EDMSuite OnDemand

Indexing Reference

Version 2.2

S544-5489-05

EDMSuite OnDemand

Indexing Reference

Version 2.2

S544-5489-05

Sixth Edition (June 1999)

This edition of *IBM EDMSuite OnDemand: Indexing Reference* applies to IBM EDMSuite OnDemand, Version 2 Release 2 and to all subsequent releases of this product until otherwise indicated in new releases or technical newsletters.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM authorized Dealer, your IBM Marketing Representative, or your IBM Software Solutions Representative.

IBM Software Solutions welcomes your comments. For your convenience, a form for reader's comments is provided at the back of this publication. You may send your comments by fax to 1-303-924-7522, by e-mail to Ondemand@us.ibm.com, or mail your comments to:

INFORMATION DEVELOPMENT
IBM SOFTWARE SOLUTIONS
DEPARTMENT MYGA BUILDING 001L
PO BOX 1900
BOULDER CO 80301-9191

Note: Visit our home page at http://www.software.ibm.com/data/edmsuite.

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.

Contents

Figures

Notices

References in this publication to products or services of IBM do not suggest or imply that IBM will make them available in all countries where IBM does business or that only products or services of IBM may be used. Noninfringing equivalents may be substituted, but the user must verify that such substitutes, unless expressly designated by IBM, work correctly. No license, expressed or implied, to patents or copyrights of IBM is granted by furnishing this document. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information, which has been exchanged, should contact: SWS General Legal Counsel, IBM Corporation, Department TL3 Building 062, P.O. Box 12195, Research Triangle Park, NC 27709–2195 USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks and Service Marks

The following terms, used in this publication and other books in the OnDemand library, are trademarks, registered trademarks, or service marks of the IBM Corporation in the United States and other countries:

ADSTAR
Advanced Function Printing
AFP
AIX
AIX/6000
AIXwindows
BookManager
BookMaster
CICS
CICS/ESA
DATABASE 2
DB2
DB2 Universal Database

eNetwork

IBM

Infoprint

IPDS

MVS

MVS/XA

MVS/ESA

OS/2

OS/390

Print Services Facility

RS/6000

S/370

S/390

SP2

SP

System/370

System/390

Ultrastar

VisualInfo

WIN-OS2

The following terms, used in this publication and other books in the OnDemand library, are trademarks of other companies as listed:

Acrobat, Acrobat Exchange, Adobe, Adobe Type Manager, ATM, Distiller, and PostScript are trademarks or registered trademarks of Adobe Systems Incorporated

Cygnet is a registered trademark of Cygnet Systems, Inc.

Ethernet is a trademark of Xerox Corporation

Exabyte is a trademark of Exabyte Corporation

FaxBox is a trademark of DCE Corporation

Hewlett-Packard, HP-UX, HP VUE, and PCL are trademarks of Hewlett-Packard Company

Intel, MMX, and Pentium are trademarks or registered trademarks of Intel Corporation

Lotus is a trademark of Lotus Development Corporation

Microsoft, TrueType, Windows, Windows 95, and Windows NT are registered trademarks of Microsoft Corporation

Motif is a trademark of Open Software Foundation, Inc.

Netscape and Netscape Navigator are trademarks of Netscape Communications Corporation

NFS is a trademark of SUN Microsystems Incorporated

Novell is a trademark of Novell Inc.

PC/TCP is a trademark of FTP Software, Inc.

Post-it is a registered trademark of 3M

Solaris is a trademark of Sun Microsystems, Inc.

Tivoli is a trademark of Tivoli Systems, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited

WinFax Lite is a trademark of Delrina Corporation

X Windows is a trademark of the Massachusetts Institute of Technology

Portions of the OnDemand Windows client program contain licensed software from Adobe Systems Incorporated, © Adobe Systems Incorporated 1987-1997. All rights reserved.

Portions of the OnDemand Windows client program contain licensed software from Pixel Translations Incorporated, © Pixel Translations Incorporated 1990, 1997. All rights reserved.

About this publication

This book contains information about indexing methods, preparing index data, and using tools to index data that you plan to store in and retrieve from EDMSuite OnDemand Version 2.2 (OnDemand).

Note: PSF for AIX, PSF/MVS, and PSF for OS/390 are commonly referred to as PSF throughout this book. MVS Download and Download for OS/390 are commonly referred to as Download throughout this book.

Who should use this publication

This book is of primary interest to OnDemand application administrators and other people responsible for preparing data to be stored in OnDemand.

How this publication is organized

This book is organized in the following parts. Each part contains information about one of the tools provided with OnDemand that you can use to index input files:

- "Part 1. OnDemand ACIF Reference" on page 1 provides information about using the enhanced AFP Conversion and Indexing Facility (ACIF) to index AFP and line data files
- "Part 2. OnDemand Generic Indexer Reference" on page 201 provides information about using the OnDemand generic indexer to index files
- "Part 3. OnDemand PDF Indexer Reference" on page 211 provides information about using the OnDemand PDF indexer to index PDF files

Our use of typefaces

Throughout this book, words and phrases appear in **Bold**, *Italic*, and other fonts. The following explains our convention when using these fonts.

Bold	Used for paragraphs that call attention to especially relevant information about a topic or command.
Italic	Used to emphasize concepts and terms.
`Monospace`	Indicates output of commands and programs in examples. Also used for information you are instructed to type.

UPPERCASE Indicates parameter or command names and sometimes file and directory names.

Product support

The IBM Support Center maintains current information about OnDemand, including program temporary fixes (PTFs).

Before you install OnDemand, contact the IBM Support Center or your IBM software service representative to obtain the latest maintenance level of OnDemand.

If you encounter problems or errors running any of the OnDemand programs, you can call the IBM Support Center to obtain software problem and defect support. The phone number for the IBM Support Center is 1-800-237-5511. The OnDemand program number is 5622–622 for AIX, 5765–D60 for HP-UX, 5765–E20 for Solaris, and 5639–E12 for Windows NT. The OnDemand component ID is 5622-66200.

