place the IOB structured field in the print data set or overlay, contact your system programmer.

System programmer response: No response is necessary.

APK388S DATA IN A PAGE SEGMENT IS INVALID: *structuredfield* STRUCTURED FIELD IS NOT ALLOWED IN A PAGE SEGMENT INCLUDED WITH AN IOB.

Explanation: Only MO:DCA-P page segments are allowed to be included with an IOB structured field. MO:DCA-P page segments cannot contain IM1 image or PTOCA data.

System action: ACIF stops processing the input data set.

User response: If you placed the IOB structured field in the input data set or overlay, correct the error and resubmit the ACIF job. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If you used a program to place the IOB structured field in the print data set or overlay, contact your system programmer.

System programmer response: No response is necessary.

APK389S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE FONT RESOLUTION AND METRIC TECHNOLOGY TRIPLET SPECIFIES AN INCORRECT VALUE.

Explanation: There is an incorrect value specified for the metric technology, the unit base, or the units per unit base field in the Font Resolution and Metric Technology triplet (X'84'). The triplet is specified on a Map Coded Font (MCF) structured field, which can be in a print data set or overlay.

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 job to ACIF. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error might 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 with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK390S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE structuredfield STRUCTURED FIELD CONTAINS A triplet TRIPLET THAT HAS AN INVALID VALUE. THE INVALID VALUE STARTS IN BYTE byte OF THE TRIPLET.

Explanation: An incorrect value was specified for a field that starts in byte offset of the triplet identified in this message. The triplet is specified on the structured field identified in this message.

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. 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 might 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 with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK392I DATA IN A FORMDEF RESOURCE IS NOT VALID: THE SCOPE VALUE IN THE MFC IS NOT VALID. EITHER DOCUMENT LEVEL OR PRINT FILE LEVEL FINISHING WAS SPECIFIED IN THE MEDIUM MAP OR MEDIUM LEVEL FINISHING WAS SPECIFIED IN THE DEG.

Explanation: Either a document environment group (DEG) or a medium map in the current form definition contains a Medium Finishing Control (MFC) structured field with an incorrect value specified for the scope.

System action: The MFC is ignored and processing continues. ACIF might issue additional messages identifying the processing environment in which the error occurred.

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* and *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the structured field has no error, the problem might 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, report the problem to your IBM service representative.

APK395I A FORMDEF RESOURCE REQUESTED A MEDIA EJECT CONTROL TO THE NEXT BACK-SIDE AND DUPLEX=NO WAS SPECIFIED ON THE OUTPUT STATEMENT.

Explanation: When a media eject control to the next back-side is specified in a form definition, the DUPLEX=NO keyword on the OUTPUT statement cannot be used to change from duplex (specified in the form definition) to simplex. The reason is that an incompatible request is being made; you cannot eject to the next back-side when simplexing.

When a media eject control to the next back-side is specified in the form definition and the form definition requests normal or tumble duplex, the only valid option for the duplex keyword is to specify either DUPLEX=NORMAL or DUPLEX=TUMBLE on the OUTPUT statement.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: Resubmit the job without requesting the duplex keyword on the OUTPUT statement.

System programmer response: None.

APK396I DATA IN A FORMDEF RESOURCE IS INVALID: THE OUTPUT BIN SELECTION VALUE IN MMC STRUCTURED FIELD, ID identifier, IS NOT ACCEPTABLE.

Explanation: In the Medium Modification Control (MMC) structured field whose identifier is specified in the message text, the output bin selection parameter value was not valid. 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK397I THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD INDICATES DIFFERENT OUTPUT BIN VALUES FOR THE FRONT AND BACK SIDES OF A DUPLEX SHEET.

Explanation: The Medium Modification Control (MMC) structured fields referenced by the Medium Copy Count (MCC) structured field repeating groups specify different output bin values along with either tumble or normal duplex. This is an attempt to place the front and back sides of a sheet into different output bins.

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 might 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 the error involves separator pages or the message data set, use the information provided in the User Response section to correct the error.

APK398I DATA IN A FORMDEF RESOURCE IS INVALID: THE SET OF MODIFICATIONS SPECIFIED IN THE MMC STRUCTURED FIELD, ID *identifier*, INCLUDES DUPLICATE CONFLICTING VALUES FOR THE *keyword* KEYWORD.

Explanation: The Medium Modification Control (MMC) structured field contains duplicate conflicting values for the keyword identified in the message text. The MMC structured field is in the form definition.

System action: ACIF issues this message and continues processing, ignoring the duplicate keyword.

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* and *Advanced Function Presentation: Programming Guide and Line Data Reference* for more information about the structured field. If the MMC has no errors, the error might 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

object with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference 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 stops.

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 stops.

User response: Use single quotes before and after the attribute name in the parameter.

System programmer response: No response is necessary.

APK402S THE PARAMETER "*parameter*" IS INVALID.

Explanation: A parameter that is not valid for ACIF was specified.

System action: ACIF stops.

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:

Type	Resource
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 stops.

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 n** parameter.

System action: ACIF stops.

User response: Correct the **INDEX n** parameter and resubmit the job.

System programmer response: No response is necessary.

APK405S A VALUE OF "*value*" IS INVALID FOR PARAMETER "*parameter*".

Explanation: The value supplied for a parameter is not valid.

System action: ACIF stops.

User response: Correct the parameter value and resubmit the job.

System programmer response: No response is necessary.

APK406S PARAMETER "*parameter*" HAS TOO MANY DATA SETS SPECIFIED.

Explanation: More than eight data sets have been supplied for the parameter.

System action: ACIF stops.

User response: Correct the number of data sets and resubmit the job.

System programmer response: No response is necessary.

APK407S A RESTYPE PARAMETER OF "*value*" 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 stops.

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 *storagerequestsize*, GETMAIN RETURN CODE *returncode*.

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 stops.

User response: Increase the REGION size and resubmit the job.

System programmer response: To interpret the GETMAIN return code, refer to the documents about application development macros for your operating system.

APK409I A DDNAME FOR *parameter* WAS NOT SUPPLIED. "*default*" WAS USED.

Explanation: No DD name was specified for either the **MSGDD** or the **PARMDD** parameter.

System action: If the missing DD name was **MSGDD**, the DD name assigned to **SYSPRINT** was used. If the missing DD name was **PARMDD**, the DD name assigned to **SYSIN** was used.

User response: If the DD name used was not acceptable, specify a DD name for the parameter and submit the job again.

System programmer response: No response is necessary.

APK410S AN ACIF STORAGE REQUEST WAS UNSUCCESSFUL - REQUEST SIZE *storagerequestsize*, *requesttype* RETURN CODE *returncode*.

Explanation: An unsuccessful attempt has been made to obtain or free ACIF subpool storage. If you requested indexing on values that do not occur in the data, ACIF often runs out of storage trying to find the second page. For example, if you specify **TRIGGER1=*,1,X'F1'** but your data does not contain any **X'F1'** carriage controls, ACIF can run out of storage. This error message returns the following information:

- Storage request size
- Request type
- Return code

System action: ACIF stops.

| **User response:** If you requested indexing, verify that
| your data matches the values you specified on the
| TRIGGER parameter.

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 *action* THE DDNAME
ddname, RETURN CODE *returncode*.**

Explanation: The file I/O macro made an unsuccessful attempt to read from, write to, or close the named DD. The return codes are:

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Incorrect data buffer address
4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
51	Length exceeds maximum
310	File format not valid

System action: ACIF stops.

User response: Use the information provided in the return code to correct the problem.

System programmer response: No response is necessary.

**APK412I MODULE *modulename* HAS RETURNED
WITH RETURN CODE *returncode*.**

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 takes place in higher level modules if required.

Note: A return code of 999 indicates that the user's input exit returned a zero length record.

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 *action* RESOURCE FILE
"*ddname*", RESOURCE MEMBER NAME
"*membername*" FAILED, RETURN CODE
returncode.**

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 takes place in higher level modules if required. If you received this message for a Data Object Font (DOF) Descriptor triplet, the resource name is not a typical member name and the name is translated to ASCII or EBCDIC for display purposes. If the resource name cannot be translated, it is presented as a hexadecimal value.

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Incorrect data buffer address
4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
51	Length exceeds maximum
310	File format not valid

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** *paths* **HAVE BEEN SPECIFIED**
 FOR *libraryname*:

Explanation: This message is issued before message APK417I and shows the resource type the data set or file type is specified for.

System action: None.

User response: No response is necessary.

System programmer response: No response is necessary.

APK417I *path: 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 999999999 documents.

System action: ACIF stops.

User response: Break the job up into a smaller number of documents.

System programmer response: No response is necessary.

APK419S **USER** *exittype* **EXIT** *programname*
 RETURNED CODE *returncode*.

Explanation: An input, output, or resource user exit program has returned a non-zero return code.

Note: A return code of 999 indicates that the user's input exit returned a zero length record.

System action: ACIF stops.

User response: Correct the error in the exit program and resubmit the job.

System programmer response: No response is necessary.

APK420S **AN ERROR OCCURRED WHILE**
 ATTEMPTING TO OPEN "*dataset*".
 RETURN CODE *returncode*.

Explanation: An attempt to open a data set failed. This message is informational and further action takes place in higher level modules if required. If you received this message for a Data Object Font (DOF) Descriptor triplet, the resource name is not a typical member name and the name is translated to ASCII or EBCDIC for display purposes. If the resource name cannot be translated, it is presented as a hexadecimal value.

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Incorrect data buffer address
4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
32	ACIF message catalog not found in paths specified by NLSPATH environment variable. ACIF uses default message catalog file /usr/lib/nls/msg/en_US/acif.cat
36	Default message catalog not accessible. Check permissions
51	Length exceeds maximum
310	File format not valid

System action: ACIF stops.

User response: Use the information provided in the return code to correct the problem.

System programmer response: No response is necessary.

APK421S **AN ERROR OCCURRED WHILE**
 ATTEMPTING TO CLOSE "*dataset*".
 RETURN CODE *returncode*.

Explanation: An attempt to close a data set failed. This message is informational and further action takes place in higher level modules if required.

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Incorrect data buffer address

4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
32	ACIF message catalog not found in paths specified by NLSPATH environment variable. ACIF uses default message catalog file /usr/lib/nls/msg/en_US/acif.cat
36	Default message catalog not accessible. Check permissions
51	Length exceeds maximum
310	File format not valid

System action: ACIF stops.

User response: Use the information provided in the return code to correct the problem.

System programmer response: No response is necessary.

APK422S AN ERROR OCCURRED WHILE ATTEMPTING TO READ "dataset". RETURN CODE *returncode*.

Explanation: An attempt to read a data set failed. This message is informational and further action takes place in higher level modules if required.

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Incorrect data buffer address
4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
32	ACIF message catalog not found in paths specified by NLSPATH environment variable. ACIF uses default message catalog file /usr/lib/nls/msg/en_US/acif.cat
36	Default message catalog not accessible. Check permissions
51	Length exceeds maximum
310	File format not valid

System action: ACIF stops.

User response: Use the information provided in the return code to correct the problem.

System programmer response: No response is necessary.

APK423S AN ERROR OCCURRED WHILE ATTEMPTING TO WRITE "dataset". RETURN CODE *returncode*.

Explanation: An attempt to write a data set failed. This message is informational and further action takes place in higher level modules if required.

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Incorrect data buffer address
4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
32	ACIF message catalog not found in paths specified by NLSPATH environment variable. ACIF uses default message catalog file /usr/lib/nls/msg/en_US/acif.cat
36	Default message catalog not accessible. Check permissions
51	Length exceeds maximum
310	File format not valid

System action: ACIF stops.

User response: Use the information provided in the return code to correct the problem.

System programmer response: No response is necessary.

APK424I PARAMETER "RESFILE=PDS" IS ONLY VALID UNDER MVS, DEFAULTING TO "RESFILE=SEQ".

Explanation: The supplied value for the **RESFILE** parameter is valid only for z/OS; it is incorrect for other operating systems.

System action: ACIF produces a sequential resource file.

User response: No response is necessary.

System programmer response: No response is necessary.

APK425S USER *type* EXIT "*program*" WAS NOT FOUND.

Explanation: The input, output, or resource user exit program named on the exit's DD parameter does not exist.

System action: ACIF stops.

User response: Correct your exit program and resubmit the job.

System programmer response: No response is necessary.

APK426S PARAMETER MISMATCH: RESTYPE *type* SPECIFIED = YES, 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 stops.

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 "*filename*" "*filetype*" "*filemode*" FOR "*ddname*" WILL BE USED.

Explanation: An incorrect *filename* was supplied. The defaults listed are 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. If you received this message for a Data Object Font (DOF) Descriptor triplet, the resource name is not a typical member name and the name is translated to ASCII or EBCDIC for display purposes. If the resource name cannot be translated, it is presented as a hexadecimal value.

System action: ACIF stops.

User response: Contact IBM Service.

System programmer response: No response is necessary.

APK430I DATA ENCODING PARAMETER *parm* DOES NOT MATCH USER DATA TYPE *datatype* FROM THE PAGE DEFINITION.

Explanation: ACIF uses the DATAENCODING and NEWLINE parameters to break the line data into records, while using the UDTtype value for formatting the text. In order for ACIF to properly process the data, the DATAENCODING and UDTtype parameters in the page definition must specify the same values. However, different values were specified for the DATAENCODING and UDTtype parameters.

System action: ACIF continues processing the data set but unexpected results might occur in the formatted output.

User response: Check the actual encoding of the data and make sure that you are specifying the correct values for both the DATAENCODING parameter on the ACIF command and the UDTtype parameter on the PAGEDEF command in your page definition.

System programmer response: No response is necessary.

APK435W THE *ddname* DD STATEMENT SPECIFIED FOR *parameter* IS MISSING.

Explanation: An ACIF DD parameter specified a DD name that was not specified in the JCL (z/OS or VSE) or FILEDEF statement (VM).

System action: ACIF stops.

User response: Ensure that the ACIF parameter specifies a DD name 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 incorrect INDEX number specified. The INDEXn range is 1–8.

System action: ACIF stops.

User response: Correct the resource and resubmit the job.

System programmer response: No response is necessary.

APK437S (TYPE=FLOAT) MAY NOT BE SPECIFIED FOR TRIGGER1.

Explanation: The 'TYPE=FLOAT' subparameter is not valid for TRIGGER1.

System action: ACIF stops.

User response: Correct the parameter and rerun ACIF.

System programmer response: No response is necessary.

APK438S THE VALUE SPECIFIED FOR *parameter1* CONFLICTS WITH THE VALUE SPECIFIED FOR *parameter2*.

Explanation: The value specified for the first parameter conflicts with the value specified for the second parameter.

System action: ACIF stops.

User response: Correct the parameters and rerun ACIF.

System programmer response: No response is necessary.

APK440I ACIF AT *aparnumber* HAS COMPLETED NORMALLY WITH RETURN CODE *returncode*.

Explanation: ACIF at the maintenance level indicated by the APAR number 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 AT *aparnumber* HAS COMPLETED ABNORMALLY WITH RETURN CODE *returncode*.

Explanation: ACIF at the maintenance level indicated by the APAR number has completed with one of these return codes:

Code	Description
4	Warning; processing continues
8	Error; processing stops
12	Severe error; processing stops
16	Fatal error; processing stops

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.

APK442S ACIF HAS BEEN INVOKED WITHOUT ANY PARAMETERS.

Explanation: ACIF needs a minimum number of parameters in order to function.

System action: ACIF stops.

User response: Specify the INPUTDD, FORMDEF, CC, and PAGEDEF parameters.

System programmer response: No response is necessary.

APK443S A BEGIN COLUMN SPECIFICATION FOR FIELD_{*n*} IS <= 0. SUCH A SPECIFICATION IS ONLY VALID WHEN (BASE=TRIGGER) IS ALSO SPECIFIED.

Explanation: FIELD_{*n*} was specified with a column offset less than or equal to zero, but (BASE=TRIGGER) was not also specified. Negative column offsets in a FIELD specification are only valid when (BASE=TRIGGER) is also specified.

System action: ACIF stops.

User response: Correct the the ACIF FIELD_{*n*} parameter specification and resubmit the job.

System programmer response: No response is necessary.

APK444S MULTIPLE COLUMNS WERE SPECIFIED FOR FIELD_{*n*} WHICH IS DEFINED WITH (BASE=TRIGGER). ONLY ONE COLUMN MAY BE SPECIFIED WHEN A FIELD IS DEFINED WITH (BASE=TRIGGER).

Explanation: FIELD_{*n*} was specified with multiple columns and (BASE=TRIGGER). Only one column can be specified for a field that is also specified with (BASE=TRIGGER).

System action: ACIF stops.

User response: Correct the ACIF FIELD_{*n*} parameter specification and resubmit the job.

System programmer response: No response is necessary.

APK445S INDEX_{*n*} WHICH IS DEFINED AS EITHER (TYPE=PAGERANGE) OR (TYPE=GROUPRANGE) INCLUDES FIELD_{*n*} WHICH IS DEFINED AS (BASE=TRIGGER). THIS COMBINATION IS INVALID.

Explanation: INDEX_{*n*} was specified as (TYPE=PAGERANGE) or (TYPE=GROUPRANGE) and with a FIELD_{*n*} that was defined as (BASE=TRIGGER). This combination is not supported.

System action: ACIF stops.

User response: Correct the ACIF parameters and resubmit the job.

System programmer response: No response is necessary.

**APK446S USE OF FIELD_n BY INDEX_n IS
INVALID. ONLY ONE FIELD IS
ALLOWED IN AN INDEX DEFINED AS
(TYPE=PAGERANGE) OR
(TYPE=GROUPRANGE).**

Explanation: More than one field was specified for INDEX_n, which is defined as either (TYPE=PAGERANGE) or (TYPE=GROUPRANGE). This is not valid.

System action: ACIF stops.

User response: Correct the ACIF parameters and resubmit the job.

System programmer response: No response is necessary.

**APK447S THE LENGTH, *length1*, OF OFFSET
PAIR *pair* FOR FIELD_n DOES NOT
EQUAL THE LENGTH, *length2*,
SPECIFIED FOR FIELD_n.**

Explanation: The length of a begin-end pair, specified by the offset keyword of a field, does not match the length of the field. This is not valid; the lengths must be equal.

System action: ACIF stops.

User response: Correct the ACIF parameters and resubmit the job.

System programmer response: No response is necessary.

**APK448S INDEXING WAS REQUESTED, BUT
NEITHER "TRIGGER1" NOR ANY
"FIELD" WAS SATISFIED WITHIN THE
FIRST PAGE.**

Explanation: Indexing was requested, but the first INDEX satisfier was outside the range of pages specified in the INDEXSTARTBY parameter.

System action: ACIF stops.

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# *number*
INPUT RECORD# *number*.**

Explanation: The FIELD_n value specified on the INDEX_n parameter references an area that is outside the length of the requested record.

System action: ACIF stops.

User response: Correct the parameters and resubmit the job.

System programmer response: No response is necessary.

**APK450S A REQUIRED ACIF PARAMETER
parametername WAS NOT FOUND IN
THE PARAMETER FILE.**

Explanation: A required ACIF parameter was not found in the parameter file.

System action: ACIF stops.

User response: Add the missing parameter to the parameter file and resubmit.

System programmer response: No response is necessary.

**APK451S FILE *action* ERROR DURING *ddname*
PROCESSING. SVC 99 ERROR *error*
INFORMATION CODE *code*.**

Explanation: An error occurred during the allocation, concatenation, or outadd of AFP resource libraries.

System action: ACIF stops.

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; then, determine the appropriate response. Refer to your operating system's authorized assembler language programs document for information about the SVC 99.

**APK452S A *trigger* NUMBER OF *number* IS
INVALID FOR *parameternumber*.**

Explanation: The trigger or record number specified in the FIELD_n or INDEX_n parameter is not valid.

System action: ACIF stops.

| **User response:** Triggers used in field definitions must
| be defined. Make sure that you have specified a
| TRIGGER parameter before using that trigger number
| on a FIELD parameter. After you correct the parameter,
| run ACIF again.

System programmer response: No response is necessary.

APK453S **THE *parameternumber* LENGTH OF *length* IS GREATER THAN THE ALLOWED MAXIMUM OF *maxlength*.**

Explanation: The combined length of all of the FIELD*n* values on an INDEX*n* parameter is too long.

System action: ACIF stops.

User response: Check the FIELD*n* and INDEX*n* parameters to find where this happens. Correct the parameter and resubmit the job.

System programmer response: No response is necessary.

APK454S **A VALUE OF *value* IS INVALID FOR *parameternumber*.**

Explanation: A FIELD*n* parameter value contains incorrect characters.

System action: ACIF stops.

User response: Correct the parameter value and resubmit the job.

System programmer response: No response is necessary.

APK455S **FIELD*n* USED BY INDEX*n* WAS NOT DEFINED.**

Explanation: An INDEX*n* parameter referred to a FIELD*n* that was not defined in the parameter file.

System action: ACIF stops.

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 stops.

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 TRIGGER*n* parameters are present.

System action: ACIF stops.

User response: If no indexing is required, delete all TRIGGER*n* parameters 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 *value* HAS BEEN SUPPLIED FOR *parameternumber*.**

Explanation: The supplied TRIGGER*n* value was not a literal.

System action: ACIF stops.

User response: Correct the parameters and resubmit the job.

System programmer response: No response is necessary.

APK459S **INDEX NEEDED FOR THE GROUPNAME WAS NOT FOUND.**

Explanation: The index used for the GROUPNAME contained a field that was based on a floating trigger; however, the trigger was not found. Therefore, there is no value for the GROUPNAME. INDEX1 is used for the GROUPNAME by default.

System action: ACIF stops.

User response: Use the GROUPNAME parameter to specify an index that does not contain a field based on a floating trigger.

System programmer response: No response is necessary.

APK460S **TRIGGERS SATISFIED, BUT INDEXES WERE INCOMPLETE AT END-OF-FILE.**

Explanation: The TRIGGER*n* parameters specified in the parameter file were met, but the end of the file was reached before the INDEX*n* parameters were located.

System action: ACIF stops.

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*n* has been supplied, but all INDEX*n* values were literals.

System action: ACIF stops.

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 n parameter, but the input file contains indexing structured fields. ACIF cannot index a file that is already indexed.

System action: ACIF stops.

User response: If you want to create an index object file for the input file, remove all TRIGGER n parameters from the ACIF parameter file and resubmit the job.

System programmer response: No response is necessary.

APK463S INDEX n USED BY THE GROUPNAME PARAMETER WAS NOT DEFINED OR WAS INVALID.

Explanation: The INDEX n specified by the GROUPNAME parameter was not defined or the index contained a field that was based on a floating trigger. When the GROUPNAME parameter is not used, INDEX1 is used by default.

System action: ACIF stops.

User response: Correct the parameters and resubmit the job.

System programmer response: No response is necessary.

APK464S "token1" WAS SPECIFIED WHEN "token2" EXPECTED.

Explanation: The syntax of the parameter printed above this message was incorrect.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the value of the parameter and rerun ACIF.

APK465S INVALID TOKEN "token" RECEIVED.

Explanation: The token identified in the message was not expected in the parameter listed above the message.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the value of the parameter and rerun ACIF.

APK466S A SUB-PARAMETER OF "subparameter" IS INVALID FOR PARAMETER "parameter".

Explanation: The named subparameter is not supported on the parameter listed above the message.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the value of the parameter and rerun ACIF.

APK467S THE NUMBER "number" IS NOT SUPPORTED FOR parameter.

Explanation: An incorrect number was specified on a FIELD n , INDEX n , or TRIGGER n parameter keyword.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the parameter keyword so that the number is within the allowed range for that parameter and rerun ACIF.

APK468S THE INPUT BUFFER IS TOO SMALL FOR THE PARAMETER VALUE "value".

Explanation: The named value was too long for the ACIF internal input buffer.

System action: ACIF stops.

User response: Use your local problem reporting system to report the error.

APK469S THE LENGTH OF THE VALUE "value" EXCEEDS THE MAXIMUM ALLOWED LENGTH FOR THE parameter PARAMETER.

Explanation: The length of the named value exceeds the maximum length.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the value so that its length is within the maximum for that parameter and rerun ACIF.

APK470S WHICH BEGINS AT OFFSET offset FOR A LENGTH OF length.

Explanation: This message is issued following a message that contains the cause of the error.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the value so that its length is within the maximum for that parameter and rerun ACIF.

APK471S THE NUMBER OF FIELD VALUES ON THE INDEX PARAMETER EXCEEDED THE MAXIMUM ALLOWED.

Explanation: There were too many FIELD*n* values specified for the INDEX*n* parameter printed above this message.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Remove the extra FIELD*n* values from the INDEX*n* parameter and rerun ACIF.

APK472S THE NUMBER OF VALUES SPECIFIED FOR THE *parameter* PARAMETER EXCEEDED THE MAXIMUM ALLOWED.

Explanation: Too many values were specified for the named parameter.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Consult the ACIF manual for the maximum number of values for this parameter, correct the parameter, and rerun ACIF.

APK473S RECORDRANGE SUB-PARAMETER ALLOWED ONLY IF RECORD VALUE IS ''.**

Explanation: The RECORDRANGE subparameter is only valid on a TRIGGER*n* parameter if the record value was specified as '**'.

System action: ACIF stops.

User response: Either specify an '**' for the record value or remove the RECORDRANGE from the TRIGGER parameter.

APK474S END-OF-FILE ENCOUNTERED BEFORE CLOSING QUOTE FOUND FOR "*value*".

Explanation: The end of the parameter file was found before the closing quote for a literal value.

System action: ACIF stops.

User response: Ensure the literal value is enclosed in quotes and rerun ACIF.

APK475S THE HEX STRING "*hexstring*" IS NOT VALID.

Explanation: The value specified was not a valid hex string.

System action: ACIF continues processing the parameter file, but does not process the report file.

User response: Correct the hex string and rerun ACIF.

APK476S THE LENGTH OF THE NUMERIC VALUE "*value*" IS INVALID.

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.

APK478I UNABLE TO SAVE FONT *filename* TO RESOURCE FILE.

Explanation: RESTYPE=FONT or RESTYPE=ALL was requested, but the font shown in the message has the embed flag set "off" in the Resource Access Table (RAT). Fonts with an embed flag set off cannot be placed inline or saved in the output resource library.

System action: ACIF skips the specified font and continues processing the page.

User response: This message is informational and is based on the embed flag setting in the RAT that is created when the font is installed. If you need to save this font in the resource file, contact your system programmer to set the embed flag on.

System programmer response: In the Font Installer for AFP Systems, find the font in its font library, select embed for the font, and then rerun the Font Installer to update the RAT. If embed was already selected for the font, contact IBM Service for assistance.

APK479I REQUESTED DATA OBJECT FONT *filename* NOT FOUND, RETURN CODE *nn*.

Explanation: A font was requested with a Map Data Resource (MDR) structured field in the input file or page definition, but the font could not be accessed by the Resource Access Table (RAT).

System action: ACIF stops processing.

User response: The requested font needs to be installed on the system in one of the directories specified with the USERPATH or FONTPATH parameter. The return code indicates why the font was not found and gives the action the user should perform:

Code Description and Action

- | | |
|-----------|--|
| 10 | Attempt to assign storage failed. |
| 20 | There was no RAT found in the paths specified with the USERPATH or FONTPATH parameter. Make sure a correct set of paths is specified for these parameters. Contact your system programmer to verify that the font is installed on your system. |

- 30** RAU handle is null. Contact your system programmer.
- 40** Font is not found. There was no entry for the requested font in the RAT. Make sure you have specified the correct set of paths to search. If the correct paths are specified, contact your system programmer to install the font into the correct directory and update the RAT.
- 50** Conversion information is missing. The font might not be found if the MDR structured field specifies the font encoding by CPGID/GCSGID names that cannot be mapped to a CCSID, or if the font file name cannot be converted to ASCII or EBCDIC for access on your system. If the font name cannot be converted or mapped, the font name is presented as a hexadecimal string. Contact your system programmer for assistance in analyzing this return code.
- 70** Unable to convert code page or character set to CCSID. Same action as return code 50.
- 120** ICONV open request has an error. Same action as return code 50.
- 130** ICONV conversion has an error. Same action as return code 50.

System programmer response: Make sure that both the font and the RAT are installed in the correct directory. If not, use the Font Installer for AFP Systems to install the font in the correct directory and build the RAT entry. If the font files and the RAT have been installed correctly, contact IBM Service for assistance.

APK499I INTERNAL ERROR IN MODULE *module* AT FUNCTION *function*.

Explanation: An internal error has occurred.

System action: ACIF stops.

User response: This message might result if your job needs more than 512 fonts. For double-byte fonts you must count one character set and one code page for each section in the font when calculating the number of fonts requested. If your job does not need more than 512 fonts, note the module and function mentioned in the message and then inform IBM Service that you have received this message.

APK532S A resource WITH A MEMBER NAME (*membername*) WAS NOT FOUND OR WAS INVALID - RETURN CODE *returncode*.

Explanation: The requested form definition, page definition, page segment, medium overlay, or setup file does not exist in any of the available paths. If the form definition member name is blank, you did not specify a form definition, which is required.

Return

Code	Description
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero
3	Incorrect data buffer address
4	Address not word aligned
6	Incorrect FILE_CB@
7	Incorrect MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Incorrect record number
12	End of file detected
13	Disk or PDS directory is full
14	RECFM not valid
20	Incorrect file ID
28	File not found
51	Length exceeds maximum
310	File format not valid

Reason

Code	Description
1	Resource name missing
2	File system open error
3	File system close error
4	File system read error
6	Resource type error
7	File system write error
8	Indexer error
9	Message write error

System action: ACIF stops.

User response: Correct the parameters and specify a form definition parameter if one was not specified. Run ACIF again.

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:

Code	Error Text
04	Inactive z/OS GTF
08	Incorrect length = xxxx
0C	Incorrect data address = xxxx
10	Incorrect FID = xx
14	Incorrect EID = xx
18	No GTF buffer space
1C	Incorrect parameter address = xxxx
20	Data paged out
xx	Unknown GTF return code

System action: The action depends on the return code; ACIF might or might 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 your operating system's service aids logic documents for more information about the return codes.

APK900S MISSING DAT POINTER IN CCM.

Explanation: An internal error has occurred in ACIF.

System action: ACIF stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

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 stops.

User response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK915S ACIF REQUESTED CODE PAGE
DEALLOCATION AS WELL AS CODE
PAGE PROCESSING.**

Explanation: An internal error has occurred in ACIF.

System action: ACIF stops.

User response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK916S ACIF REQUESTED ACTIVATION OF AN
OUTLINE FONT CHARACTER SET, BUT
DOES NOT SUPPORT OUTLINE
FONTS.**

Explanation: An internal error has occurred in ACIF.

System action: ACIF stops.

User response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK917S ACIF REQUESTED ACTIVATION OF A
FONT RESOURCE, BUT THE GLOBAL
NAME WAS NOT PROVIDED OR HAD
AN INCORRECT LENGTH.**

Explanation: An internal error has occurred in ACIF.

System action: ACIF stops.

User response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK918S NO FREQUENT FONT TABLE OR FGID
LOOK ASIDE TABLE WAS PROVIDED
TO *modulename*.**

Explanation: An internal error has occurred in ACIF.

System action: ACIF stops.

User response: Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK919S THE CCM COMPONENT OF ACIF HAS
USED UP ITS OBJECT STACK AREA IN
modulename.**

Explanation: The common conversion machine (CCM) component of ACIF has run out of its object stack area. This could be a data stream error or a logic error. A begin structured field must have a matching end structured field following it in the data stream. If this requirement is not met, the CCM can run out of its object stack area.

System action: ACIF stops.

User response: Check the data stream to make sure each begin structured field has a matching end structured field following it. If this is not true, correct the data stream and resubmit the job to ACIF. If the data stream meets the begin structured field requirement, this message indicates an internal logic error. Contact IBM Service and inform them that you have received this message indicating an internal error.

**APK920S ALL THE INPUT BINS ON YOUR
PRINTER ARE EITHER DISABLED OR
ARE INSERTER BINS.**

Explanation: This abend is issued by module APKMSGEX. All bins reported back from the printer are either disabled or are inserter bins. As a result, there are no bins available for printing and the current data set cannot be printed.

System action: ACIF processing stops abnormally.

APK921S NO RECORD LENGTH WAS PASSED TO CCM WHEN PROCESSING AN OBJECT CONTAINER RESOURCE.

Explanation: This abend is issued by module APRMSGEX. No record length was passed to common conversion machining (CCM) when processing an object container resource. This is a logic error.

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.

System programmer response: This ACIF abend reason code indicates a logic error. Contact your service representative in the IBM Support Center or use your electronic link with IBM service for assistance regarding this error code.

APK922S NO DATA OBJECT FONT (DOF) DESCRIPTOR TRIPLET INFORMATION WAS PROVIDED FOR A DATA OBJECT FONT ACTIVATION REQUEST.

Explanation: This abend is issued by module APKMSGEX — No Data Object Font (DOF) Descriptor triplet information was provided for a data object font activation request.

System action: ACIF processing stops abnormally.

APK923S NO OID WAS SPECIFIED FOR A DATA OBJECT RESOURCE OR DATA OBJECT FONT COMPONENT ACTIVATION REQUEST.

Explanation: This abend is issued by module APKMSGEX — No OID was specified for a data object resource or data object font component activation request.

System action: ACIF processing stops abnormally.

APK2000S THE REGISTERED OBJECT TYPE ID (*identifier*) OF AN OBJECT CONTAINER RESOURCE, NAME *resourcename* OR OBJECT OID *objectoid* IS NOT VALID OR IS NOT SUPPORTED BY ACIF.

Explanation: The registration ID specified for an object container is not supported by the printer. The registration ID is specified in the Object Classification triplet on an IOB, BOC, or BRS structured field. A value of *** for the resource name or object OID means that it was not specified.

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 container resource, ensure that the registration ID is correct. If the registration ID is correct, submit the print job to a printer that supports this object

type. For more information about what object types are supported by your printer, refer to your printer documentation.