OnDemand documentation

The following publications contain information about OnDemand Version 2.2:

Introduction and Planning Guide, G544-5281

Installation and Configuration Guide for UNIX Servers, G544-5598

Installation and Configuration Guide for Windows NT Servers, G544-5526

Administrator's Reference, S544-5293

Indexing Reference, S544-5489

Getting Started with the Administrative Client, S544-5463

User's Guide, SC26-9810

OS/2 Client Customization Guide, S544-5465

OLE Control and Win32 Client Customization Guide and Reference, S544-5466

Related documentation

The following publications contain information about ADSM Version 3, DB2 Universal Database Version 5, and PSF:

ADSM Messages, GC35-0271

ADSM for AIX Version 3 Quick Start, GC35-0273

ADSM for AIX Version 3 Administrator's Guide, GC35-0274

ADSM for AIX Version 3 Administrator's Reference, GC35-0275

ADSM for Windows NT Version 3 Administrator's Guide, GC35-0292

ADSM for Windows NT Version 3 Administrator's Reference, GC35-0293

ADSM for Windows NT Version 3 Quick Start, GC35-0295

ADSM for HP-UX Version 3 Administrator's Reference, GC35-0321

ADSM for HP-UX Version 3 Administrator's Guide, GC35-0320

ADSM for HP-UX Version 3 Quick Start, GC35-0322

ADSM for Sun Solaris Version 3 Administrator's Guide, GC35-0328

ADSM for Sun Solaris Version 3 Administrator's Reference, GC35-0329

ADSM for Sun Solaris Version 3 Quick Start, GC35-0330

DB2 Universal Database Extended Enterprise Edition for UNIX Version 5 Quick Beginnings, S99H-8314

DB2 Universal Database Version 5 Administration Getting Started, S10J-8147

DB2 Universal Database for UNIX Version 5 Quick Beginnings, S10J-8148

DB2 Universal Database for Windows NT Version 5 Quick Beginnings, S10J-8149

DB2 Universal Database Version 5 Administration Guide, S10J-8157

DB2 Universal Database Version 5 Command Reference, S10J-8166

DB2 Universal Database Version 5 Message Reference, S10J-8168

PSF for AIX: Print Submission, S544-3878

PSF for AIX: Print Administration, S544-3817

PSF for MVS: MVS Download Guide, G544-5294

PSF for OS/390: Download for OS/390, S544-5624

Related IBM EDMSuite Products

OnDemand is one of the web-enabled products included in the IBM Enterprise Document Management Suite (EDMSuite), which is a portfolio of IBM software that includes imaging, computer output to laser disk (COLD) document management, and workflow. In addition to OnDemand, EDMSuite contains four more products:

- ImagePlus VisualInfo
- ContentConnect
- MQSeries Workflow
- Domino Doc

For more information on EDMSuite, visit the EDMSuite homepage at http://www.software.ibm.com/data/edmsuite.

ImagePlus VisualInfo

ImagePlus VisualInfo is a distributed and scalable client/server solution that enables the management and storage of a broad array of document types, such as document images, graphics, spreadsheets, text, audio, and video. ImagePlus consists of scalable, multiplatform offerings for organizations of all sizes. ImagePlus implementations range from simple store-and-retrieve applications to image and workflow-enabling, very high-volume, transaction-oriented business processes. Some ImagePlus capabilities allow you to:

- Develop a robust library to store documents
- Handle multiple file types in addition to images, such as word processing documents, audio and video clips, and spreadsheets
- Manage and migrate documents across magnetic, optical, and tape storage automatically

ContentConnect

ContentConnect is a client toolkit based on Java that provides single-query access to multiple repositories from Web browsers, Lotus Notes clients, and stand-alone clients. ContentConnect provides a search engine that supplies reliable and immediate access to advanced document imaging, workflow, COLD archiving, and collaborative document management components. ContentConnect presents its search results in an organized, seamless, easy-to-read manner. ContentConnect not only connects document archives

and processes to other components of EDMSuite, but also connects its users to the document management world outside EDMSuite.

MQSeries Workflow (formerly FlowMark)

MQSeries Workflow is a workflow engine designed for the client/server environment. MQSeries Workflow is dedicated to managing the flow of work, allowing companies to integrate the applications required to meet the needs of their business processes. MQSeries Workflow allows you to:

- Separate application logic from business process rules
- Tie legacy and client/server applications to business process steps
- Monitor processes to show where a specific piece of work is in the overall process
- Record live production process data for analysis by management
- Model, animate, and simulate your business processes rapidly
- Enable processes across the Internet and Intranets

Domino Doc

Domino.doc is a Web-based document and content management solution that allows organizations to capture, file, retrieve, and distribute content across the Internet using desktop applications, Web browsers, or any Lotus Notes client. Domino.doc allows you to:

- Create document profiles that enforce the capture of key document search terms, so documents can be searched and retrieved easily
- Create a common library to provide a knowledge base of documents for your enterprise
- Use version control to provide a history of previous changes
- Use check-in and check-out controls to ensure data integrity

Part 1. OnDemand ACIF Reference

This part of the book provides information about the OnDemand AFP Conversion and Indexing Facility (ACIF).

Chapter 1. Introducing ACIF

Overview

This section of the book provides an overview of ACIF, information about important indexing concepts, and briefly describes how OnDemand uses the index data ACIF extracts from reports.

You can use ACIF to index many different types of AFP and line data reports. The *Introduction and Planning Guide* provides information about the types of input data that can be indexed with ACIF, requirements you need to consider when indexing files, and general information about converting line data reports to AFP.

Before you index a report with ACIF, you need to create a set of processing parameters that describe characteristics of the report and identify where ACIF can locate index data. Collecting the information needed for the processing parameters requires several steps:

1. Examine the input data to determine usage and indexing requirements.
2. Decide whether the (line data) input file will be converted to AFP data.
   - If you plan to enhance the appearance of line data with fonts and bar codes or you need to compose a line data input file into pages, you must convert the source data to AFP.
   - If you need to generate page-level index information for a line data input file, you must convert the input data to AFP.
3. Examine the input data to determine resource requirements. Determine the fonts and form and page definitions needed to view and print the data.
4. Create parameters for indexing.
5. Create parameters for converting line data input files.
6. Create parameters for collecting resources for viewing and printing AFP data.

You can run ACIF on OnDemand servers, MVS systems, and OS/390 systems. On an OnDemand server, you can run ACIF from the command prompt or you can use the OnDemand data indexing and loading programs to process reports. The information provided in this book assumes you will use the OnDemand data indexing and loading programs to process reports with ACIF.

About ACIF

ACIF is a batch utility that provides three major functions:

- Sophisticated indexing functions.

  ACIF can logically segment report files into individual items, such as statements, policies, and bills. You can define up to 32 index fields for each item in a report.

- Conversion of print data streams.

  ACIF processes the output print data streams of application programs, for example, line data reports and unformatted ASCII. The converted output can be printed, viewed, and archived on any system supported by OnDemand.

- Collection of AFP resources.

  ACIF can determine the resources necessary to print, view, and archive the print data stream and collect the resources from PSF and user libraries. Resources allow end-users to view the report file as it appeared in the original printed version, regardless of when or where the report was created.