System programmer response: If an IBM licensed program was used to create the structured fields, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK2003S DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD *structuredfield* CONTAINED AN EXTENDED RESOURCE LOCAL IDENTIFIER VALUE THAT WAS USED IN A PREVIOUS STRUCTURED FIELD OF THE SAME TYPE.

Explanation: More than one structured field used the same Extended Resource Local Identifier value for different resources of the same type. The Extended Resource Local Identifier is specified by using the Extended Resource Local Identifier (X'22') triplet on the structured field. The structured field that attempted to use the same Extended Resource Local Identifier value is identified in the message.

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 resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 form definition with the error, verify that the input to that program was valid. If the input was valid, refer to your system's diagnosis reference for assistance in determining the source of the problem.

APK2004S DATA IN A FORMDEF RESOURCE IS INVALID: MEDIA TYPE LOCAL IDENTIFIER IN MMC STRUCTURED FIELD, ID *identifier* WAS NOT FOUND IN THE STRUCTURED FIELD.

Explanation: The Media Type local ID in the Medium Modification Control (MMC) structured field was not present in the Map Media Type (MMT) structured field. The MMC and MMT structured fields are 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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2005S DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS NOT VALID: RECORD IDENTIFIER *identifier* COULD NOT BE FOUND WITHIN THE RCD STRUCTURED FIELDS.

Explanation: The record identifier specified in an input record could not be matched to a Record Descriptor (RCD) structured field in the current data map. The RCD structured field is 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 form definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2007S DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE FIELD RCD POINTER VALUE IN RCD STRUCTURED FIELD NUMBER *number* WILL CAUSE AN INFINITE LOOP.

Explanation: The FIELD RECORD DESCRIPTOR POINTER parameter in the Record Descriptor (RCD) structured field identified in this message caused an infinite-loop condition. The RCD 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2008I DATA IN A PAGEDEF RESOURCE IS NOT VALID: RCD OR XMD STRUCTURED FIELD NUMBER *number* SPECIFIES A VALUE THAT IS NOT VALID AS A POINTER TO A FIELD RCD OR XMD. THE VALUE *rcdvalue* EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF *Incvalue*.

Explanation: The Record Descriptor (RCD) or XML Descriptor (XMD) structured field identified in this message specifies a value as a pointer to a Field RCD or XMD. The value specified is not valid. The value is greater than the COUNT value in the Line Descriptor Count (LNC) structured field in the current data map. The LNC, RCD, and XMD structured fields are 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2010S RECORD FORMATTING WAS REQUESTED BY THE PAGE DEFINITION BUT THAT FUNCTION IS NOT SUPPORTED BY THIS RELEASE OF ACIF.

Explanation: The record formatting function is not supported by this release of ACIF.

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: To use the record formatting function, submit this job to a version of ACIF that supports record formatting.

System programmer response: Select a page definition that does not use the record formatting function.

APK2011I DATA IN A PAGEDEF RESOURCE IS NOT VALID: DATA MAP *datamap1* AND DATA MAP *datamap2* ARE FOR PROCESSING DIFFERENT TYPES OF DATA. ALL DATA MAPS IN THE PAGE DEFINITION MUST SPECIFY THE SAME DATA FORMATTING.

Explanation: A page definition can only be used for one type of data. A single page definition cannot be used to mix the processing of traditional line data, record-format line data, and XML 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 page definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2012S DATA IN A PAGEDEF RESOURCE IS NOT VALID: A NON-ZERO RECORD IDENTIFIER PARAMETER VALUE *value* WAS SPECIFIED IN RCD STRUCTURED FIELD NUMBER *number*.

Explanation: For Record Descriptor (RCD) structured fields that are marked as either a field or a conditional

processing RCD, the RECORD IDENTIFIER parameter value must be all zeros. The RCD structured fields are 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2013S DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE SAME RECORD IDENTIFIER *identifier* WAS SPECIFIED IN RCD STRUCTURED FIELD NUMBERS *number1* AND *number2*. ALL RECORD IDENTIFIERS MUST BE UNIQUE IN THE SAME DATA MAP.

Explanation: With the exception of the default Page Header Record Descriptor (RCD) structured field, the default Page Trailer RCD structured field, Field RCD structured fields, and Conditional Processing RCD structured fields, all other RCD structured fields in a data map must have a unique record identifier parameter value specified.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2014I DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS NOT VALID: THE PAGE SIZE IS NOT LARGE ENOUGH TO PLACE THE FIRST RECORD OF THE PAGE BY USING RCD OR XMD STRUCTURED FIELD NUMBER *number* AND ITS ASSOCIATED FIELD RCD OR XMD STRUCTURED FIELDS.

Explanation: The Body Record Descriptor (RCD) or XML Descriptor (XMD) structured field selected for placing the first body record of the page does not fit within the area of the page defined by the bottom margin. If Field RCD or XMD structured fields are being used, one of the Field RCD or XMD structured fields might be positioning data beyond the bottom margin. This error prevents PSF from being able to place the record and continuing.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2015S DATA IN A PAGEDEF RESOURCE IS NOT VALID: AN RCD STRUCTURED FIELD SPECIFIED A GRAPHICS DESCRIPTOR TRIPLET TO END ALL STARTED GRAPHICS DESCRIPTOR TRIPLETS THAT HAVE A MATCHING GRAPHIC PARAMETER VALUE *value*, BUT A MATCH COULD NOT BE FOUND.

Explanation: A graphics object can be started by one Record Descriptor (RCD) structured field and ended with another RCD structured field. When this is done, the Graphics Descriptor triplets that start and end a graphics object must have matching GRAPHID parameter values specified and the RCD structured fields must have matching orientations. ACIF could not find a match between the start and end Graphics Descriptor triplets using the GRAPHID parameter from the end Graphics Descriptor triplet and the TEXT

ORIENTATION parameter value from the RCD 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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2016S DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE USE RECORD IDENTIFIER FLAG WAS SET BUT THE SUM OF THE DATA START POSITION AND THE DATA LENGTH PARAMETER VALUES IN RCD STRUCTURED FIELD NUMBER *number* SELECTS DATA BEYOND THE RECORD IDENTIFIER FIELD.

Explanation: For Record Descriptor (RCD) structured fields that are marked to use only the record identifier portion of an input record, only the record identifier can be accessed by the RCD. The DATA START parameter plus the DATA LENGTH parameter of this RCD accesses data beyond the 10-byte record identifier area of the input record.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2017S DATA IN A PAGEDEF RESOURCE IS NOT VALID: A FONT IS NEEDED FOR THE *structuredfield* STRUCTURED FIELD IN DATA MAP *datamap* BUT NO FONTS WERE MAPPED IN THE DATA MAP.

Explanation: Fonts needed for printing record-format line data or XML data must be selected in the data map. The CHARS JCL parameter cannot be used to select fonts. The data map identified in this message contained a Record Descriptor (RCD) or an XML Descriptor (XMD) structured field that requires a font, but no fonts were specified in the data map.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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.

APK2018S DATA IN A PAGEDEF RESOURCE IS NOT VALID: RCD STRUCTURED FIELD *structuredfield* REQUESTED THAT THE PAGE NUMBER BE RESET, BUT THE PAGE NUMBER PARAMETER CONTAINS ZERO.

Explanation: The PAGE NUMBER parameter in a Record Descriptor (RCD) structured field cannot be zero when the RCD requests that ACIF reset the page number.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2019I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE FONT SELECTED FOR PRINTING THE PAGE NUMBER ON RCD OR XMD STRUCTURED FIELD NUMBER *number* CANNOT BE A DOUBLE-BYTE FONT WHEN USING THE ASCII ENCODING SCHEME.

Explanation: ACIF cannot determine the correct code points to generate when a double-byte font is used to print the page number using the ASCII encoding scheme. The structured field identified in this message selected a double-byte ASCII font for printing the page number. This is not allowed.

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 *Mixed Object Document Content Architecture Reference* and *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 might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2020I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE OBJECT OID LENGTH SPECIFIED IN A FULLY QUALIFIED NAME TRIPLET ON AN *structuredfield* STRUCTURED FIELD IS GREATER THAN 129 BYTES.

Explanation: An object OID being specified in a Fully Qualified Name triplet has a length that exceeds 129 bytes.

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error

might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2021I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE *structuredfield* STRUCTURED FIELD CONTAINS UNPAIRED FQN X'BE' AND FQN X'DE' TRIPLETS.

Explanation: If this is an Include Object (IOB) or Preprocess Presentation Object (PPO) structured field, the Fully Qualified Name (FQN) triplet with an FQNType of Data Object Internal Resource Reference (X'BE') must immediately follow an FQN triplet with an FQNType of Data Object External Resource Reference (X'DE'). If this is a Map Data Resource (MDR) structured field, a repeating group with an FQN triplet type X'BE' must also include an FQN triplet type X'DE'.

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 might 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.

APK2022I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A RESOURCE NAMED AS A SECONDARY RESOURCE ON AN IOB STRUCTURED FIELD IS NOT NAMED IN THE ACTIVE ENVIRONMENT GROUP OF THE PAGE OR OVERLAY.

Explanation: An Include Object (IOB) structured field calls for a secondary resource. This resource must be named in an MDR in the active environment group of the page or overlay containing the IOB 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 print data set or the resource, 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 is correct, the error might 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 the error involves separator pages or the message data set, use the information provided in the User Response section to correct the error. 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2023I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE FORMAT SPECIFIED IN A FULLY QUALIFIED NAME TRIPLET ON AN *structuredfield* STRUCTURED FIELD IS NOT VALID.

Explanation: The FQNFmt specified in a Fully Qualified Name (FQN) triplet on a structured field is not valid.

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2024I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A FULLY QUALIFIED NAME TRIPLET MAPPED AN OBJECT THAT IS NOT ALLOWED IN AN MDR STRUCTURED FIELD IN AN OBJECT ENVIRONMENT GROUP.

Explanation: A Fully Qualified Name (FQN) triplet with an FQNType of Begin Resource Object Reference (X'84') or Other Object Data Reference (X'CE') is not allowed on a Map Data Resource (MDR) structured field in an Object Environment Group. In addition, an FQN triplet with an FQNType of Data Object External

Resource Reference (X'DE') can only map a data object font in a graphics object or a bar code object.

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 more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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.

APK2025I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: AN ERROR WAS FOUND IN A FULLY QUALIFIED NAME TRIPLET SPECIFIED IN A REPEATING GROUP ON AN structuredfield STRUCTURED FIELD.

Explanation: One of these errors was found:

- More than one Fully Qualified Name (FQN) triplet with one of these FQNTypes was found in a repeating group on a Map Data Resource (MDR) structured field:
 - Begin Resource Object Reference (X'84')
 - Other Object Data Reference (X'CE')
 - Data Object External Resource Reference (X'DE')
 - Code Page Name Reference (X'85')
- More than one FQN triplet with one of these FQNTypes was found in a repeating group on a Preprocess Presentation Object (PPO) structured field:
 - Begin Resource Object Reference (X'84')
 - Other Object Data Reference (X'CE')
- An FQNType that is not valid was specified on a FQN triplet on an MDR.

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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.

APK2026I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: THE OBJECT CLASS SPECIFIED IN AN OBJECT CLASSIFICATION TRIPLET ON AN MDR STRUCTURED FIELD IS NOT VALID.

Explanation: The ObjClass specified in an Object Classification triplet on a Map Data Resource (MDR) structured field must be X'40' or X'41' if the Fully Qualified Name (FQN) triplet type in the repeating group is a Data Object External Resource Reference (X'DE'). The ObjClass specified must be X'01' if the FQN triplet type in the repeating group is an Other Object Data Reference (X'CE').

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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.

APK2027I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: AN MDR STRUCTURED FIELD SPECIFIES THE SAME RESOURCE REFERENCE MORE THAN ONCE IN AN ENVIRONMENT GROUP.

Explanation: The same resource reference cannot be made in a Map Data Resource (MDR) structured field in an environment group.

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2029I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: OBJECT OID *objectoid1* DOES NOT MATCH THE OBJECT OID *objectoid2* SPECIFIED ON THE *structuredfield* STRUCTURED FIELD.

Explanation: The object OID specified on a structured field must match the object OID specified on the Map Data Resource (MDR) or Include Object (IOB) structured field that referenced it. A value of *** indicates an OID was not specified.

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2030I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A SECONDARY RESOURCE IS SPECIFIED ON AN IOB STRUCTURED FIELD THAT INCLUDES A BAR CODE OR GRAPHICS OBJECT.

Explanation: A Fully Qualified Name (FQN) triplet of type Data Object External Resource Reference (X'DE') is specified on an Include Object (IOB) structured field that is including a bar code or graphics object. Secondary resources are not allowed for these objects.

System action: The secondary resource reference is ignored and processing continues.

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 more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 your system's diagnosis reference for assistance in determining the source of the problem.

APK2033I THE BAR CODE DATA OR BAR CODE DATA PLUS THE ADDITIONAL 2D BAR CODE PARAMETERS EXCEED THE OUTPUT COMMAND BUFFER.

Explanation: Either the bar code data itself or the bar code data plus the macro control block data specified for a 2D bar code exceeds the size of the output command buffer. The macro control block data is specified in your page definition as part of the BCXPARMS (additional bar code parameters).

System action: ACIF issues this message and continues processing.

User response: Change the amount of data specified for your bar code or reduce the amount of data in the macro control block.

System programmer response: None.

APK2039I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: A DUPLICATE FINISHING OPERATION WAS FOUND IN THE *mapname* MEDIUM MAP.

Explanation: The same finishing operation was specified more than once in a medium map. This nesting of the same finishing operation is not allowed. The Media Finishing Control (MFC) structured field is in a form definition or an internal medium map in the print data set.

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* and *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 might 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: None.

APK2040I THE NUMBER OF MEDIA COLLECTION FINISHING NESTING LEVELS IS MORE THAN 4.

Explanation: A maximum of four levels of nesting is allowed for media collection finishing. The Medium Finishing Control (MFC) structured field can be

contained in a form definition or internal medium map in a page.

System action: ACIF stops processing the data set.

User response: If you created the form definition or internal medium map, you must remove one or more levels of media collection finishing operations. Refer to *Mixed Object Document Content Architecture Reference* for more information about the structured field. Resubmit the print request. If the total number of nesting levels is less than or equal to four, the error might be an ACIF logic error.

System programmer response: None.

APK2041I DATA IN AN INPUT RECORD OR PAGEDEF IS NOT VALID: INPUT DATA BEING USED FOR A VARIABLE RESOURCE NAME IN LND OR RCD STRUCTURED *number* IS DOUBLE BYTE DATA.

Explanation: A Resource Object Include triplet or an Extended Resource Local ID triplet on a Line Descriptor (LND) or Record Descriptor (RCD) structured field requests that the input data for the resource name is included. This input data cannot be double-byte 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 page definition, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* and *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 might 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 is valid.

APK2042I DATA IN A PAGEDEF RESOURCE IS NOT VALID: AN XML PAGE DEFINITION REQUESTED THAT THE INPUT DATA BE USED FOR A RESOURCE NAME ON XMD STRUCTURED FIELD NUMBER *number*.

Explanation: An Object Reference Qualifier (ORQ) triplet has been specified on an XML Descriptor (XMD) structured field. This function is not supported when using an XML 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 resource, correct the error and resubmit the print request. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error might be an ACIF logic error. If you used a program to create the structured fields for the resource, contact your system programmer.

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 is valid.

APK2044I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE FIELD XMD POINTER VALUE IN XMD STRUCTURED FIELD NUMBER *number* WILL CAUSE AN INFINITE LOOP.

Explanation: The Field XML Descriptor Pointer value in the XML Descriptor (XMD) structured field identified in this message caused an infinite loop condition. The XMD 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* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field does not have an error, the error might 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 is valid.

APK2045I THE ENCODING SCHEME SPECIFIED IN A PAGE DEFINITION USED TO PROCESS XML DATA IS NOT SUPPORTED BY ACIF.

Explanation: The encoding scheme specified is not supported by ACIF.

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: You need to use an encoding scheme that is supported by ACIF for XML data processing. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* and *Mixed Object Document Content Architecture Reference* for more

information about the structured field. If the structured field does not have an error, the error might 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 is valid.

APK2046I XML DATA FORMATTING WAS REQUESTED BY THE PAGE DEFINITION BUT THAT FUNCTION IS NOT SUPPORTED BY THIS RELEASE OF ACIF.

Explanation: The XML data formatting function is not supported by this release of ACIF.

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: To use the XML data formatting function, submit this job to a version of ACIF that supports XML data formatting.

System programmer response: None.

APK2047I DATA IN A PAGEDEF RESOURCE IS NOT VALID: DATA MAP *datamapname1* AND DATA MAP *datamapname2* HAVE DIFFERENT ENCODING SCHEMES SPECIFIED FOR THE USER DATA. ALL DATA MAPS IN THE PAGE DEFINITION MUST SPECIFY THE SAME ENCODING SCHEME.

Explanation: All the data maps in a page definition used to process XML data must use the same encoding scheme for the user data.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

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* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field does not have an error, the error might 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 is valid.

APK2048I DATA IN AN INPUT RECORD IS NOT VALID: A DTD DECLARATION AT CHARACTER COUNT NUMBER *number* IS SPECIFIED OUTSIDE OF A DTD.

Explanation: A document type definition (DTD) declaration is only allowed inside a DTD. The character count number specified in this message is relative to the start of the record.

System action: ACIF stops processing the current data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2049I DATA IN AN INPUT RECORD IS NOT VALID: THE XML COMMENT SYNTAX AT CHARACTER COUNT NUMBER *number* IS NOT VALID.

Explanation: After an XML comment has been started, you can only use two dashes in a row when ending a comment. The character count number specified in the message is relative to the start of the record.

System action: ACIF stops processing the current data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2050I DATA IN AN INPUT RECORD IS NOT VALID: THE XML END TAG AT CHARACTER COUNT NUMBER *number* DOES NOT MATCH THE LAST START TAG.

Explanation: An XML end tag must exactly match its start tag. The character count number specified in this message is relative to the start of the record.

System action: ACIF stops processing the current

data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2051I DATA IN AN INPUT RECORD IS NOT VALID: THE END OF A DOCUMENT TYPE DECLARATION AT CHARACTER COUNT NUMBER *number* IS NOT THE CORRECT SYNTAX.

Explanation: The end of a document type declaration (DTD) did not have the correct syntax. The character count number specified in this message is relative to the start of the record.

System action: ACIF stops processing the current data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2052I DATA IN AN INPUT RECORD IS NOT VALID: THE CHARACTER CODE AT CHARACTER COUNT NUMBER *number* IS NOT A VALID VALUE FOR A CHARACTER REFERENCE.

Explanation: A character code inside a character reference is not one of the allowed values. The character count number specified in this message is relative to the start of the record.

System action: ACIF stops processing the current data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2053I DATA IN AN INPUT RECORD IS NOT VALID: THE ENTITY AT CHARACTER COUNT NUMBER *number* IS NOT DEFINED IN THE DOCUMENT TYPE DEFINITION.

Explanation: ACIF only allows internal general entity references, which must be defined in an internal document type definition (DTD). The character count number specified in this message is relative to the start of the record.

System action: ACIF stops processing the current data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2054I DATA IN AN INPUT RECORD IS NOT VALID: THE CHARACTER IN A TAG NAME AT CHARACTER COUNT NUMBER *number* IS NOT VALID.

Explanation: A character in an XML tag name is not valid. The character count number specified in this message is relative to the start of the record.

System action: ACIF stops processing the current data set and issues a message identifying the position of the error in the data stream.

User response: If you created the XML data, correct the error and resubmit the print request. Refer to the XML specification, Extensible Markup Language (XML) 1.0, on the World Wide Web Consortium Web site. If the XML data does not have an error, the error might be an ACIF logic error. If you used a program to create the XML data, contact your system programmer.

System programmer response: If an IBM licensed program was used to create the XML data with the error, verify that the input to that program is valid.

APK2055I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE ENCODING SCHEME IDENTIFIER FOR THE USER DATA IS NOT SPECIFIED IN THE ENCODING SCHEME TRIPLET ON THE BDM STRUCTURED FIELD.

Explanation: The Encoding Scheme Identifier for User Data (ESidUD) is missing on the Encoding Scheme triplet (X'50') on a Begin Data Map (BDM) structured field. This information is required when processing an XML page definition.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: You need to provide the encoding scheme for the user data. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field does not have an error, the error might 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 is valid.

APK2056I DATA IN A PAGEDEF RESOURCE IS NOT VALID: THE SAME QUALIFIED TAG WAS SPECIFIED IN XMD STRUCTURED FIELD NUMBERS number1 AND number2. ALL QUALIFIED TAGS MUST BE UNIQUE IN THE SAME DATA MAP.

Explanation: All XML Descriptor (XMD) structured fields in a data map must have a unique qualified tag specified; with the exception of these types of XMD structured fields:

- Default Page Header
- Default Page Trailer
- Field
- Conditional Processing
- Attribute

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

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* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field does not have an error, the error might 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 is valid.

APK2057I DATA IN A PAGEDEF RESOURCE IS NOT VALID: RELATIVE INLINE POSITIONING ON AN XMD STRUCTURED FIELD CAN ONLY BE USED TO PLACE TEXT DATA.

Explanation: A Resource Object Include, Extended Resource Local ID, Bar Code Symbol Descriptor, or Graphics Descriptor triplet is specified on an XML Descriptor (XMD) structured field that uses relative inline positioning when including a page segment, overlay, or object with an XMD structured field. You must also use absolute inline positioning when generating a bar code or graphics object with an XMD structured field.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: You need to change your inline positioning to an absolute value. Refer to *Advanced Function Presentation: Programming Guide and Line Data Reference* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field does not have an error, the error might 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 is valid.

APK2072I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: AN INCORRECT COMBINATION OF TRIPLETS WAS SPECIFIED WHEN MAPPING A DATA OBJECT FONT IN AN MDR STRUCTURED FIELD.

Explanation: When mapping a data object font (DOF) in a Map Data Resource (MDR) structured field, you must have Fully Qualified Name (FQN) type X'DE', FQN type X'BE', and DOF Descriptor X'8B' triplets specified as well. In addition, the FQN type X'BE' triplet must specify a one-byte local ID.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

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* and *Mixed Object Document Content Architecture Reference* for more information about the structured field. If the structured field has no error, the error might 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 is valid.

APK2073I DATA IN AN INPUT RECORD IS NOT VALID: THE FULLY QUALIFIED TRIPLET TYPE AND THE OBJECT TYPE SPECIFIED IN A REPEATING GROUP ON A PREPROCESS PRESENTATION OBJECT STRUCTURED FIELD DO NOT AGREE.

Explanation: When a repeating group in a Preprocess Presentation Object (PPO) structured field is mapping an object container, you must use a Fully Qualified Name (FQN) triplet of type X'CE'. When the repeating group is mapping an IOCA object or overlay, you must use an FQN triplet of type X'84'.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: If you created the structured fields for the print data set or resource, correct the error and resubmit the print request. It is possible that the problem is the object type OID specified in the X'10' Object Classification triplet. It might specify that a TrueType or OpenType collection is being mapped when a font has really been mapped. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 resource with the error, verify that the input to that program is valid.

APK2074I DATA IN AN INPUT RECORD OR RESOURCE IS NOT VALID: AN MDR STRUCTURED FIELD IS MAPPING THE NAME OF A TRUETYPE OR OPENTYPE COLLECTION.

Explanation: When mapping a data object font (DOF), you can only specify the name of a TrueType or OpenType font. This font might actually reside in a collection, but the Map Data Resource (MDR) structured field needs the font name in the collection.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: If you created the structured fields for the print data set or resource, correct the error and resubmit the print request. It is possible that the

problem is the object type OID specified in the X'10' Object Classification triplet. It might specify that a TrueType or OpenType collection is being mapped when a font has really been mapped. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 page definition with the error, verify that the input to that program is valid.

APK2075S TRCS ARE NOT ALLOWED WITH A PAGE THAT HAS FONTS MAPPED IN BOTH AN MCF AND AN MDR. THE DATA MAP BEING PROCESSED IS *datamap*.

Explanation: You can use table reference characters (TRCs) with fonts mapped in a Map Coded Font (MCF) structured field (FOCA fonts) or a Map Data Resource (MDR) structured field (TrueType and OpenType fonts). However, you cannot have a mixture of both types of fonts in a data map and use TRCs.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: Change your page definition to use either all FOCA fonts (mapped in an MCF) or all TrueType or OpenType fonts (mapped in an MDR).

System programmer response: If the error involves separator pages or the message data set, use the information provided in the User Response section to correct the error.

APK2076I PAGE BASED SOSI PROCESSING HAS BEEN REQUESTED BUT A SINGLE BYTE FONT WITH A FONT ID OF 1 AND A DOUBLE BYTE FONT WITH A FONT ID OF 2 HAVE NOT BEEN MAPPED. THE DATA MAP BEING PROCESSED IS *datamap*.

Explanation: When doing page based SOSI processing, you are switching back and forth between the same two fonts. As a result, there must be two fonts mapped using font IDs 1 and 2.

System action: ACIF stops processing the current data set and issues additional messages that identify the processing environment when the error was found.

User response: Change your page definition to map a single byte font with font ID 1 and a double byte font with font ID 2. Refer to the documentation for the application that you use to generate page definitions for

information about how to map fonts to specific font IDs.

System programmer response: If the error involves separator pages or the message data set, use the information provided in the User Response section to correct the error.

APK2077I DATA IN AN INPUT RECORD IS NOT VALID: AN INLINE TRUETYPE OR OPENTYPE COLLECTION DOES NOT HAVE ANY BASE FONTS SPECIFIED.

Explanation: The Begin Resource (BRS) structured field must have Fully Qualified Name triplet of type X'6E' for each base font contained in the collection.

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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 is valid.

APK2078I DATA IN AN INPUT RECORD IS NOT VALID: AN INLINE RESOURCE HAS AN INCORRECT SPECIFICATION OF BASE FONTS OR LINKED FONTS OR BOTH.

Explanation: A Begin Resource (BRS) structured field has an error. Base fonts (Fully Qualified Name triplets of type X'6E') can only be specified for TrueType or OpenType collections. Linked fonts (Fully Qualified Name triplets of type X'7E') can only be specified for a TrueType or OpenType font or for a base font of a TrueType or OpenType collection.

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 print data set, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 is valid.

APK2079I THE INFORMATION NEEDED TO DESCRIBE A TRUETYPE OR OPENTYPE FONT OR COLLECTION *name* EXCEEDS THE BEGIN OBJECT CONTAINER STRUCTURED FIELD.

Explanation: The number of names and linked fonts for a TrueType or OpenType font or the number of base fonts and linked fonts for a TrueType or OpenType collection exceeds the Begin Object Container structured field.

System action: ACIF stops processing the object container.

User response: You cannot collect this TrueType or OpenType font or font collection.

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 is valid.

APK2080I RESOURCE NAME *resourcename* CANNOT BE CONVERTED TO UTF16 ENCODING. RETURN CODE *returncode* AND REASON CODE *reasoncode* VALUES WERE RETURNED BY THE CONVERTER.

| **Explanation:** ACIF must convert the input data to the
| specified encoding to continue processing. An error
| occurred during this conversion. If the input data is a
| resource name, the CCSID indicates that mixed
| single-byte or double-byte EBCDIC data is being
| converted to mixed single-byte or double-byte ASCII
| data for a QR Code Bar code. If the error occurs on
| data for a bar code, only the first 50 bytes of the data
| are shown in the message. The return codes and
| reason codes are returned by the system's conversion
| services (ICONV on z/OS and UCONV on AIX and
| Windows). Refer to the system documentation for these
| conversion services for more information about the
| errors.

| **System action:** ACIF stops processing the input file
| and issues a message identifying the position of the
| structured field in the file or resource.

| **User response:** Correct the error as described by the
| system documentation for the conversion service.

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 is valid.

APK2082I DATA IN AN INPUT RECORD IS NOT VALID. RESOURCE *resourcename* OR OBJECT OID *objectoid* IS SPECIFIED ON A PREPROCESS PRESENTATION OBJECT (PPO) STRUCTURED FIELD BUT IS NOT MAPPED IN THE RESOURCE ENVIRONMENT GROUP.

Explanation: All resources specified on a PPO structured field must be mapped in the resource environment group. Overlays must be mapped with a Map Page Overlay structured field. IOCA image and object containers must be mapped with a Map Data Resource 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 print data set or resource, correct the error and resubmit the print request. Refer to *Mixed Object Document Content Architecture Reference* for more information about the correct format of the referenced structured field. If the structured field is correct, the error might 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 is valid.

APK3506I DATA OBJECT RESOURCE TYPE *type*. OBJECT ID *objectid*, COULD NOT BE FOUND IN THE RESOURCE LIBRARY.

Explanation: The registration ID (object-type OID) for the specified data object resource cannot be read from the resource library. Only objects with valid data object resource names or resource locator names are supported by ACIF. The registration ID is specified in the Object Classification triplet on an IOB, BOC, BRS, or MDR structured field. If the *objectid* specified in this message is ***, ACIF either does not support the registration ID or does not have enough information to identify the *objectid*.

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:** Modify the structured field that refers
| to the object to include a resource locator triplet.

System programmer response: If an IBM licensed program was used to create the structured fields, verify that the input to that program was valid and the correct printer is being used.

| **APK3507I** RESIDENT COLOR PROFILE *type*,
| OBJECT ID *objectid*, COULD NOT BE
| FOUND IN THE RESOURCE LIBRARY.

| **Explanation:** The resident color profile with object ID
| (object-type OID) could not be found because no
| resource locator name was specified. ACIF cannot
| access objects by OID name only.

| **System action:** ACIF ignores the request for the
| resident color profile and, because the profile is not
| necessary, continues processing.

| **User response:** Modify the structured field that refers
| to the object to include a resource locator triplet.

Appendix A. Helpful Hints

When using ACIF, these topics might prove helpful to you:

- Working with control statements that contain numbered lines (z/OS, VM, and VSE environments only)
- Placing TLEs in named groups to avoid storage problems
- Understanding how ANSI and machine carriage controls are used
- Transferring files into AIX and Windows
- Understanding common methods of transferring files into AIX or Windows from other systems:
 - Physical media such as tape
 - PC file transfer program
 - File Transfer Protocol (FTP)
 - Download for z/OS
 - Other considerations
- Creating Invoke Medium Map (IMM) structured fields
- Indexing considerations
- Concatenating the resource file and the document file
- Processing inline resources
- Specifying the IMAGEOUT parameter
- Creating a MO:DCA-P object container
- Understanding error code 310

Working with Control Statements That Contain Numbered Lines

This section applies only to the z/OS, VM, VSE environments. If you work in the AIX or Windows environment, you can skip this information about control statements.

Because ACIF reads all 80 columns of the control statements for processing purposes, you sometimes can receive unexpected results when data set names are continued and the control statements have line numbers in columns 73-80. (ACIF attempts to use the line number as a data set name, and issues message APK451S and message APK417I 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.

Placing TLEs in Named Groups to Avoid Storage Problems

You should be aware that if you request **INDEXOBJ=ALL** for a job that has an input data set containing composed (MO:DCA-P) pages, page-level TLEs (TLE records after the AEG), and no named groups (BNG/ENG), your job might end with message APK410S or message APK408S.

To avoid having ACIF end your job, IBM recommends that you place page-level Tag Logical Elements (TLEs) inside named groups, using one named group per page. This is because, when 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-P 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 continues to build the index in memory. If the input file is large enough, there will not be enough memory, and ACIF will stop. The ACIF

memory manager currently limits the number (but not the size) of memory blocks that can be allocated; therefore, increasing REGION size might not alleviate the problem.

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 characters perform the required spacing before the line is printed. ANSI controls can be encoded in EBCDIC (**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 can 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 are treated as printing characters. If no carriage controls are specified, the file is printed as though it were single spaced.

Transferring Files into AIX and Windows

Information in this section only applies to transferring files into the AIX and Windows environments; therefore, if you work in the z/OS, VM, or VSE environment, you can skip this section.

ACIF needs to know two things about a file in order to print it:

- The length of each print record
- The kind of carriage control used

As simple as this sounds, it is the source of most of the difficulty people have printing with ACIF in an AIX or Windows environment.

ACIF processes print records. A record is a sequence of contiguous characters, usually representing a printed line or a MO:DCA-P structured field.² 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 can use a length prefix to indicate the length of the record in the file. The length prefix is a 2-byte binary number that prefixes each record. The record length is all the bytes in the record other than the 2-byte length prefix, but including X'5A' carriage control characters at the start of structured fields. Use the **FILEFORMAT=RECORD** parameter to identify files with length prefixes.
- Variable-length files can use a separator or delimiter (also called a new-line character) 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 part of the record. For AIX or Windows, the default delimiter is X'0A'. If the file uses EBCDIC encoding, the default delimiter is X'25'. Use the **FILEFORMAT=STREAM** parameter to designate files that use delimiters to indicate record boundaries. NEWLINE can be used with **FILEFORMAT=STREAM** to specify the delimiter if the default is not correct. See page 43 for the NEWLINE values you can specify.
- If NEWLINE is not specified, ACIF reads the first six bytes to decide whether a file is encoded in ASCII or EBCDIC characters. If only characters below X'7F' are found, ACIF assumes the file is ASCII and looks for the ASCII new-line character (X'0A') to delimit the end of a record. Otherwise, ACIF looks for the EBCDIC new-line character (X'25') to delimit the end of a record. Because ACIF might decide incorrectly whether the file is ASCII or EBCDIC (for example, if the data contains non-English characters), the best way to ensure that your data is processed correctly is to explicitly specify NEWLINE with the **FILEFORMAT** parameter. See page 43 for the NEWLINE values you can specify.

- **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

2. Structured fields are similar to print commands.

record length. You must know the record length and use the **FILEFORMAT=RECORD,*n*** parameter, where *n* represents the length of each record.

For variable- and fixed-length files using length prefixes, MO:DCA-P 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 are processed correctly if there is a length prefix.

Note: CRLF characters found in mixed mode at the end of MO:DCA-P structured fields are treated as a separate record. This results in extra blank lines being printed.