Indexing concepts

Indexing parameters include information that allow ACIF to identify key items in the print data stream, *tag* these items, and create *index elements* pointing to the tagged items. ACIF uses the tag and index data for efficient, structured search and retrieval. You specify the index information that allows ACIF to segment the data stream into individual items called *groups*. A group is a collection of one or more pages. You define the bounds of the collection, for example, a bank statement, insurance policy, phone bill, or other logical segment of a report file. A group can also represent a specific number of pages in a report. For example, you might decide to segment a 10,000 page report into groups of 100 pages. ACIF creates indexes for each group. Groups are determined when the value of an index changes (for example, account number) or when the maximum number of pages for a group is reached.

A tag is made up of an *attribute name* (for example, Customer Name) and an *attribute value* (for example, Earl Hawkins). Tags include pointers that tell ACIF where to locate the attribute information in the data stream. For example, the tag *Account Number* with the pointer *1,21,16* means ACIF can expect to find Account Number values starting in column 21 of specific input records (later we will explain how ACIF locates the specific input records). ACIF collects 16 bytes of information starting at column 21 and adds it to a list of attribute values found in the input. ACIF creates an *index object* file when you index report files. The index object file includes *index elements* that

contain the offset and length of a group. ACIF calculates an index element for every group found in the input file. ACIF writes the attribute values extracted from the input file to the index object file and if the input file is converted to AFP, to the (converted) output file.

Indexing parameters

Processing parameters can contain index and conversion parameters, options, and values. For most reports, ACIF requires three parameters to generate index data:

- TRIGGER

  ACIF uses triggers to determine where to locate data. A trigger instructs ACIF to look for certain information in a specific location in the report file. When ACIF finds a record in the data stream that contains the information specified in the trigger, it can begin to look for index information.

  – ACIF compares data in the report file with the set of characters specified in a trigger, byte for byte.

  – A maximum of eight triggers can be specified.

  – All fixed group triggers must match before ACIF can generate index information. However, floating triggers can occur anywhere in the data stream. That is, index data based on a floating trigger can be collected from any record in the report file.

- FIELD

  The field parameter identifies the location, offset, and length of the data that ACIF uses to create index values.

  – Field definitions are based on TRIGGER1 by default, but can be based on any of eight TRIGGER parameters.

  – A maximum of 32 fields can be defined.

  – A field can also specify all or part of the actual index value stored in the database.

- INDEX

  The index parameter is where you specify the attribute name, identify the field or fields on which the index is based, and specify the type of index that ACIF generates. For the group-level indexes that OnDemand stores in the database, we strongly encourage you to name the attributes the same as the application group database field names.

  – ACIF can create indexes for a page, group of pages, and the first and last sorted values on a page or group of pages. OnDemand stores group-level index values in the database. Users can search for items using group-level indexes. Page-level indexes are stored with the AFP document (for example, a statement). After retrieving the AFP document, users can navigate it using the page-level indexes.

- You can concatenate field parameters to form an index.
- A maximum of 32 index parameters can be specified.

ACIF creates a new group and extracts new index values when one or more of the fixed group index values change or the GROUPMAXPAGES value is reached.

Figure 1 depicts a portion of a page from a sample input file.

```
 ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9
01                                                                    Page 0001
1
2                                     Jon Smyth
3                                     123 Ubik Way
4                                     Meadow Bridge WV 99999-9999
5
6                                     Statement Date: 08/01/1995
7                                     Account Number: 3727-1644-0081-0099
8
9                                     Balance: $1,096.54
```

Figure 1. Indexing a Report

The following indexing parameters could be used to generate index data for the report depicted in Figure 1. The TRIGGER definitions tell ACIF how to identify the beginning of a group in the input. ACIF requires two TRIGGER definitions to identify the beginning of a group (statement) in the sample file. For example:

- TRIGGER1 looks for a 1 in the first byte of each input record.
- TRIGGER2 looks for the string `Page 0001` in column 72 of the same record.

Together, the triggers uniquely identify the start of a statement in the report.

The FIELD definitions determine the location of index values in a statement. Fields are based on the location of trigger records. For example:

- FIELD1 identifies customer name index values, beginning in column 40 of the second record following the TRIGGER1 record.
- FIELD2 identifies statement date index values, beginning in column 56 of the sixth record following the TRIGGER1 record.
- FIELD3 identifies account number index values, beginning in column 56 of the seventh record following the TRIGGER1 record.

An INDEX definition identifies the attribute name of the index field. Indexes are based on one or more field definitions. For example:

- INDEX1 identifies the attribute name `custnam`, for values extracted using FIELD1.

- INDEX2 identifies the attribute name `sdate`, for values extracted using FIELD2.
- INDEX3 identifies the attribute name `acctnum`, for values extracted using FIELD3.

Converting data to AFP

You can convert line data or mixed-mode data into AFP data, which is an architected, device-independent data stream used for interchanging data between different platforms.

ACIF can process the following input data streams to create an AFP file:

- AFP data
- MO:DCA-P data
- Line data
- Mixed-mode data
- Unformatted ASCII

AFP data

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

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

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

Mixed Object Document Content Architecture Data

ACIF supports MO:DCA-P data as a valid input data stream, with the following restrictions:

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

ACIF does not transform the MO:DCA-P data it processes, but may change certain structured fields. For example, ACIF converts MCF1 structured fields

in the input to MCF2 structured fields in the output. If the MO:DCA-P input data stream contains multiple Begin Document (BDT) and End Document (EDT) structured fields, the output contains only one BDT/EDT structured field pair. The output page always remains the same; the output MO:DCA-P data may not contain the same structured fields or the structured fields may not appear in the same order.

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

Line data

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

Unformatted ASCII data

Unformatted ASCII data is data that is generated in the workstation environment and has not been formatted for printing. Unformatted ASCII data is formatted by ACIF using a page definition resource. Unformatted ASCII data is contrasted with the type of ASCII data that contains control characters (or escape sequences) for a line printer, such as an IBM Proprinter or IBM Quietwriter.

AFP resources

The ACIF indexing parameters that you use to process reports can contain information about resources. ACIF uses resources to reproduce a version of the input that appears the same as the original printed version. During processing, ACIF determines the list of required AFP resources needed to view or print the data and can retrieve these resources from specified directories (libraries, in MVS and OS/390). The directories must be resident on the system where ACIF is running (or you must provide access to them). ACIF collects the resources and places them in a resource file. OnDemand loads the resource file at the same time it loads the indexed report files.

When you store a report in OnDemand, you can archive the resources (for example, page segments) in the form which they existed when the report was created. By archiving the original resources, you can reproduce the report with fidelity later, even if the resources have changed since that time. Table 1 lists typical values for the RESTYPE parameter.

Table 1. Collecting Resources

restype	**Meaning**	**Why**
NONE	Do not collect resources.	Indexing line data without conversion or AFP data that does not reference external resources.
FDEF, PSEG, OVLY, BCOCA, GOCA, IOCA	Collect all except fonts.	Viewing items.
ALL, with user-defined resource exit	User-defined.	Include or exclude specific resources.

The resources that ACIF collects is based on the value of the RESTYPE parameter. When ACIF processes a 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 a page segment.
- Creates a resource file.

  ACIF collects resources in an AFP resource group and stores the resource group in a resource file. Depending on the options that you specify on the RESTYPE parameter, the resource file contains all the resources necessary to view or print the report with fidelity.
- Calls the specified resource exit for each resource it retrieves.

  You can specify the name of a resource exit on the RESEXIT parameter so that ACIF filters out any resources that you do not want included in the resource file.
- Includes the name of the output document in the resource file and the name of the resource file in the output document. This provides a method of correlating resources files with the appropriate output document.

How OnDemand uses index information

Every item stored in OnDemand is indexed with one or more group-level indexes. Groups are determined when the value of an index changes (for example, account number) or when the maximum number of pages for a group is reached. When you store a report in OnDemand, the arsload command invokes the arsacif command to process the index parameters and create index data. The arsload command then loads the index data in the database, storing the group-level attribute values that ACIF extracted from the data into their corresponding database fields. Figure 2 shows an overview of the index creation and loading process.

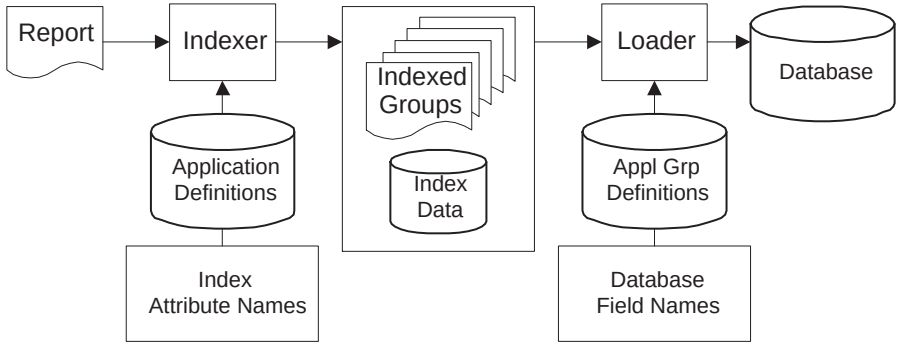


Figure 2. Index Creation and Loading

Note: Only group-level indexes can be stored in the database. Although ACIF can generate page-level indexes for AFP data (and line data that you convert to AFP or line data that you store in large objects), the page-level indexes are not stored in the database. This means that users cannot use page-level indexes to search for reports stored in OnDemand. Page-level indexes are stored with the document. After retrieving a document, the user can use the page-level indexes to navigate pages of the document.

You create an application for each report that you want to archive in OnDemand. When you create an application, you define the indexing parameters that ACIF uses to process the report and create the index data that is loaded into the database. The parameters contain indexing specifications, determine whether ACIF converts line data reports to AFP files, and indicate the types of resources that ACIF collects. For example, an INDEX parameter includes an attribute name and identifies the FIELD parameter that ACIF uses to locate the attribute value in the input data. When you create an application, you must assign the application to an application group. The attribute name you specify on an INDEX parameter should be the same as the name of the corresponding application group database field.

You define database fields when you create an application group. OnDemand creates a column in the application group table for each database field that you define. When you index a report, you create index data that contains index field names and index values extracted from the report. OnDemand stores the index data into the database fields.

To search for reports stored in OnDemand, the user opens a folder. The search fields that appear when the user opens the folder are mapped to database fields in an application group (which in turn, represent ACIF attribute names). The end-user constructs a query by entering values in one or more search fields. OnDemand searches the database for items that contain index values (ACIF attribute values) that match the search values entered by the end-user. Each item contains group-level index information. OnDemand lists the items that match the query. When the user selects an item for viewing, the OnDemand client program retrieves the selected item from cache storage or archive media. If the item is an AFP document or a line data document stored in large objects, the user can navigate pages of the item using the page-level index information (if page-level index information was generated when the report was processed by ACIF).

ACIF parameters for EBCDIC data

Reports to be stored in OnDemand are typically created on an MVS or OS/390 system in EBCDIC format. With EBCDIC data, certain index values must be coded in hexadecimal to correctly process the data. This topic describes the index parameters, options, and data values that can be used to process a report that contains EBCDIC data.

Accessing reports

Reports generated on an MVS or OS/390 system are typically transmitted to an OnDemand server using Download. The report must be transmitted to the server as a binary file to retain the data as EBCDIC. The input data contains variable length records. You must specify FILEFORMAT=RECORD to correctly process the file.

Creating indexing parameters

If you index reports on an OnDemand server, you can process a sample input data file with the graphical indexer, create a file that contains indexing parameters and import the file into the application, or enter the indexing parameters on the Indexer Information page. If you index reports on an MVS or OS/390 system, define the indexing parameters in an indexing data set on the system that is accessible to the ACIF program.

Literal values that you specify in the FIELD, INDEX, and TRIGGER parameters must be expressed as hexadecimal strings. For example, the string *Customer Name* is represented as follows:

```
index6=X'C39AA2A39694859940D5819485',field6 /* Customer Name */
```

Figure 3. Index Parameter for EBCDIC Data

Specifying indexing parameters

Let's assume that our sample report uses the following data values for indexing attributes:

- Account Number (acctnum)
- Customer Name (custnam)
- Statement Date (sdate)

To locate these indexing attributes in the sample report, two TRIGGER parameters are required. The first trigger tells ACIF to examine the first byte of every input record until it finds the occurrence of an ANSI skip-to-channel one carriage control character. After locating a record containing a F1 in the first byte, ACIF uses the second trigger to look for the string D7C1C7C540F0F0F0F1 (PAGE 0001) starting in column 72 of the same input record. When this condition is found, a new statement exists, and the record containing an F1 in the first byte is considered the anchor record. ACIF uses the anchor record to locate index values. The trigger specifications are expressed in hexadecimal, as follows:

```
trigger1=*,1,X'F1'                  /* Skip to Channel 1 */
trigger2=0,72,X'D7C1C7C540F0F0F0F1' /* PAGE 0001         */
```

Figure 4. Trigger Parameters for EBCDIC Data

ACIF uses both trigger values to locate the place in the report file to begin searching for the data supplied in the INDEX parameters.