Understanding Common Methods of Transferring Files into AIX or Windows from Other Systems

You can transfer files from other systems into AIX or Windows 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 or Windows and produce output that ACIF can use are:

- Physical media (such as tape)
- PC file transfer program
- File Transfer Program (FTP)
- Download for z/OS

Other considerations for transferring files are also listed in this section.

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 can transfer files from other systems to AIX or Windows using a PC file transfer program such as IND\$FILE. You can 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 z/OS 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 **INPEXIT=asciinpe** 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 delimiters and carriage returns.

FTP

From most systems, FTP works similarly to PC file transfer; most of the same options are provided. Also, when executing FTP on an AIX or Windows 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.

Download for z/OS

The Download for z/OS feature of PSF for z/OS automatically transmits data from the JES spool to another systems in the IBM TCP/IP network. A cooperative print server or archive server receives the data sets for printing with PSF for z/OS, Infoprint Manager for AIX or Infoprint Manager for Windows, or for archiving with Content Manager OnDemand. Download for z/OS can supply the two-byte record length prefix for each file downloaded from the JES spool.

Other Considerations

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 might also contain machine control characters, mixed line data and structured fields, or special code points that have no standard mapping.³ 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 might 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-P) 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.

Creating Invoke Medium Map (IMM) Structured Fields

To ensure that pages are reprinted (or viewed) using the correct medium map, retrieval programs must be able to detect which medium map is active. 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.

3. When ASCII is specified, for example, the file transfer program might destroy the data in translation. When binary is specified, the file transfer program might not be able to indicate record lengths.

If an ACIF input file consists of more than one document (determined by the BDT and EDT structured fields) and **INDEXOBJ=BDTLY** is not specified, ACIF removes all BDT and EDT structured fields when it processes the file. This can cause a document to begin with an incorrect medium map. To prevent an incorrect medium map from being used, specify **INSERTIMM=YES**. ACIF inserts the appropriate IMM before the first page that was indicated by the BDT structured field that ACIF removed.

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 must 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 six branches, using the account number, statement date, and branch number.
- The resultant output files were archived using a system that lets these statements 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 might 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 might produce two indexing tags for the same physical page of output. In this situation, only the first index attribute name appears as a group name, when you are using AFP Workbench Viewer. 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 points to the page on which the carriage control character was found, not to the new page created by the carriage control character. This is because machine controls write before executing any action and are, therefore, associated with the page or line on which they appear. Using machine carriage control characters for triggers is not recommended.
- 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 makes 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 is placed on the first page.

Concatenating the Resource File and the Document File

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. Remember two things when doing this:

- First, although AFP 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 AFP Workbench Viewer, because it removes any inline objects before using the offset information.

Processing Inline Resources

To process inline resources, the inline resources must be included in the input file in the order in which they are used, or you must specify **EXTENSIONS=RESORDER** (see “EXTENSIONS” on page 39). To write resources that are inline in the data file to the output file (OUTPUTDD), as well as to the resource file (RESOBJDD), specify **RESTYPE=INLINE**.

Note: The input file cannot have inline resources in XML data.

If **EXTENSIONS=RESORDER** is not specified, and a resource references another resource, the referenced resource must be included inline before the resource that references it. For example, if an overlay references a coded font that consists of the character set C0D0GT18 and code page T1D0BASE, the inline resources must be in this order:

```
code page T1D0BASE
character set C0D0GT18
coded font
overlay
```

ACIF does not look ahead in the inline resources, so, if the inline resources are not in the correct order and **EXTENSIONS=RESORDER** is not specified, ACIF tries to read the referenced resource from a resource library. If the resource is not found, ACIF ends processing with an error.

Keep in mind that when you are indexing and writing inline resources to the output file, the offsets in the index object file are the same as if you are doing regular resource collection to a resource file. This is because the offsets are calculated from the Begin Document (BDT) structured field, not from the beginning of the

output document file. The offset from the BDT structured field to the indexed data is the same regardless of whether resources precede it.

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 can use a significantly higher number of bytes than an IM1 image and can 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 might require it, all products cannot accept IOCA data. All software products from the IBM Printing Systems Division do, however, accept IOCA data as well as IM1 image data.

IBM recommends that you specify **IMAGEOUT=ASIS**, which produces all image data in the same format as in the input file, unless you have a specific requirement for IOCA images.

Creating MO:DCA-P Object Containers

Object containers are MO:DCA-P resources that contain non-OCA objects, such as TIFF images, encapsulated PostScript, JFIF, and microfilm setup. Object containers can be included in a data stream using the Include Object (IOB) structured field.

Not all presentation systems can present non-OCA objects, but ACIF includes them as part of the resource object if the **RESTYPE** parameter is set to **ALL** or includes **OBJCON**. When ACIF processes an IOB, it checks that the object type value is X'92' for **OTHER** and the named object read from the resource library is not a MO:DCA-P object already. ACIF then creates a MO:DCA-P object container object by wrapping the raw object data in Object Container Data (OCD) structured fields and creating an Object Environment Group using the values given by the IOB. The result is a MO:DCA-P object container saved in the resource object file. For information about the structure of object containers, see *Mixed Object Document Content Architecture Reference*.

Understanding Error Return Code 310

When running ACIF on AIX or Windows, the most common I/O errors occur in the input file and produce error messages APK411S and APK413S with return code 310. This return code usually indicates one of these:

- The value you specified for the **FILEFORMAT** parameter does not match the actual format of the data file. Check that **NEWLINE** and its encoding values are correct if you have specified **FILEFORMAT=STREAM**, or that the record length is correct for **FILEFORMAT=RECORD,n**.
- ACIF has read an input line (record) larger than 32 KB. In this case, the default **FILEFORMAT=STREAM** was specified, but no record separator was found within the first 32 KB of the current input line. This is usually due to the value of the **NEWLINE** characters not matching the record separator (line endings) used in the input file. ACIF uses a default value of X'0A' for **NEWLINE**, which is the UNIX new-line character. Files from PC workstations typically use carriage returns or line feeds (X'0D0A') as delimiters, while stream files from z/OS systems can use EBCDIC new-line characters (X'15') or EBCDIC line-feed characters (X'25'). ACIF cannot always reliably detect the record separator; therefore, you must ensure that the value specified for **NEWLINE** is correct for the input file ACIF is processing.

- ACIF has not read enough bytes at the end of file. Some input files contain no record separator characters. These files must be processed as **FILEFORMAT=RECORD,*n***, where *n* is the length of each record, or as **FILEFORMAT=RECORD**, where each input record must be preceded by its two-byte length. If **FILEFORMAT=RECORD,*n*** is specified and the number of bytes in the file is not an exact multiple of *n*, ACIF returns an error code of 310 when the last byte in the file is read.

Appendix B. Data Stream Information

General-use Programming Interface and Associated Guidance Information is contained in this appendix.

This appendix describes these structured fields: Tag Logical Element (TLE), Begin Resource Group (BRG), Begin Resource (BRS or BR), End Resource Group (ERG), End Resource (ERS or ER), and No Operation (NOP). It also describes the formats of the resource data sets.

Note: All MO:DCA-P data and resource files processed by ACIF must contain a X'5A' carriage control character at the start of each structured field.

Tag Logical Element (TLE) Structured Field

TLE structured fields are allowed only in AFP data stream (MO:DCA-P) documents. AFP Toolbox 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:

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, two reserved bytes (X'0000'), and the tag value.

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 might want to associate specific indexing information at the statement level (for example, account number, date, customer name, and so on).

You might 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, Bob 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 input files that contain both group-level and page-level indexing tags. In the case where ACIF indexes the print file, it supports indexing specific pages if you are using enhanced ACIF indexing. See Chapter 4, “Enhanced Indexing Parameters,” on page 77.

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 can contain records of variable length.

Begin Resource Group (BRG) Structured Field

ACIF assigns a null token name (X'FFFF') to this structured field and also creates several additional triplets, including: a Fully Qualified Name (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 MO:DCA-P output print file name identified by the DDname specified in the **OUTPUTDD** parameter.

Begin Resource (BRS) Structured Field

ACIF uses this structured field to delimit the resources in the file. ACIF also uses the X'21' triplet on the BRS structured field (also called the BR structured field) to identify the type of resource that follows this structured field. For more information, refer to *Mixed Object Document Content Architecture Reference*.

For TrueType and OpenType fonts, the eight-character name on the BRS structured field is always DOFFONT. The actual full font name is stored in an FQN triplet on the BRS structured field and is used to match the font to the Map Data Resources (MDR) structured field. For more information, refer to *Using OpenType Fonts in an AFP System*.

End Resource (ERS) and End Resource Group (ERG) Structured Fields

ACIF always assigns a null token name (X'FFFF') to the ERS and ERG structured fields it creates. The null name forces a match with the corresponding BRS and BRG structured fields. The ERS structured field is also called the ER structured field.

No Operation (NOP) Structured Field

NOP structured fields found *inside* inline resources are copied to the output resource library. NOP structured fields found *between* inline resources appear in the output AFP document following the Begin Document (BDT) structured fields (see “Begin Document (BDT) Structured Field” on page 211). NOP structured fields found *within* the line data or AFP input file are copied to the output file.

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:

```
BRG
  BRS
    AFP Resource 1
  ERS
  BRS
    AFP Resource 2
  ERS
  ..
  BRS
    AFP Resource n
  ERS
ERG
```

ACIF does not limit the number of resources that can be included in this object, but available storage is certainly a limiting factor.

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 index object 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 an index object file:

```
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.

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 supports page-level indexing if you are using enhanced ACIF indexing. See Chapter 4, “Enhanced Indexing Parameters,” on page 77.

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 AFP Workbench Viewer.

Begin Document Index (BDI) Structured Field

ACIF assigns a null token name (X'FFFF') and a Fully Qualified Name (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 page 38.

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 Count (X'58')
This triplet specifies the number of pages in a page group. This triplet applies only to group level IEL structured fields.
- 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.

ACIF can create three separate output files, one of which is the print file in AFP data stream format. In doing so, ACIF might create these structured fields:

- Tag Logical Element (TLE)
- Begin Named Group (BNG)
- End Named Group (ENG)

The TLE is described in Appendix C, “Format of the Index Object File,” on page 205; the other two structured fields are described later in this appendix. The examples on the next two pages illustrate the two possible AFP data stream document formats ACIF can 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 32. Example of Code Containing Group-Level Indexing

Figure 32 illustrates the format ACIF produces when it converts and indexes a print file with group-level indexing.

```

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 33. Example of Code Containing Group- and Page-Level Indexing

Figure 33 illustrates an input file that has already been indexed (tagged) and converted to MO:DCA-P format. 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 AFP Workbench Viewer, 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 a Fully Qualified Name (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 can contain multiple BDT structured fields, unless **INDEXOBJ=BDTLY** is specified, the ACIF output contains only one BDT structured field. (The same is true of End Document (EDT) structured fields.)

In the case of line data, 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 about the **CPGID** parameter, see page 38. ACIF assigns a null value (X'FFFF') to the Graphic Character Set Global Identifier.

ACIF 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.

ACIF also creates a comment triplet (X'65') that shows the current APAR level of the code used to build the AFP document.

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 and, unless **UNIQUEBNGS=NO** is specified, 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. Unless **UNIQUEBNGS=NO** is specified, 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 mentioned in “Tag Logical Element (TLE) Structured Field” on page 207, 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 data 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 an AFP input file 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. Unless **MCF2REF=CF** is specified, 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 might 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

| Inline resources at the beginning of the input file are copied to the resource file
| (RESOBJDD) if the appropriate **RESTYPE** value is specified. Inline resources are
| only copied to the output file (**OUTPUTDD**) if **RESTYPE=INLINE** is specified.

Page Definitions

Because page definitions are used only to compose line 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.

Appendix E. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, use software products successfully. The major accessibility features in z/OS let users:

- Use assistive technologies such as screen readers and screen magnifier software.
- Operate specific or equivalent features using only the keyboard.
- Customize display attributes such as color, contrast, and font size.

Using Assistive Technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard Navigation of the User Interface

Users can access z/OS user interfaces using TSO/E or ISPF. For more information, refer to *z/OS TSO/E Primer*, SA22–7787, *z/OS TSO/E User's Guide*, SA22–7794, and *z/OS ISPF User's Guide Vol I*, SC34–4822. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS Documentation

Most z/OS documentation is accessible using screen readers with the BookServer/Library Server versions of z/OS documents in the Internet library at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/>

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department 11PA, Building 002S
P.O. Box 1900
Boulder, CO 80301
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Programming Interfaces

This publication includes documentation of intended Programming Interfaces that let the customer write programs to obtain the services of ACIF.

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

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

Trademarks

The following terms are trademarks or registered trademarks of the IBM Corporation:

Advanced Function Presentation	MO:DCA
AFP	MVS
AIX	MVS/DFP
Bar Code Object Content Architecture	OS/390
BCOCA	OS/400
CICS	Print Services Facility
CICS/ESA	Quietwriter
@server	RACF
i5/OS	S/390
IBM	System z9
Infoprint	VM/ESA
Intelligent Printer Data Stream	VSE/ESA
IPDS	z/OS
Mixed Object Document Content Architecture	zSeries

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

- Microsoft and Windows are registered trademarks of Microsoft Corporation.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names might be trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in PSF for z/OS documentation. If you do not find the term you are looking for, see this publication's index or view *IBM Glossary of Computing Terms*, available from: <http://www.ibm.com/ibm/terminology>

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 (ISO) *Vocabulary—Information Processing* or from a published section of *Vocabulary—Office Machines* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Committee (ISO/IEC JTC1/SC1) 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 can 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), Joint Technical Committee 1 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 the 3820 printer.”

These cross-references are used in this glossary:

- **Contrast with.** Refers to a term that has an opposite or substantively different meaning.
- **See.** Refers to multiple-word terms in which this term appears.
- **See also.** Refers to related terms that have similar, but not synonymous, meanings.
- **Synonym for.** Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

- **Synonymous with.** Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning.

A

ACIF. See *AFP Conversion and Indexing Facility*.

Advanced Function Presentation (AFP). A set of licensed programs, together with user applications, that use the all-points-addressable concept to print data on a wide variety of printers or to display data on a variety of display devices. AFP includes creating, formatting, archiving, retrieving, viewing, distributing, and printing information.

AFP. See *Advanced Function Presentation*.

AFP Conversion and Indexing Facility (ACIF). An optional feature of PSF for z/OS that converts a print file into a MO:DCA-P document, creates an index file for later retrieval and viewing, and retrieves resources used by an AFP document into a separate file.

AFP Workbench Viewer. An IBM-licensed product that lets you display AFP and ASCII files at your Windows workstation in the same format they are printed.

| **AIX.** Advanced Interactive Executive. A UNIX operating
| system developed by IBM that is designed and
| optimized to run on POWER microprocessor-based
| hardware, such as servers, workstations, and blades.

alphanumeric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks.

American National Standards Institute (ANSI). A private, nonprofit organization whose membership includes private companies, U.S. government agencies, and professional, technical, trade, labor, and consumer organizations. ANSI coordinates the development of voluntary consensus standards in the U.S.

| **American Standard Code for Information**
| **Interchange (ASCII).** A standard code used for
| information exchange among data processing systems,
| data communication systems, and associated
| equipment. ASCII uses a coded character set consisting
| of 7-bit coded characters. Contrast with *EBCDIC*.

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. See *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. See *American Standard Code for Information Interchange*.

B

| **Bar Code Object Content Architecture™ (BCOCA).**
| An architected collection of constructs used to
| interchange and present bar code data.

BCOCA. See *Bar Code Object Content Architecture*.

C

carriage control character. An optional character in an input data record that specifies a write, space, or skip operation.

| **CCSID.** See *coded character set identifier*.

| **character.** (1) Any symbol that can be entered on a
| keyboard, printed, or displayed. For example, letters,
| numbers, and punctuation marks are all characters. (2)
| A member of a set of elements that is used for the
| representation, organization, or control of data.
| Characters can be letters, digits, punctuation marks, or
| other symbols represented in the form of a spatial
| arrangement of adjacent or connected strokes or in the
| form of other physical conditions in data media. A
| character can be a graphic character or a control
| character. (3) See also *graphic character*.

| **character rotation.** The alignment of a character with
| respect to its character baseline, measured in degrees
| in a clockwise direction. See *rotation*. See also
| *orientation*.

character set. A defined set of characters that can be recognized by a configured hardware or software system. See also *coded font* and *font character set*.

| **coded character set identifier (CCSID).** A 16-bit
| number that includes a specific set of encoding scheme
| identifiers, character set identifiers, code page
| identifiers, and other information that uniquely identifies
| the coded graphic-character representation.

coded font. In AFP support, a font file that associates a code page and a font character set. For double-byte fonts, a coded font associates multiple pairs of code pages and font character sets.

| **code page.** A particular assignment of code points to
| graphic characters. Within a given code page, a code
| point can have only one specific meaning. A code page
| also identifies how undefined code points are handled.
| See also *coded font*.

copy group. In PSF, An internal object in a form definition or a print data set that controls such items as modifications to a form, page placement, and overlays.