To create the indexing tag for the customer name attribute, define the hexadecimal string 839AA2A3958194 (custnam) as the indexing attribute. The index field name is the same as the application group database field name. Locate customer name index values in the second record following the anchor record, starting at byte 40 and extending for 20 bytes. The FIELD and INDEX specifications are expressed as follows:

```
field1=2,40,20                  /* custnam field           */
index1=X'839AA2A3958194',field1 /* index/db field = custnam */
```

Figure 5. Field and Index Parameters for EBCDIC Data

To create the indexing tag for the statement date attribute, define the hexadecimal string A28481A385 (sdate) as the indexing attribute. The index field name is the same as the application group database field name. Locate statement date index values in the sixth record following the anchor record, starting at byte 56 and extending for 10 bytes. The FIELD and INDEX specifications are expressed as follows:

```
field2=6,56,10              /* sdate field            */
index2=X'A28481A385',field2 /* index/db field = sdate */
```

Figure 6. Field and Index Parameters for EBCDIC Data

To create the indexing tag for the account number attribute, define the hexadecimal string 818383A395A494 (acctnum) as the indexing attribute. The index field name is the same as the application group database field name. Locate account number index values in the seventh record following the anchor record, starting at byte 56 and extending for 19 bytes. The FIELD and INDEX specifications are expressed as follows:

```
field3=7,56,19                  /* acctnum field           */
index3=X'818383A395A494',field3 /* index/db field = acctnum */
```

Figure 7. Field and Index Parameters for EBCDIC Data

After indexing the report, OnDemand stores index values in the database for each of the three indexing attributes for each statement in the data stream. Using an OnDemand client program, users can locate a specific customer statement using a date, and optionally, any combination of customer name and customer number.

Chapter 2. Using ACIF

Example one

About the report

This example describes how to create indexing information for a sample loan report. A loan report typically contains hundreds of pages of line data. Each page in the report follows the same basic format: a report page header (five records) that includes the report data, a report field header (three records), and up to 58 sorted detail records. The detail records contain several fields, including the loan number. Figure 8 on page 16 shows a sample page of the loan report, as it appears when viewed using an OnDemand client program.

For faster loading and retrieval, we plan to segment the report into 100 page groups when we load it into the database. We'll create one row for each group of pages. The row contains three user-defined indexes: the report date, the beginning loan number, and the ending loan number. Figure 9 on page 17 shows the indexer parameters required for ACIF to process the loan report.

Accessing the sample report

This example provides instructions about using the OnDemand graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the OnDemand administrative client, a Windows 32-bit application. To process a sample report, you typically create or extract a subset of a complete report. The report in this example was generated on an MVS system. We transferred the report to the PC as a binary file. If you download reports from an MVS system to an OnDemand server, you can mount the filesystem (or share the directory) that contains the reports on the PC. You can then load the sample data into the graphical indexer.

It is extremely important that the sample data used to create the indexing information matches the actual data to be indexed and loaded into the database. When you load a report, OnDemand uses the indexing parameters, options, and data values stored in the application definition to index the data. If the data being loaded does not match the data that you use to generate indexing parameters with the graphical indexer, OnDemand cannot properly index the data. For example, OnDemand won't be able to locate triggers, indexes, and fields or extract the correct index values.

REPORT D94100100 PENNANT NATIONAL BANK DATE 10/01/94
BANK 001 TIME 16:03:46
FROM 10/01/94 MODE 9
TO 10/01/94 LOAN DELINQUENCY REPORT PAGE 00001

LOAN NUMBER	CUSTOMER NAME	LOAN AMOUNT	DELINQUENT 30 DAYS	DELINQUENT 60 DAYS	DELINQUENT 90 DAYS
0000010000	MCMULLIGAN, PATRICK	$10000000.00	$ 50.00	$ 50.00	$ .00
0000010001	ABBOTT, DAVID	$ 11000.00	$ 100.00	$ 200.00	$ .00
0000010002	ABBOTT, DAVID	$ 12000.00	$ 140.00	$ .00	$ .00
0000010003	ABBOTT, DAVID	$ 13000.00	$ 150.00	$ .00	$ .00
0000010005	ROBINS, STEVEN	$ 500.00	$ 50.00	$ .00	$ .00
0000010006	PALMER, ARNOLD	$ 1000.00	$ 75.00	$ 150.00	$ 225.00
0000010007	PETERS, PAUL	$ 650.00	$ 50.00	$ .00	$ .00
0000010008	ROBERTS, ABRAHAM	$ 9000.00	$ 120.00	$ .00	$ .00
0000010009	SMITH, RANDOLPH	$ 8000.00	$ 115.00	$ .00	$ .00
0000010010	KLINE, PETER	$ 8500.00	$ 110.00	$ .00	$ .00
0000010017	WILLIAMS, ALFRED	$ 10000.00	$ 50.00	$ 50.00	$ .00
0000010019	JAMES, TIMOTHY	$ 11000.00	$ 100.00	$ 200.00	$ .00
0000010022	THOMAS, JAMES	$ 12000.00	$ 140.00	$ .00	$ .00
0000010026	ROBBINS, KARL	$ 13000.00	$ 150.00	$ .00	$ .00
0000010029	MILLER, FREDERICK	$ 500.00	$ 50.00	$ .00	$ .00
0000010033	DAVIDSON, ALBERT	$ 1000.00	$ 75.00	$ 150.00	$ 225.00
0000010049	STEVENS, MARY	$ 650.00	$ 50.00	$ .00	$ .00
0000010050	MICHAELS, LOUISE	$ 9000.00	$ 120.00	$ .00	$ .00
0000010051	ABEL, CHARLIE	$ 8000.00	$ 115.00	$ .00	$ .00
0000010056	BAKER, THOMAS	$ 8500.00	$ 110.00	$ .00	$ .00
0000010101	TAYLOR, ADRIANNE	$ 13000.00	$ 150.00	$ .00	$ .00
0000010111	MILLER, ROBERT	$ 500.00	$ 50.00	$ .00	$ .00
0000010123	DAVID, NEIL	$ 1000.00	$ 75.00	$ 150.00	$ 225.00
0000010132	STEVENS, SUSAN	$ 650.00	$ 50.00	$ .00	$ .00
0000010133	MITCHELL, PAMELA	$ 9000.00	$ 120.00	$ .00	$ .00
0000010135	FRANCIS, WILLIAM	$ 8000.00	$ 115.00	$ .00	$ .00
0000010146	THOMAS, GEORGIA	$ 8500.00	$ 110.00	$ .00	$ .00
0000010152	PHILLIPS, CHARLES	$ 13000.00	$ 150.00	$ .00	$ .00
0000010158	WATKINS, DIANA	$ 500.00	$ 50.00	$ .00	$ .00
0000010171	FRANKLIN, ELIZABETH	$ 1000.00	$ 75.00	$ 150.00	$ 225.00
0000010179	TOMLIN, FRANK	$ 650.00	$ 50.00	$ .00	$ .00
0000010200	CASTLES, AARON	$ 9000.00	$ 120.00	$ .00	$ .00
0000010207	WILLOBOUGHY, LUKE	$ 8000.00	$ 115.00	$ .00	$ .00
0000010229	HOPKINS, GEORGE	$ 8500.00	$ 110.00	$ .00	$ .00
0000010251	SHEPHERD, RANDY	$ 8000.00	$ 115.00	$ .00	$ .00
0000010316	AARON, ROBERT	$ 8500.00	$ 110.00	$ .00	$ .00
0000010324	JOHNSON, JONATHON	$ 13000.00	$ 150.00	$ .00	$ .00
0000010327	SELLERS, NELSON	$ 500.00	$ 50.00	$ .00	$ .00
0000010328	ATKINS, ELWOOD	$ 1000.00	$ 75.00	$ 150.00	$ 225.00

Figure 8. Loan Report

```
/* DATA INPUT/OUTPUT CHARACTERISTICS                                    */
CC=YES                                  /* carriage controls present   */
CCTYPE=A                                /* ANSI controls in EBCDIC     */
CONVERT=NO                              /* line data in OD             */
CPGID=500                               /* code page id                */
TRC=NO                                  /* table ref chars not present */
FILEFORMAT=RECORD,133                   /* fixed length records        */

/* TRIGGER/FIELD/INDEX DEFINITIONS                                      */
TRIGGER1=*,1,X'F1',(TYPE=GROUP)                    /* 1                */
FIELD1=0,83,8,(TRIGGER=1,BASE=0)                   /* report date field */
FIELD2=*,*,10,(OFFSET=(3:12),MASK='##########',ORDER=BYROW)  /* loan number field */
INDEX1=X'939681846D8481A385',FIELD1,(TYPE=GROUP,BREAK=YES)   /* report date index */
INDEX2=X'D396819540D5A494828599',FIELD2,(TYPE=GROUPRANGE)    /* loan number index */

/* INDEXING INFORMATION                                                 */
DCFPAGENAMES=NO                         /* page names in input data    */
UNIQUEBNGS=YES                          /* unique group names          */
GROUPMAXPAGES=100                       /* 100 page groups             */
IMAGEOUT=ASIS                           /* leave images alone          */
INDEXOBJ=GROUP                          /* group-level indexes         */
INDEXSTARTBY=1                          /* must find index by page 1   */
INSERTIMM=NO                            /* do not add IMMs to groups   */

/* RESOURCE INFORMATION                                                 */
RESTYPE=NONE                            /* do not collect resources    */
```

Figure 9. ACIF Parameters

Key concepts

Group Trigger. Group triggers identify the beginning of a group. You must define at least one group trigger. Trigger1 must be a group trigger.

Transaction Field. A field used to index a report that contains one or more columns of sorted data and it is not practical to store every value in the database.

Field Offset. The location of the field from the beginning of the record.

Field Mask. A pattern of symbols that ACIF matches with data located in the field columns.

Field Order. Identifies row-oriented data or column-oriented data.

Group Index. Indexes generated once for each group. All data stored in OnDemand must be indexed by group (even if a group contains only one page).

Grouprange Index. Indexes generated for the starting and ending sorted values in each group.

Index Break. Indexes that determine when ACIF closes the current group and begins a new group. A group index determines when ACIF breaks groups. However, a group index based on a floating trigger cannot be used to control group breaks. A grouprange index cannot be used to control group breaks.

Defining the application — part 1

An application identifies the type of data that is stored in OnDemand, the method used to index the data, and other information that OnDemand uses to load and view reports. This topic provides information about the application we created to process the sample loan report.

General

The General page is where we name the application and assign the application to an application group.

We assigned the application to the application group where we plan to store the loan reports. The application group contains database field definitions for the report date and beginning and ending loan numbers.

View Information

The View Information page is where we specify information needed by OnDemand client programs to display the loan report. This information is also used by the indexing program.

Because the loan report will be stored in OnDemand as line data, we set the Data Type to Line. Other important settings on this page include:

- Code Page. We set the code page to 500. This is the code page of the data as stored in OnDemand and viewed using programs such as the graphical indexer.
- RECFM. Records in the input data are fixed length, 133 characters in length.
- CC. The input data contains carriage control characters in column one of the data.
- CC Type. The input data contains ANSI carriage control characters coded in EBCDIC.

Indexer Information

The Indexer Information page is where we need to do the bulk of our work.

First, we change the Indexer to ACIF. Notice the ACIF indexing parameters, options, and data values that the administrative client automatically set, based on our choice of indexer and the settings we chose on the View Information page:

- `CONVERT=NO`. Since we plan to store line data in OnDemand, we do not want ACIF to convert the line data to AFP.
- `CPGID=500`. The code page of the data as stored in OnDemand.
- `FILEFORMAT=RECORD,133`. The fileformat parameter contains information that identifies the format and length of the input records.

The remaining parameters are standard ACIF parameters with default values for processing line data.

We need to define additional ACIF parameters, including those that determine the index data extracted from the report and loaded into OnDemand. We will process sample report data with the OnDemand graphical indexer and define the parameters.

Opening the sample report

Make sure that the Add an Application window is turned to the Indexer Information page. In the Parameter Source area, select Sample Data. Click Modify. OnDemand displays the Open dialog box. Select the name of the file that contains the sample data. Click Open. OnDemand loads the file into the report window.

The name of the input file is displayed at the top of the window along with a warning that the data must match the data being loaded.

Note: When you load a report, OnDemand uses the indexing parameters, options, and data values stored in the application definition to index the data. If the data being loaded does not match the data that you use to generate indexing parameters with the graphical indexer, OnDemand cannot properly index the data. For example, OnDemand will not be able to locate triggers, indexes, and fields or extract correct index values.

Defining triggers

When processing sample data with the graphical indexer, you typically define triggers first, then fields, and finally, indexes.