D

data stream. (1) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form using a defined format. (2) All information (data and control commands) sent over a data link, usually in a single read or write operation.

DCF. See *Document Composition Facility*.

| **document.** (1) A machine-readable collection of one or
| more objects that represent a composition, a work, or a
| collection of data. (2) Data that has already been
| composed into pages and that contains a Begin
| Document and an End Document structured field. (3) A
| publication or other written material.

| **Document Composition Facility (DCF).** An IBM
| licensed program used to format input to a printer.

| **download.** To transfer data from a computer to a
| connected device, such as a workstation or a
| microcomputer. Typically, users download from a large
| computer to a diskette or fixed disk on a smaller
| computer or from a system unit to an adapter.

E

EBCDIC. See *Extended Binary Coded Decimal Interchange Code*.

| **Extended Binary Coded Decimal Interchange Code (EBCDIC).** A coded character set of 256 eight-bit characters developed for the representation of textual data. EBCDIC is not compatible with ASCII character coding. Contrast with *ASCII*.

F

FOCA. See *Font Object Content Architecture*.

| **font.** A set of graphic characters with a common
| design that reflects the font designer's concept of how
| the graphic characters should appear. The common
| design specifies the characteristics of its graphic
| characters. Examples of characteristics are shape,
| graphic pattern, style, size, weight, and increment.
| Examples of fonts are fully described fonts, symbol sets,
| outline fonts, and their internal printer representations.
| See *coded font*.

font character set. Part of an AFP font that contains the raster patterns, identifiers, and descriptions of characters. See *character set*.

| **Font Object Content Architecture (FOCA).** An architecture that defines the content of IBM's digital font resources by means of a set of parameter definitions.

form definition. An AFP resource object used by PSF that defines the characteristics of the form or printed media, including: overlays to be used, duplex printing, text suppression, the position of composed-text data on the form, and the number and modifications of a page. Contrast with *page definition*.

G

GOCA. See *Graphics Object Content Architecture*.

| **graphic character.** (1) A visual representation of a character, other than a control character, that is normally produced by writing, printing, or displaying. (2) A member of a set of symbols that represent data. Graphic characters can be letters, digits, punctuation marks, or other symbols.

| **Graphics Object Content Architecture (GOCA).** An architecture that provides a collection of graphics values and control structures used to interchange and present graphics data.

group. A named collection of sequential pages that form a logical subset of a document.

H

| **hexadecimal.** Pertaining to a numbering system with base of 16.

I

image. A pattern of toned and untoned pels that form a picture.

| **Image Object Content Architecture (IOCA).** An architecture that provides a collection of constructs used to interchange and present images, such as printing image data on a page, page segment, or overlay.

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.

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.

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.

Infoprint Manager. A software component of IBM Infoprint that runs on AIX or Windows. Infoprint Manager handles the scheduling, archiving, retrieving, and assembly of a print job and its related resource files. It also tracks the finishing and packaging of the printed product.

| **inline resource.** A resource contained in a print file or a print data set.

| **Intelligent Printer Data Stream (IPDS).** An all-points-addressable data stream that allows users to position text, images, graphics, and bar codes at any defined point on a printed page. IPDS is the strategic AFP printer data stream generated by PSF.

IOCA. See *Image Object Content Architecture*.

IPDS. See *Intelligent Printer Data Stream*.

J

JCL. See *job control language*.

JES. See *Job Entry Subsystem*.

JES2. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

JES3. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In complexes that have several loosely coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue.

| **job control language (JCL).** A command language that is used to identify a job to an operating system and to describe the job's requirements.

| **Job Entry Subsystem (JES).** An IBM licensed program that receives jobs into the system and processes all output data that is produced by jobs.

L

| **library.** (1) A system object that serves as a directory to other objects. A library groups related objects, and allows the user to find objects by name. (2) A data file that contains copies of a number of individual files and

| control information that allows them to be accessed
| individually. A library exists as a partitioned data set or a
| series of concatenated partitioned data sets in z/OS. (3)
| In VSE, a collection of data stored in sublibraries on
| disk. A library consists of at least one sublibrary in
| which data is stored as members of various types such
| as phase, object module, or source book.

| **licensed program.** A separately priced program and
| its associated materials that bear a copyright and are
| offered to customers under the terms and conditions of
| a licensing agreement.

line data. Application data that is prepared for printing,
without any data placement or presentation information.
See *record format line data* and *traditional line data*.

M

Mixed Object Document Content Architecture (MO:DCA). An IBM-architected, device-independent data stream for interchanging documents.

| **Mixed Object Document Content Architecture for Presentation (MO:DCA-P).** The subset of MO:DCA
| that defines presentation documents.

MO:DCA. See *Mixed Object Document Content Architecture*.

MO:DCA-P. See *Mixed Object Document Content Architecture for Presentation*.

MO:DCA-P data. Print data that has been composed into pages. Text-formatting programs (such as DCF) can produce composed text data consisting entirely of structured fields.

| **MVS.** Multiple Virtual Storage. The primary operating
| system used on IBM mainframes. This operating system
| manages large amounts of memory and disk space.
| See *z/OS*.

O

| **object.** In AFP architecture, a collection of structured
| fields, bounded by a begin-object function and an
| end-object function. The object can contain other
| structured fields containing data elements of a particular
| type. Examples of objects are text, fonts, graphics,
| images, and bar codes.

offset. The number of measuring units from an arbitrary starting point in a record, area, or control block to some other point.

OpenType font. An extension of the TrueType font format that adds support for PostScript outlines and more support for international character sets and advanced typographic control.

| **orientation.** In printing, the number of degrees an
| object is rotated relative to a reference; for example, the
| orientation of an overlay relative to the logical page
| origin, or the orientation of printing on a page relative to
| the page coordinates. Orientation usually applies to
| blocks of information, whereas character rotation applies
| to individual characters. See also *text orientation* and
| *character rotation*.

OS/390. An IBM operating system for the IBM S/390 family of enterprise servers that includes and integrates functions previously provided by many IBM software products (including the MVS operating system). See *z/OS*.

| **outline font.** A font whose graphic character shapes
| are defined by mathematical equations rather than by
| raster patterns. Contrast with *raster font*.

| **overlay.** An AFP resource object that is a collection of
| predefined data, such as lines, shading, text, boxes, or
| logos, that can be merged with variable data on a page
| or form while printing. An overlay can be either an
| electronic overlay or a forms flash. Contrast with *page segment*.

P

| **page.** (1) A collection of data that can be printed on
| one side of a sheet of paper or a form. (2) A data
| stream object delimited by a Begin Page structured field
| and an End Page structured field. A page can contain
| presentation data such as text, image, graphics, and bar
| code data.

| **page definition.** An AFP resource object used by PSF
| that defines the rules for transforming line data and
| XML data into MO:DCA-P data and text controls.
| Contrast with *form definition*.

| **page segment.** An AFP resource object containing
| text, image, graphics, or bar code data that can be
| positioned on any addressable point on a page or an
| electronic overlay. Contrast with *overlay*.

| **parameter.** A value or reference passed to a function,
| command, or program that serves as input or to control
| actions. The value is supplied by a user or by another
| program or process.

pitch. A unit of measurement for the width of type (or a printed character), based on the number of characters that can be set (or printed) in one linear inch; for example, 10-pitch has 10 characters per inch. Uniformly spaced fonts are measured in pitch. Contrast with *point*.

| **point.** A unit of measurement used mainly for
| describing type sizes. Each pica has 12 points, and an
| inch has approximately 72 points. Contrast with *pitch*

point size. The height of a font in points.

| **Print Services Facility (PSF).** An IBM licensed
| program that manages and controls the input data
| stream and output data stream required by supported
| IBM page printers. PSF is supported under z/OS, VSE,
| VM, and i5/OS operating systems.

PSF. See *Print Services Facility*.

R

raster font. A font in which the characters are defined directly by the raster bit map. Contrast with *outline font*.

record format line data. A form of line data where each record is preceded by a 10-byte identifier.

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 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. See *character*
| *rotation*. See also *orientation*.

S

structured field. A self-identifying string of bytes and its data or parameters.

syntax. The programming language rules for constructing a command or statement.

T

tag. A type of structured field used for indexing in an AFP document. Tags associate an index attribute-value pair with a specific page or group of pages in a document.

text orientation. A description of the appearance of text as a combination of print direction and character rotation.

traditional line data. A form of line data that is prepared for printing on a line printer, such as 6262 or 3211.

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 also *anchor point*.

TrueType font. A font format based on scalable outline technology in which the graphic character shapes are based on quadratic curves. The font is described with a set of tables contained in a TrueType font file.

| **typeface.** All characters of a single type family or
| style, weight class, width class, and posture, regardless
| of size. An example is Helvetica bold condensed italic,
| in any point size.

U

| **Unicode.** A universal character encoding standard that
| supports the interchange, processing, and display of
| text that is written in any of the languages of the
| modern world. It also supports many classical and
| historical texts in a number of languages. The Unicode
| standard has a 16-bit international character set defined
| by ISO 10646.

V

VM. Virtual machine. A functional simulation of a computer and its associated devices.

| **VSE.** Virtual Storage Extended. A system that consists
| of a basic operating system (VSE/Advanced Functions),
| and any IBM supplied and user-written programs
| required to meet the data processing needs of a user.
| VSE and the hardware that it controls form a complete
| computing system.

W

Workbench Viewer. See *AFP Workbench Viewer*.

X

| **XML data.** Data identified with the Extensible Markup
| Language (XML), which is a standard metalanguage for
| defining markup languages that is based on Standard
| Generalized Markup Language (SGML). For printing on
| page printers, a page definition is required to provide
| the data placement and presentation information. The
| XML data processed by ACIF can be encoded in
| EBCDIC, ASCII, UTF-8 or UTF-16.

Z

| **z/OS.** An IBM operating system for the IBM zSeries
| and System z9 families of enterprise servers that
| includes and integrates functions previously provided by
| many IBM software products (including the MVS and
| OS/390 operating systems).

z/OS.e. An IBM operating system available on the IBM
@server® zSeries 800 (z800) and zSeries 890 (z890)
that provides select z/OS function at a reduced price.

Bibliography

This bibliography lists the titles of publications containing additional information about PSF, AFP, the z/OS operating system, and related products.

The titles and order numbers may change from time to time. To verify the current title or order number, consult your IBM marketing representative.

You can obtain many of the publications listed in this bibliography from the Printing Systems support and maintenance Web page:

http://www.printers.ibm.com/internet/wwsites.nsf/vwwebpublished/supportmanuals_ww

Print Services Facility (PSF) for z/OS

Publication	Order Number
<i>AFP Conversion and Indexing Facility: User's Guide</i>	S550-0436
<i>PSF for z/OS: AFP Download Plus</i>	S550-0433
<i>PSF for z/OS: Customization</i>	S550-0427
<i>PSF for z/OS: Diagnosis</i>	G550-0428
<i>PSF for z/OS: Download for z/OS</i>	S550-0429
<i>PSF for z/OS: Introduction</i>	G550-0430
<i>PSF for z/OS: Licensed Program Specifications</i>	G550-0431
<i>PSF for z/OS: Messages and Codes</i>	G550-0432
<i>PSF for z/OS: Security Guide</i>	S550-0434
<i>PSF for z/OS: User's Guide</i>	S550-0435
<i>PSF for AIX: AFP Upload Configuration Guide Using SNA</i>	S544-5422
<i>PSF for AIX: AFP Upload Configuration Guide Using TCP/IP</i>	S544-5423
<i>PSF for z/OS Collection Kit CD-ROM (BookManager)</i>	SK3T-9927
<i>PSF for z/OS Collection Kit CD-ROM (PDF)</i>	SK3T-9928

Infoprint Server

Title	Order Number
<i>Infoprint Server Transforms Licensed Program Specifications</i>	G544-5797
<i>z/OS Infoprint Server Customization</i>	S544-5744
<i>z/OS Infoprint Server Introduction</i>	S544-5742
<i>z/OS Infoprint Server Messages and Diagnosis</i>	G544-5747
<i>z/OS Infoprint Server Operation and Administration</i>	S544-5745
<i>z/OS Infoprint Server User's Guide</i>	S544-5746

Advanced Function Presentation (AFP)

Publication	Order Number
<i>Advanced Function Printing: Host Font Data Stream Reference</i>	S544-3289
<i>AFP Conversion and Indexing Facility: User's Guide</i>	S550-0436
<i>AFP Toolbox User's Guide</i>	G544-5292
<i>AFP Workbench for Windows: Using the Viewer Application</i>	G544-3813
<i>Guide to Advanced Function Presentation</i>	G544-3876
<i>IBM Printing Systems: Printer Information</i>	S544-5750
<i>IBM Printing Systems: Printer Summary</i>	S544-5749
<i>Overlay Generation Language User's Guide and Reference</i>	S544-3700
<i>Page Printer Formatting Aid: User's Guide</i>	S544-5284
Architecture	
<i>Advanced Function Presentation: Programming Guide and Line Data Reference</i>	S544-3884
<i>Bar Code Object Content Architecture Reference</i>	S544-3766
<i>Font Object Content Architecture Reference</i>	S544-3285
<i>Graphics Object Content Architecture Reference</i>	SC31-6804
<i>Using OpenType Fonts in an AFP System</i>	G544-5876
<i>Image Object Content Architecture Reference</i>	SC31-6805
<i>Intelligent Printer Data Stream Reference</i>	S544-3417
<i>Mixed Object Document Content Architecture Reference</i>	SC31-6802
<i>Presentation Text Object Content Architecture Reference</i>	SC31-6803
Fonts	
<i>IBM AFP Fonts: Font Samples</i>	S544-3792
<i>IBM AFP Fonts: Font Summary for AFP Font Collection</i>	S544-5633
<i>IBM AFP Fonts: Type Transformer User's Guide</i>	G544-3796
<i>IBM Infoprint Fonts: Font Summary</i>	G544-5846
<i>IBM Infoprint Fonts: Introduction to Type Transformer and Utilities for Windows</i>	G544-5853

Data Transforms

Publication	Order number
<i>Infoprint XML Extender for z/OS</i>	S544-5855
<i>Infoprint XT Extender for z/OS: Customization and Usage</i>	S544-5879
<i>Infoprint XT Extender for z/OS: Messages and Diagnosis</i>	G544-5878

Text Processing

Publication	Order number
<i>DCF/DLF General Information</i>	GH20-9158

Publication	Order number
<i>Document Composition Facility: Bar Code User's Guide</i>	S544-3115
<i>Document Composition Facility: SCRIPT/VS Text Programmer's Guide</i>	SH35-0069
<i>Document Composition Facility: SCRIPT/VS Language Reference</i>	SH35-0070
<i>Publishing Systems BookMaster General Information</i>	GC34-5006
<i>Publishing Systems BookMaster User's Guide</i>	SC34-5009

Infoprint Manager

Publication	Order Number
<i>IBM Infoprint Manager for AIX: Getting Started</i>	G544-5817
<i>IBM Infoprint Manager for AIX: Introduction and Planning Guide</i>	G544-5833
<i>IBM Infoprint Manager for AIX: Procedures</i>	G544-5815
<i>IBM Infoprint Manager for Windows: Getting Started</i>	G544-5717
<i>IBM Infoprint Manager for Windows: Introduction and Planning Guide</i>	G544-5716
<i>IBM Infoprint Manager for Windows: Procedures</i>	G544-5814
<i>IBM Infoprint Manager for Windows: PSF Direct Network Configuration Guide</i>	Web-based
<i>IBM Infoprint Manager: Reference</i>	S544-5475
<i>PSF for AIX: AFP Upload Configuration Guide Using SNA</i>	S544-5422
<i>PSF for AIX: AFP Upload Configuration Guide Using TCP/IP</i>	S544-5423

Content Manager OnDemand

Publication	Order Number
<i>IBM DB2 Content Manager OnDemand for Multiplatforms: Administration Guide</i>	SC18-9237
<i>IBM DB2 Content Manager OnDemand for Multiplatforms: Indexing Reference</i>	SC18-9235
<i>IBM DB2 Content Manager OnDemand for Multiplatforms: Installation and Configuration Guide</i>	SC18-9232
<i>IBM DB2 Content Manager OnDemand for Multiplatforms: Introduction and Planning Guide</i>	GC18-9236
<i>IBM DB2 Content Manager OnDemand: Messages and Codes</i>	SC27-1379
<i>IBM DB2 Content Manager OnDemand: Report Distribution Installation, Use, Reference</i>	SC18-9233
<i>IBM DB2 Content Manager OnDemand: User's Guide</i>	SC27-0836
<i>IBM DB2 Content Manager OnDemand for Multiplatforms: Web Enablement Kit Implementation Guide</i>	SC18-9231
<i>IBM DB2 Content Manager OnDemand: Windows Client Customization Guide</i>	SC27-0837

i-data

Publication	Order Number
<i>i-data 7913 IPDS Printer LAN Attachment for Ethernet Installation Guide</i>	none
<i>i-data 7913 IPDS Printer LAN Attachment for Token Ring Installation Guide</i>	none

z/OS Version 1 Release 7

Publication	Order Number
<i>SMP/E User's Guide</i>	SA22-7773
<i>z/OS Collection</i>	SK3T-4269
<i>z/OS and Software Products DVD Collection</i>	SK3T-4271
<i>z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference</i>	SC31-8788
<i>z/OS Communications Server: IP Configuration Reference</i>	SC31-8776
<i>z/OS Communications Server: IP and SNA Codes</i>	SC31-8791
<i>z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures</i>	GC31-6850
<i>z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT</i>	GC31-6851
<i>z/OS Communications Server: SNA Messages</i>	SC31-8790
<i>z/OS Communications Server: SNA Network Implementation Guide</i>	SC31-8777
<i>z/OS Communications Server: SNA Operation</i>	SC31-8779
<i>z/OS Communications Server: SNA Programming</i>	SC31-8829
<i>z/OS Communications Server: SNA Resource Definition Reference</i>	SC31-8778
<i>z/OS DFSMS Macro Instructions for Data Sets</i>	SC26-7408
<i>z/OS DFSMS Using Data Sets</i>	SC26-7410
<i>z/OS DFSMSdfp Utilities</i>	SC26-7414
<i>z/OS HCD Planning</i>	GA22-7525
<i>z/OS HCD User's Guide</i>	SC33-7988
<i>z/OS HCD Scenarios</i>	SC33-7987
<i>z/OS JES2 Commands</i>	SA22-7526
<i>z/OS JES2 Initialization and Tuning Guide</i>	SA22-7532
<i>z/OS JES2 Initialization and Tuning Reference</i>	SA22-7533
<i>z/OS JES2 Messages</i>	SA22-7537
<i>z/OS JES3 Commands</i>	SA22-7540
<i>z/OS JES3 Initialization and Tuning Guide</i>	SA22-7549
<i>z/OS JES3 Initialization and Tuning Reference</i>	SA22-7550
<i>z/OS JES3 Messages</i>	SA22-7552
<i>z/OS Language Environment Programming Guide</i>	SA22-7561
<i>z/OS MVS Diagnosis: Procedures</i>	GA22-7587
<i>z/OS MVS Diagnosis: Tools and Service Aids</i>	GA22-7589

Publication	Order Number
<i>z/OS MVS Initialization and Tuning Reference</i>	SA22-7592
<i>z/OS MVS IPCS User's Guide</i>	SA22-7596
<i>z/OS MVS IPCS Commands</i>	SA22-7594
<i>z/OS MVS JCL Reference</i>	SA22-7597
<i>z/OS MVS JCL User's Guide</i>	SA22-7598
<i>z/OS MVS Programming: Authorized Assembler Services Guide</i>	SA22-7608
<i>z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN</i>	SA22-7609
<i>z/OS MVS Programming: Authorized Assembler Services Reference ENF-IXG</i>	SA22-7610
<i>z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU</i>	SA22-7611
<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>	SA22-7612
<i>z/OS MVS Recovery and Reconfiguration Guide</i>	SA22-7623
<i>z/OS MVS System Codes</i>	SA22-7626
<i>z/OS MVS System Commands</i>	SA22-7627
<i>z/OS MVS System Management Facilities (SMF)</i>	SA22-7630
<i>z/OS MVS Using the Functional Subsystem Interface</i>	SA22-7641
<i>z/OS SDSF Operation and Customization</i>	SA22-7670
<i>z/OS Security Server RACF General User's Guide</i>	SA22-7685
<i>z/OS Security Server RACF Security Administrator's Guide</i>	SA22-7683
<i>z/OS Security Server RACF System Programmer's Guide</i>	SA22-7681
<i>z/OS Support for Unicode: Unicode Services</i>	SA22-7649
<i>z/OS UNIX System Services Command Reference</i>	SA22-7802

Index

A

- accessibility 215
- acif command
 - automatically invoking 21
 - line2afp 21
 - parameters 31
 - pdpr 21
 - running ACIF with 106
 - syntax rules 30
- ACIF JCL statement, z/OS 23
- AFP Conversion and Indexing Facility (ACIF)
 - AFP data as input to 3
 - AFP Toolbox 16
 - AFP Workbench Viewer 15
 - application, example of using 95
 - converting data streams 3
 - Document Composition Facility (DCF) 16
 - document files, viewing 108
 - enhanced indexing parameters 77
 - enhanced indexing, example of using 109
 - functions 3
 - indexing functions 5
 - invoking in
 - AIX 21
 - VM 25
 - VSE 26
 - Windows 21
 - z/OS 23
 - messages 127
 - output file format 212
 - overview 1
 - parameters 29
 - preparing files for
 - archiving and retrieval 13
 - printing 11
 - viewing 10
 - processing parameters
 - examples of 91
 - specifying 100
 - related IBM products 14
 - retrieving resources 9
 - running jobs 106
 - scenarios 10
 - syntax rules 29
 - understanding 1
 - user exits 113
 - using 19
- AFP data stream as input 4
- AFP Toolbox 16
- AFP Workbench Viewer
 - description of 15
 - fonts used to display documents 15
 - group names used by 49
 - ignored objects 15
 - indexing for 52
 - preparing files for viewing with 10
 - retrieving resources for 71
- AFP Workbench Viewer (*continued*)
 - viewing document files with 108
- AIX
 - acif command 30
 - concatenating output files 107
 - converting literal values 99
 - examples
 - ASCII input data 99
 - drawing graphics 95
 - EBCDIC input data 99
 - locating resource libraries 94
 - parameter file 100
 - resource retrieval 92
 - specifying coded fonts 93
 - specifying TrueType and OpenType fonts 94
 - transforming line or XML data 91
 - files provided with ACIF 22
 - input record exits 116
 - invoking ACIF automatically 21
 - line2afp 21
 - literal values for
 - ASCII data 99
 - EBCDIC data 99
 - mounting directories on workstations 109
 - NLS messages 22
 - pdpr 21
 - sample user exits 113
 - search order for
 - resources 20
 - user exit load modules 124
 - shell commands 99
 - syntax rules for ACIF 30
 - system prerequisites 17
 - transferring files to 194
 - using ACIF in 19
- anchor point, definition of 7
- anchor record
 - FIELDn parameter 41, 78
 - set by INDEXn parameter 49, 84
 - set by TRIGGERn parameter 72, 87
- ANSI carriage-control characters
 - ASCII, encoded in 34
 - EBCDIC, encoded in 34
 - indexing considerations 196
 - machine code 34
 - specifying 33
 - type of, specifying 33
 - using 192
- apka2e exit program
 - setting parameters for 117
 - specifying 54
- APKACIF
 - CMS statement, VM 25
 - JCL statement, VSE 27
- archiving
 - indexing considerations 196
 - indexing data for 5
 - preparing files for 13

- archiving (*continued*)
 - retrieving resources for 9
- ASCII
 - converting to EBCDIC 54
 - data, literal values for 99
 - encoded carriage control characters 34
 - input data to ACIF 5
 - parameter file for input data 99
- asciinp input record exit 117
- asciipe input record exit 117
- attribute name for indexing, specifying 49, 85
- attributes
 - indexing 49, 85
 - print file 125
- axeb command 100

B

- BDI structured field 206
- BDT structured field 211
- Begin Document (BDT) structured field 211
- Begin Document Index (BDI) structured field 206
- Begin Named Group (BNG) structured field 211
- Begin Page (BPG) structured field 212
- Begin Resource (BRS) structured field 203
- Begin Resource Group (BRG) structured field 202
- blank characters in parameter file 29
- BNG structured field 211
- BPG structured field 212
- BR structured field 203
- BRG structured field 202
- BRS structured field 203

C

- carriage-control characters
 - ASCII, encoded in 34
 - EBCDIC, encoded in 34
 - indexing considerations 196
 - machine code 34
 - specifying 33
 - type of, specifying 33
 - using 192
- CC parameter
 - format 33
 - print file attribute 125
- CCSID, converting encoded data to 53, 54, 58, 117
- CCTYPE parameter
 - format 33
 - print file attribute 125
- CHARS parameter
 - format 34
 - print file attribute 125
 - TRCs, font order 72
- CMS commands
 - concatenating VM files with 108
 - for ACIF jobs 25
 - VM parameter file 103
- code page identifier 37
- coded fonts for MCF-2 structured fields 123

- COM setup file
 - in OBJCONLIB 56
 - in the resource file 123
 - inline resource requirements 37
 - specified in setup parameter 36
- commands
 - concatenation 107
 - shell, converting literal values 99
- comments in parameter file 29
- Composed Text Control (CTC) structured field 212
- COMSETUP parameter 36
- concatenation
 - AIX files 107
 - output files 107
 - resource group to document 197
 - VM files 108
 - Windows files 108
 - z/OS files 108
- constant fields 80
- control statements with numbered lines 191
- conventions, syntax 29
- converting
 - ASCII data for printing 5
 - ASCII to EBCDIC 54
 - data streams 3
 - encoded data to CCSID 53, 54, 58, 117
 - line or XML data, example 91
 - literal values, example 99
- CPGID parameter 37
- CTC structured field 212

D

- data fields, specifying 41, 78
- data sets, specifying
 - FOCA font 45
 - form definition 40
 - object container 57
 - overlay 60
 - page definition 63
 - page segment 65
 - user 75
- data streams, converting 3
- data values, indexing with 7
- DCB requirements
 - index object file 51
 - on INDEX JCL statement 24
 - on OUTPUT JCL statement 24
 - on RESOBJ JCL statement 24
 - output file 59
 - resource file 68
- DCF
 - adding indexing tags 16
 - support for the TLE structured field 201
- DCFPAGENAMES parameter 38
- DD statement
 - VM 25
 - z/OS 23
- directories
 - example of locating 94
 - FOCA font 45

- directories *(continued)*
 - form definition 40
 - mounting on workstations 109
 - object container 56
 - overlay 59
 - page definition 63
 - page segment 65
 - resource
 - example of locating 94
 - locations of 106
 - TrueType and OpenType font 46, 75
 - user 74
- disability 215
- distributed printing
 - limitations 16
 - preparing files for 11
- DLBL statements for VSE 26
- Document Composition Facility (DCF)
 - adding indexing tags 16
 - support for the TLE structured field 201
- document files
 - concatenating 107
 - creating 107
 - indexing 5
 - transferring to workstations 109
 - viewing 108
- document output format 209
- double-byte fonts required in SOSI process 64
- Download for z/OS, transferring files with 195
- dummy definition
 - form 48
 - page 62

E

- EBCDIC
 - converting ASCII to 54
 - data, literal values for 99
 - encoded carriage control characters 34
 - parameter file for input data 99
 - using the shell with literal values 99
- EDI structured field 207
- EDT structured field 212
- encoded carriage-control characters
 - ASCII 34
 - EBCDIC 34
- encoded data, converting to CCSID 53, 54, 58, 117
- End Document (EDT) structured field 212
- End Document Index (EDI) structured field 207
- End Named Group (ENG) structured field 212
- End Page (EPG) structured field 212
- End Resource (ERS) structured field 203
- End Resource Group (ERG) structured field 203
- ENG structured field 212
- enhanced indexing 77
- enhanced indexing parameter file 111
- environment variables
 - PATH
 - index record exit 53
 - input record exit 53
 - output record exit 58

- environment variables *(continued)*
 - PATH *(continued)*
 - resource exit 66
 - PSFPATH 36
- EPG structured field 212
- ER structured field 203
- ERG structured field 203
- error message files, naming 56
- error return code 310 198
- ERS structured field 203
- examples
 - ACIF application 95
 - ACIF output for indexed input file 210
 - AFP document output formats 209
 - AIX and Windows parameter file 100
 - ASCII input data, parameter file for 99
 - CMS commands to invoke ACIF 25
 - drawing graphics 95
 - EBCDIC input data, parameter file for 99
 - enhanced indexing 109
 - enhanced indexing parameter file 111
 - index record exit 118
 - indexing input files 104
 - indexing with data values 7
 - input file 97
 - input record exit 114
 - line data application 95
 - literal values in AIX 99
 - locating resource libraries 94
 - output record exit 119
 - print file attributes 125
 - processing parameters 100
 - resource exit 121
 - retrieving resources 92
 - specifying coded fonts 92
 - specifying TrueType and OpenType fonts 93
 - transforming line or XML data 91
 - using ACIF 91
 - VM parameter file 103
 - VSE JCL to invoke ACIF 26
 - VSE parameter file 103
 - z/OS JCL to invoke ACIF 23
 - z/OS parameter file 101
- exits
 - apka2e 117
 - asciinp 117
 - asciinp 117
 - index record 53, 118
 - input record 53, 114
 - non-zero return codes 113
 - output record 58, 119
 - print file attributes provided 125
 - provided with ACIF
 - index record 118
 - input record 114
 - output record 119
 - resource 121
 - resource
 - naming 66
 - overview 121
 - search order 124

exits (*continued*)
 user programming 113
EXTENSIONS parameter 39

F

FDEFLIB parameter 40
FIELDn parameter
 format 41, 78
 specified in multiple INDEXn parameters 50, 85
 specifying with indexing data 105
fields
 constant 80
 data, specifying 41, 78
 indexing 50, 85
 inserting IMM structured 55
 mask 82
 specifying MCF-2 structured 56
 specifying with indexing data 105
 transaction 80
 trigger 78
file extensions, search order for resources 20
file transfer 193
FILEFORMAT parameter 43
files
 concatenating output 107
 fixed-length 193
 preparing for
 archiving and retrieval 13
 printing 11
 viewing 10
 provided with ACIF 22
 transferring to AIX or Windows 194
 variable-length 193
 viewing document 108
fixed-length files 193
FONTECH parameter 44
FONTLIB parameter 45
FONTPATH parameter 46
fonts
 3800 44
 converting ASCII to EBCDIC 54
 converting one CCSID to another 53, 54, 58, 117
 double-byte requirements in SOSI process 64
 example of specifying coded 92
 example of specifying TrueType and OpenType 93
 for line data 35
 libraries for FOCA 45
 libraries for TrueType and OpenType 46, 75
 specifying with CHARS 34
 TRC assignment 72
 unbounded box 44
 used by AFP Workbench Viewer 15
form definition
 inline resource requirements 48
 libraries 40
 specified with FORMDEF 47
 specifying 106
format of input file 43
FORMDEF parameter
 format 47

FORMDEF parameter (*continued*)
 print file attribute 126
FTP, transferring files with 109, 195

G

group name, unique 74
group-level IEL structured field 205
GROUPMAXPAGES parameter 84
GROUPNAME parameter 49
groups for indexing
 definition of 5
 GROUPMAXPAGES parameter 84
 GROUPNAME parameter 49
 structured fields 202, 211
groups, page 210
GTF trace 71

I

iconv command 100
IEL structured field
 group-level 205
 index object file 196
 page-level 206
 triplets composing 206
IMAGEOUT parameter
 format 49
 hints for specifying 198
IMM structured field
 hints for creating 195
 inserting 55
INDEX
 CMS statement, VM 25
 parameter 49, 84
 specifying for indexing 105
 VSE JCL statement 27
 z/OS JCL statement 24
Index Element (IEL) structured field
 group-level 205
 index object file 196
 page-level 206
 triplets composing 206
index object file
 archiving considerations 196
 concatenating 107
 creating 107
 DCB characteristics 51
 defining for
 VM 25
 VSE 27
 z/OS 24
 description of 5
 naming 51
 structured fields 205
 type of information in 52
 used by AFP Workbench Viewer 15
index record exit
 examples 118
 naming 53
 overview 118

- INDEXDD parameter 51
- indexing
 - ACIF application, tasks for 97
 - anchor point, definition of 7
 - anchor record 41, 72, 78, 87
 - description of 5
 - effect on document 209
 - enhanced 77
 - example 104
 - FIELDn parameter 41, 78
 - functions 5
 - helpful hints 196
 - input files 6
 - limitations 8
 - page ACIF should start 52
 - page-level information 87
 - parameter 49, 84
 - record exit 118
 - structured fields 201
 - TLE structured field 201
 - to print output file 17
 - trigger parameter 72, 87
 - triggers, definition of 8
 - with data values 7
 - with DCF 16
 - with literal values 6
- indexing tags
 - definition of 6
 - End Resource Group structured field 203
 - End Resource structured field 203
 - example 105
 - INDEXn parameter 49, 84
 - structured fields 202, 203
- INDEXn parameter 49, 84
- INDEXOBJ parameter 52
- INDEXSTARTBY parameter 52
- INDEXEXIT parameter 53
- inline resources
 - COM setup files 37
 - form definition 48
 - order of 197
 - page definition 62
 - processing 197
 - specified with RESTYPE 70
- INPCCSID parameter 53
- INPEXIT parameter 53
- INPUT
 - CMS commands, VM 25
 - VSE JCL statement 27
 - z/OS JCL statement 24
- input data streams, processing 3
- input file
 - example 97
 - exit 114
 - format, specifying 43
 - indexing 104
 - naming 54
- input print file 125
- input record exit
 - apka2e 117
 - asciinp 117

- input record exit *(continued)*
 - asciinp 117
 - examples 114
 - naming 53
 - overview 114
- INPUTDD parameter 54
- INSERTIMM parameter 55
- Invoke Medium Map (IMM) structured field
 - hints for creating 195
 - inserting 55