ACIF uses one or more triggers to determine where to begin to locate index values. For the loan report, we need to define a trigger that instructs ACIF to examine the first byte of every input record for the presence of an EBCDIC skip-to-channel one carriage control character (X'F1'). This is the only trigger we need to define.

Since the graphical indexer displays the report as it is viewed in OnDemand, we cannot see the carriage control characters in column one of the data. To define the trigger, select any column in the first record. When you select a column, the graphical indexer highlights the data. Next, click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger1) determines the name of the trigger parameter. Trigger1 must always be defined and establishes a starting point where other group triggers and non-floating fields can be found. The Records to Search area determines the records ACIF searches to locate the trigger. For the loan report, we want ACIF to search every record. The Columns to Search area determines the columns of the trigger record ACIF searches. We set the Columns to Search to Carriage Control so that ACIF searches column one of each record. When we do this, the graphical indexer displays the trigger value (X'F1') that ACIF searches for.

Click OK to add the trigger and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the trigger. Note the trigger parameter with the options and data values we specified in the Add a Trigger dialog box.

Close the Display ACIF Parameters dialog box.

Defining fields

ACIF uses field definitions to determine where to locate index values. For the sample loan report, we must define two fields. The first field identifies where ACIF locates the date. The second field identifies where ACIF locates the loan number.

Select a field by clicking on the area in the report that contains the field data. In the sample loan report, we select the date value displayed in column 83 of the first record on the page. The date is displayed as 10/01/94 (mm/dd/yy). After selecting the field, the graphical indexer highlights the value. Next, with the pointer on the field, click the right mouse button once and select Field from the list. OnDemand displays the Add a Field dialog box.

The Identifier (Field1) determines the name of the field parameter. The Trigger (Trigger1) determines the name of the trigger parameter that ACIF uses to locate the field. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger. For the loan report, the field record and the trigger record are the same. The Columns to Search area

determines the column number (83) where ACIF locates the beginning of the field. The Size area determines the length (8) of the field. The Reference String area lists the field value we selected.

We don't need to make any changes to the information extracted by the graphical indexer. Click OK to add the field and return to the report window.

The second field we need to define contains the loan number. Because of the way we want OnDemand to segment and load the data and the way users will search for reports, we need to define a `transaction field`. A transaction field allows OnDemand to index a group of pages using the first index value on the first page and the last index value on the last page. This is an excellent way to segment large reports, resulting in good data loading and retrieval performance. Select the field by clicking on the area in the report that contains the field data. We selected the loan number displayed in column three of the ninth record on the page. The loan number is displayed as `0000010000`. Next, with the pointer on the field, click the right mouse button once and select Transaction Field from the list. OnDemand displays the Add a Field dialog box.

Verify the options and data values for the field. The Identifier is Field2. The Order (By Row) identifies how field data is organized. The Mask determines the pattern of symbols that ACIF matches with data located in the field. The number symbol (#) matches any numeric. The string of ten number symbols matches a ten-character numeric field. The Size area contains the field length (10). The Column Offsets area determines the location of the field value from the beginning of the record. A transaction field can identify up to eight data values, each located with a Start and End value. For the loan report, the field identifies one value, starting in column three and ending in column twelve.

We don't need to make any changes to the information extracted by the graphical indexer. Click OK to add the field and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the fields. Note the field parameters with the options and data values we specified in the Add a Field / Transaction Field dialog boxes.

Close the Display ACIF Parameters dialog box.

Defining indexes

Our next task in defining indexing parameters for the loan report is to define indexes. The index definitions determine the values stored in the database and

the type of index. For the loan report, we must define two indexes. The first index contains a date value for a group of pages. The second index contains the beginning and ending loan number values for a group of pages.

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index1) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, the application group database field name, `report_date`. When we load data into the application group, the date index values will be stored in the `report_date` application group database field. The Type of Index determines the type of index generated by ACIF. Select Group. (To store data in OnDemand, you must define at least one group index.) We set Break to Yes because a group index must always control the break. (Even though in this example, the group index doesn't really control the break, the GROUPMAXPAGES parameter does. More on that later.) The Fields area lists the field parameters that have been defined (in the Fields list) for the report. We need to identify the field parameter that ACIF uses to locate the index. For the date index values, that is Field1. Select Field1 in the Fields list and click Add. OnDemand moves Field1 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

The second index contains the beginning and ending loan number values. The Identifier is Index2. Since we can't map the loan number values directly to a database field, we must enter our own index name in the Attribute field. Later, on the Load Information page, we'll map the index to application group database fields. Type `Loan Number` in the Attribute field. Because we want ACIF to extract the beginning and ending loan numbers for a group of pages, we need to select an index Type of GroupRange. Since a grouprange trigger can never break a group, Break must always be set to No. Finally, we need to identify the field parameter that ACIF uses to locate the index. Select Field2 and add it to the Order list.

Click OK to add the index and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the indexes. Note the index parameters with the options and data values we specified in the Add an Index dialog box.

Close the Display ACIF Parameters dialog box.

Displaying triggers, fields, and indexes

Click the Display and Add Parameters icon on the toolbar to verify the indexing information. When you click the icon, the graphical indexer changes to display mode (note the status bar). The triggers and fields appear highlighted. Scroll through pages of the report to verify that they appear in the correct location on all pages. When you have finished, toggle the Display and Add Parameters icon to add mode.

Click the Select Trigger, Index, Field Parameters icon on the toolbar. The Select dialog box appears. Using this dialog box, you can display and maintain trigger, index, and field information. For example, click Field1. OnDemand highlights the area of the report where we defined the field. Click Field1 again. OnDemand highlights the area on the next page. Click Field2. OnDemand highlights the field in the report. Click Index1 and then click Properties. OnDemand displays the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties. OnDemand displays the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

ACIF Indexer Properties

After defining the View Information, triggers, fields, and indexes, we can complete the indexing information for the loan report by setting values in the ACIF Indexer Properties dialog box. This is a central place to maintain information that describes the format of the data, resources required to index the data, indexes generated by ACIF, and optional user-defined exit programs to process input, output, and index records and resources. Many of the parameter values are based on the choices made on the View Information page and trigger, field, and index definitions. We won't go over every field on every page; you can click Help on each page to display information about the fields. You can also refer to "Chapter 3. ACIF Parameter Reference" on page 61 for details.

You can review the parameter values on the Data Format page (although we don't need to change any of them).

The report contains line data that we don't want ACIF to convert to AFP. Therefore, we don't need to specify values on the Resource Information page.