J

- JCL
 - enhanced indexing parameter file 111
 - for concatenating z/OS files 108
 - for VSE jobs 26
 - for z/OS jobs 23
 - statements
 - VSE 26
 - z/OS 23
 - VSE parameter file 103
 - z/OS parameter file 101

K

- keyboard 215

L

- libraries
 - example of locating 94
 - FOCA font 45
 - form definition 40
 - mounting on workstations 109
 - object container 56
 - overlay 59
 - page definition 63
 - page segment 65
 - resource
 - example of locating 94
 - locations of 106
 - TrueType and OpenType font 46, 75
 - user 74
- limitations
 - indexing 8
 - printing 16
- line data
 - as input 4
 - carriage controls 33
 - fonts 35
 - input to ACIF indexing 6
 - search paths for resources
 - FOCA fonts 45
 - form definitions 40
 - overlays 59
 - page definitions 63
 - page segments 65
 - system resources 68
 - TrueType and OpenType fonts 46, 75
 - user resources 74

- line2afp, relation to acif command 21
- literal values
 - converting 99
 - determining how expressed 98
 - indexing with 6
- locations of resources libraries, identifying 106
- LookAt message retrieval tool xvi

M

- machine code carriage control characters 34
- Map Coded Font Format 1 (MCF-1) structured field 212
- Map Coded Font Format 2 (MCF-2) structured field
 - coded fonts 123
 - modification 213
 - specifying 56
- mask fields 82
- MCF-1 structured field 212
- MCF-2 structured field
 - coded fonts 123
 - modification 213
 - specifying 56
- MCF2REF parameter 56
- members, partitioned data set
 - concatenating output 107
 - fixed-length 193
 - preparing for
 - archiving and retrieval 13
 - printing 11
 - viewing 10
 - provided with ACIF 22
 - transferring to AIX or Windows 194
 - variable-length 193
 - viewing document 108
- message files
 - creating 107
 - defining for
 - VM 25
 - VSE 27
 - z/OS 24
 - naming, error 56
- message retrieval tool, LookAt xvi
- messages
 - ACIF 127
 - NLS 22
- mixed-mode data as input 4
- MO:DCA-P data stream as input 4
- MO:DCA-P object containers, creating 198
- mounting directories on workstations 109
- MSGDD parameter 56
- multiple-up output from page definition 196

N

- named groups, placing TLEs in 191
- new-line character 43
- NLS messages 22
- No Operation (NOP) structured field 203
- non-zero return codes 113
- numbered lines in control statements 191

O

- OBJCONLIB parameter 56
- object containers, creating MO:DCA-P 198
- OpenType fonts, specifying 93
- order of inline resources 197
- OS/390
 - See z/OS
- out-of-storage problem, possible cause 191
- OUTCCSID parameter 58
- OUTEXIT parameter 58
- OUTPUT
 - CMS commands, VM 25
 - VSE JCL statement 27
 - z/OS JCL statement 24
- output files
 - concatenating 107
 - creating 107
 - format 209
 - MO:DCA-P data stream 212
 - naming 59
 - processing inline resources to 197
- output record exit
 - examples 119
 - naming 58
 - overview 119
- OUTPUTDD parameter 59
- overlay library 59
- overview of ACIF 1
- OVLYLIB parameter 59

P

- page definition
 - excluded from resource file 213
 - fonts 35
 - inline resource requirements 62
 - libraries 63
 - multiple-up output 196
 - specified with PAGEDEF 61
 - specifying 106
- page groups 210
- page segment library 65
- page-level
 - IEL structured field 206
 - information, indexing 87
- PAGEDEF parameter
 - format 61
 - print file attribute 126
- parameter files
 - AIX and Windows, example of 100
 - ASCII input data, example of 99
 - blank characters in 29
 - comments in 29
 - creating output files with 107
 - defining
 - for VM 25
 - for z/OS 23
 - resource libraries 106
 - determining form and page definitions 106
 - EBCDIC input data, example of 99

- parameter files (*continued*)
 - enhanced indexing, example of 111
 - examples of 100
 - specifying 100
 - syntax rules 29
 - using to run ACIF jobs 106
 - values spanning multiple records 30
 - VM, example of 103
 - VSE, example of 103
 - z/OS, example of 101
- parameter values spanning multiple records 30
- parameters
 - ACIF 29
 - CC 33
 - CCTYPE 33
 - CHARS 34
 - COMSETUP 36
 - CPGID 37
 - DCFPAGENAMES 38
 - enhanced indexing 77
 - EXTENSIONS 39
 - FDEFLIB 40
 - FIELDn 41, 78
 - FILEFORMAT 43
 - FONTECH 44
 - FONTLIB 45
 - FONTPATH 46
 - FORMDEF 47
 - GROUPMAXPAGES 84
 - GROUPNAME 49
 - IMAGEOUT 49
 - INDEXDD 51
 - INDEXn 49, 84
 - INDEXOBJ 52
 - INDEXSTARTBY 52
 - INDEXEXIT 53
 - INPCCSID 53
 - INPEXIT 53
 - INPUTDD 54
 - INSERTIMM 55
 - MCF2REF 56
 - MSGDD 56
 - OBJCONLIB 56
 - OUTCCSID 58
 - OUTEXIT 58
 - OUTPUTDD 59
 - OVLYLIB 59
 - PAGEDEF 61
 - PARMDD 63
 - PDEFLIB 63
 - PRMODE 64
 - PSEGLIB 65
 - RESEXIT 66
 - RESFILE 67
 - RESLIB 68
 - RESOBJDD 68
 - RESTYPE 69
 - syntax rules 29
 - TRACE 71
 - TRC 72
 - TRIGGERn 72, 87
- parameters (*continued*)
 - types in AIX and Windows 21
 - UNIQUEBNGS 74
 - USERLIB 74
 - USERMASK 89
 - USERPATH 75
 - values for 31
 - values for enhanced indexing 77
- PARMDD parameter 63
- partitioned data sets, definition of 9
- PATH environment variable
 - index record exit 53
 - input record exit 53
 - output record exit 58
 - resource exit 66
 - searching for user exits 124
- paths
 - FOCA font directory 45
 - form definition directory 40
 - object container directory 56
 - overlay directory 59
 - page definition directory 63
 - page segment directory 65
 - resource directory 68
 - TrueType and OpenType font directory 46, 75
 - user resource directory 74
- PC file transfer program, transferring files with 194
- PDEFLIB parameter 63
- pdpr, relation to acif command 21
- physical media, transferring files with 194
- PRD2 JCL statement, VSE 27
- prerequisites
 - application programmer xiii
 - system 17
- Presentation Text Data Descriptor (PTD) structured field 213
- print file attributes
 - provided to exits 125
 - user exits 113
- printing
 - limitations 16
 - preparing files for 11
- PRINTOUT JCL statement, z/OS 23
- PRMODE parameter
 - format 64
 - print file attribute 126
- PRNTOUT JCL statement, VSE 26
- processing inline resources 197
- processing mode 64
- processing parameters
 - ASCII input data 99
 - EBCDIC input data 99
 - specifying 100
- programming interface information
 - AFP Toolbox 16
 - overview 218
- PSEGLIB parameter 65
- PSF/VSE limitations 16
- PSFPATH environment variable 36
- PTD structured field 213
- publications, related 227

R

- record format page definitions, drawing graphics with 95
- related products 14
- related publications 227
- RESEXIT parameter 66
- RESFILE parameter 67
- RESLIB parameter 68
- RESOBJ
 - CMS statement, VM 25
 - JCL statement
 - VSE 27
 - z/OS 24
- RESOBJDD parameter 68
- resource exit
 - examples 121
 - filtering resources 9
 - naming 66
 - overview 121
- resource files
 - AFP data stream resource group 9
 - concatenating 107
 - creating 107
 - DCB characteristics 68
 - defining for
 - VM 25
 - VSE 27
 - z/OS 24
 - format 203
 - naming 68
 - output format 9
 - partitioned data set, z/OS 9
 - processing inline resources to 197
 - structured fields 202
 - type created 67
 - type of resources retrieved for 69
- resource libraries
 - example of locating 94
 - locations of 106
- resource retrieval
 - description of 9
 - example 92
 - exit program 66
 - exit, overview 121
 - file format 203
 - file type created 67
 - search order 68
 - type included 69
- RESTYPE parameter
 - format 69
 - specifying resources for retrieval 9
- retrieval, preparing files for 13
- return code 310, error 198
- return codes, non-zero 113

S

- sample code
 - ACIF application 95
 - ACIF output for indexed input file 210

sample code (*continued*)

- AFP document output formats 209
- AIX and Windows parameter file 100
- ASCII input data, parameter file for 99
- CMS commands to invoke ACIF 25
- drawing graphics 95
- EBCDIC input data, parameter file for 99
- enhanced indexing 109
- enhanced indexing parameter file 111
- index record exit 118
- indexing input files 104
- indexing with data values 7
- input file 97
- input record exit 114
- line data application 95
- literal values in AIX 99
- locating resource libraries 94
- output record exit 119
- print file attributes 125
- processing parameters 100
- resource exit 121
- retrieving resources 92
- specifying coded fonts 92
- specifying TrueType and OpenType fonts 93
- transforming line or XML data 91
- using ACIF 91
- VM parameter file 103
- VSE JCL to invoke ACIF 26
- VSE parameter file 103
- z/OS JCL to invoke ACIF 23
- z/OS parameter file 101
- search order
 - AFP resources 19
 - by file extension for resources 20
 - user exits 124
- search paths for
 - FOCA fonts 45
 - form definitions 40
 - object container files 56
 - overlays 59
 - page definitions 63
 - page segments 65
 - system resources 68
 - TrueType and OpenType fonts 46, 75
 - user resources 74
- separator pages, removal from output 196
- sequential data sets
 - concatenating output 107
 - fixed-length 193
 - preparing for
 - archiving and retrieval 13
 - printing 11
 - viewing 10
 - provided with ACIF 22
 - transferring to AIX or Windows 194
 - variable-length 193
 - viewing document 108
- setup file
 - location of 56
 - name 36

- shell commands
 - concatenation of 107
 - converting EBCDIC literal values 99
- shortcut keys 215
- SOSI processing 64
- storage problem, possible cause 191
- structured fields
 - Begin Document 211
 - Begin Document Index 206
 - Begin Named Group 209, 211
 - Begin Page 212
 - Begin Resource 203
 - Begin Resource Group 202
 - Composed Text Control 212
 - End Document 212
 - End Document Index 207
 - End Named Group 209, 212
 - End Page 212
 - End Resource 203
 - End Resource Group 203
 - Index Element
 - group-level 205
 - index object file 196
 - page-level 206
 - triplets composing 206
- inserted in index object file 6
- Invoke Medium Map
 - hints for creating 195
 - inserting 55
- Map Coded Font Format 1 212
- Map Coded Font Format 2
 - coded fonts 123
 - modification 213
 - specifying 56
- No Operation 203
- Presentation Text Data Descriptor 213
- Tag Logical Element 196, 201, 207, 209, 212
- syntax rules for parameter files 29
- SYSIN JCL statement, z/OS 24
- SYSPRINT JCL statement, z/OS 24
- system resource libraries
 - example of locating 94
 - search order 68

T

- table reference characters 72
- Tag Logical Element (TLE) structured field
 - created in the output document file 209
 - examples and rules 201
 - in indexing process 207, 212
 - in named groups 191
 - index object file 196
- tags for indexing
 - definition of 6
 - example 105
 - INDEXn parameter 49, 84
 - structured fields 202
- tasks for ACIF application 97
- TLE structured field
 - created in the output document file 209

- TLE structured field (*continued*)
 - examples and rules 201
 - in indexing process 207, 212
 - in named groups 191
 - index object file 196
- TRACE parameter 71
- transaction fields 80
- transferring files
 - to AIX or Windows 194
 - to workstations, example of 109
- TRC parameter
 - format 72
 - print file attribute 126
- trigger fields 78
- TRIGGERn parameter 72, 87
- triggers for indexing
 - definition of 8
 - set by TRIGGERn parameter 72, 87
 - specifying 105
 - used with FIELDn parameter 41, 78
 - used with INDEXn parameter 49, 84
- TrueType fonts, specifying 93

U

- uconv command 117
- understanding ACIF 1
- unformatted ASCII
 - exit programs 54
 - input data to ACIF 5
 - search paths for
 - FOCA fonts 45
 - form definitions 40
 - object container files 56
 - overlays 59
 - page definitions 63
 - page segments 65
 - system resources 68
 - TrueType and OpenType fonts 46, 75
 - user resources 74
- UNIQUEBNGS parameter 74
- user exits
 - index record 53, 118
 - input record 53, 114
 - non-zero return codes 113
 - output record 58, 119
 - print file attributes provided 125
 - provided with ACIF, sample 113
 - resource, overview 121
 - search order 124
 - specified with RESEXIT 66
- user library
 - fonts 36
 - form definitions 47
 - page definitions 62
 - requesting access to 75
- user programming exits, provided with ACIF 113
- USERAPPL
 - CMS command, VM 25
 - JCL statement
 - VSE 26

USERAPPL (*continued*)
 JCL statement (*continued*)
 z/OS 23

USERLIB parameter 74

USERMASK parameter 89

USERPATH parameter 75

using ACIF

 examples of 91

 in AIX and Windows 19

 in VM 25

 in VSE 26

 in z/OS 23

 running jobs 106

 scenarios for 10

 to prepare files for

 archiving and retrieval 13

 printing 11

 viewing 10

V

values for enhanced indexing parameters 77

values for parameters 31

variable-length files 193

Viewer application of AFP Workbench

 description of 15

 fonts used to display documents 15

 group names used by 49

 ignored objects 15

 indexing for 52

 preparing files for viewing with 10

 retrieving resources for 71

 viewing document files with 108

viewing

 concatenated document files 108

 indexing data for 5

 preparing files for 10

 retrieving resources for 9

 tasks for 97

 with AFP Workbench Viewer 15

VM

 CMS commands to invoke ACIF 25

 concatenating output files 108

 DD statement for document file 25

 examples

 CMS commands to invoke ACIF 25

 drawing graphics 95

 locating resource libraries 94

 parameter file 103

 resource retrieval 92

 specifying fonts 93

 transforming line or XML data 91

 message file, defining 25

 parameter file 25

 search order, user exit load modules 124

 syntax rules for ACIF 31

 system prerequisites 17

 using ACIF 25

VSE

 examples

 drawing graphics 95

VSE (*continued*)

 examples (*continued*)

 JCL to invoke ACIF 26

 locating resource libraries 94

 parameter file 103

 resource retrieval 92

 specifying fonts 93

 transforming line or XML data 91

 JCL statement to invoke ACIF 26

 message file, defining 27

 parameters 27

 PSF limitations 17

 search order, user exit load modules 124

 syntax rules for ACIF 31

 system prerequisites 17

 using ACIF in 26

W

Windows

 acif command 30

 concatenating output files 108

 converting literal values 99

 examples

 drawing graphics 95

 locating resource libraries 94

 parameter file 100

 resource retrieval 92

 specifying coded fonts 93

 specifying TrueType and OpenType fonts 94

 transforming line or XML data 91

 files provided with ACIF 22

 input record exits 116

 invoking ACIF automatically 21

 line2afp 21

 literal values for

 ASCII data 99

 EBCDIC data 99

 mounting directories on workstations 109

 NLS messages 22

 pdpr 21

 sample user exits 113

 search order for

 resources 20

 user exit load modules 124

 shell commands 99

 syntax rules for ACIF 30

 system prerequisites 17

 transferring files to 194

 using ACIF in 19

Workbench, Viewer application

 description of 15

 fonts used to display documents 15

 group names used by 49

 ignored objects 15

 indexing for 52

 preparing files for viewing with 10

 retrieving resources for 71

 viewing document files with 108

workstations

 mounting directories on 109

workstations (*continued*)
transferring document files to 109

X

XML data 4

Z

z/OS

- concatenating output files 108
- DD statement for document file 23
- enhanced indexing 77
- examples
 - drawing graphics 95
 - enhanced indexing parameter file 111
 - JCL to invoke ACIF 23
 - locating resource libraries 94
 - parameter file 101
 - resource retrieval 92
 - specifying coded fonts 93
 - specifying TrueType and OpenType fonts 94
 - transforming line or XML data 91
- message file, defining 24
- parameter file
 - defining 23
 - for enhanced indexing 111
 - JCL for 101
- partitioned data set, definition of 9
- search order, user exit load modules 124
- syntax rules for ACIF 31
- system prerequisites 17
- using ACIF in 23

Readers' Comments — We'd Like to Hear from You

Print Services Facility
AFP Conversion and Indexing Facility
User's Guide

Publication No. S550-0436-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name

Address

Company or Organization

Phone No.



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



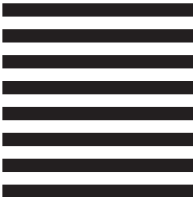
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
IBM Printing Systems Division
Department H7FE Building 004N
Boulder, CO 80301-9817



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5639-N49
5655-M32
5684-141
5686-040
5765-F68

Printed in USA

S550-0436-00



Spine information:



Print Services Facility

ACIF User's Guide