Click the Output Information page. Enter 100 (one hundred) in the Max Pages in a Group field. This is the maximum number of pages in a group. The loan report will be indexed in groups of 100 pages.

We're not providing user exits for ACIF to process the data, index, or resources. Therefore, we don't need to specify values on the Exit Information page.

Click OK to save the changes and return to the report window. Since we're finished defining indexing information for the report, we can close the report window. OnDemand asks us if we want to save our changes. Click Yes. We return to the Indexer Information page.

Defining the application — part 2

Indexer parameters

The Indexer Parameters area now contains all of the indexing parameters required for ACIF to process the loan report. Use the scroll bars to review the parameters, including those added based on our settings in the ACIF Indexer Properties dialog box.

Load Information

The Load Information page is where we map the index attribute name we defined to hold loan number attribute values to application group data base fields. For the loan report, ACIF generates indexes for the first and last loan numbers in a group of pages. The attribute name is Loan Number. We want OnDemand to store the attribute values in the bgn_loan_num and end_loan_num database fields.

In the Application Group DB Name list, select `bgn_loan_num`. Type `Loan Number` in the Load ID Name field. Select `end_loan_num`. Type `Loan Number` in the Load ID Name field.

Adding the application

We've completed the minimum requirements for adding an application to the database, including defining the indexer information. Click OK to add the application to OnDemand and return to the Administrative Tasks window.

Example two

About the report

This example describes how to create index information for a sample telephone bill report. A telephone bill report typically contains hundreds of pages of line data. The report is logically segmented into statements. The beginning of a statement occurs when two conditions exist: a record that contains a skip-to-channel one carriage control and a record that contains the string ACCOUNT NUMBER. Each statement can contain one or more pages.

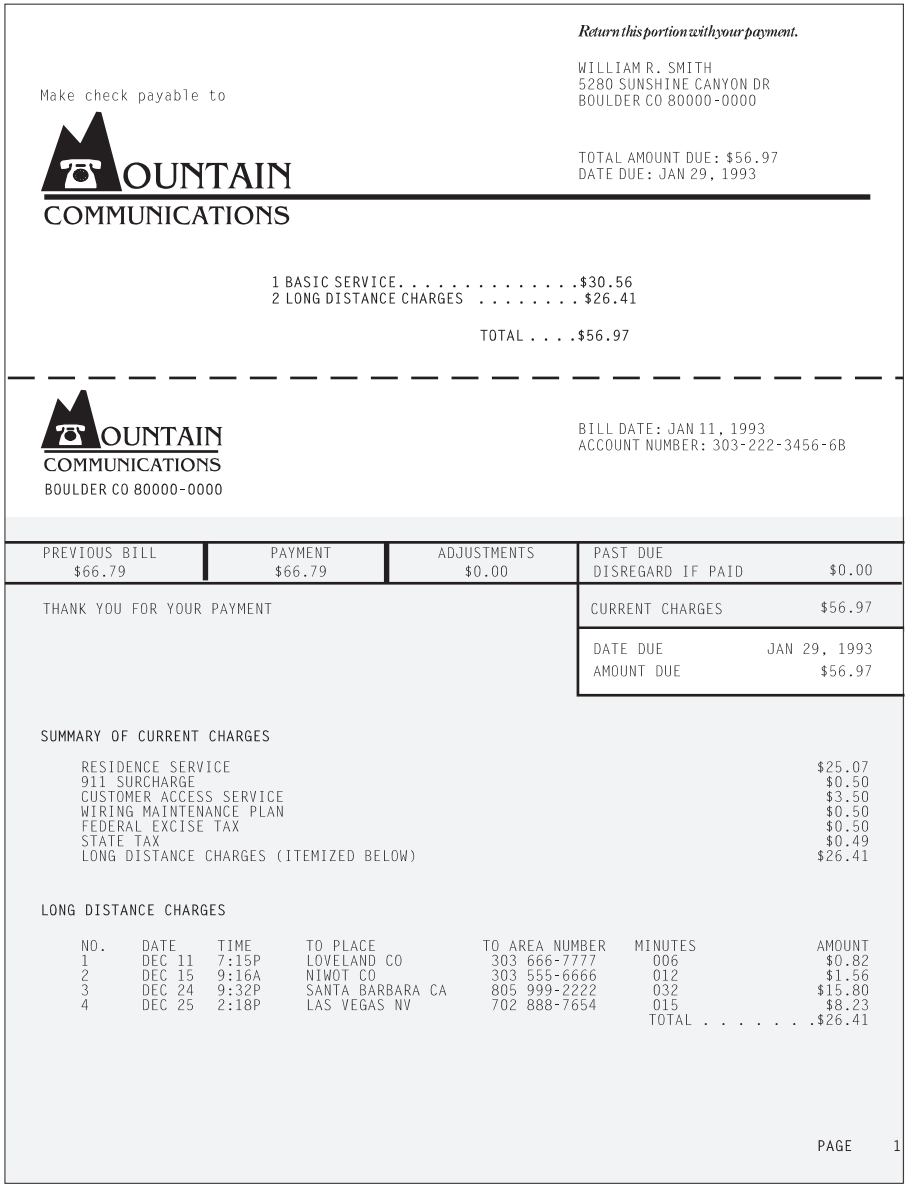
Because we want users to view the statements in the same format as the customer's printed copy, we'll use ACIF to convert the input line data to AFP and collect the resources required to view the statements. Figure 10 on page 26 shows an example of a statement viewed with one of the OnDemand client programs. Figure 11 on page 27 shows what the input data looks like viewed with an ISPF browser on the MVS system. Since the input data is encoded in EBCDIC, we must code the ACIF trigger and index parameter values in hexadecimal.

For ease of retrieval, we'll segment the report into groups of pages, one statement in each group. We'll create one index row for each group. The row contains three user-defined indexes: the account number, the customer's name, and the bill date. Figure 12 on page 28 shows the ACIF indexer parameters required to process the data.

Accessing the sample report

This example provides instructions about using the OnDemand graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the OnDemand administrative client, a Windows 32-bit application. To process a sample report, you typically create or extract a subset of a complete report. The report in this example was generated on an MVS system. We transferred the report to the PC as a binary file. If you download reports from an MVS system to an OnDemand server, you can mount the filesystem (or share the directory) that contains the reports on the PC. You can then load the sample data into the graphical indexer.

It is extremely important that the sample data used to create the indexing information matches the actual data to be indexed and loaded into the database. When you load a report, OnDemand uses the indexing parameters, options, and data values stored in the application definition to index the data. If the data being loaded does not match the data that you use to generate indexing parameters with the graphical indexer, OnDemand cannot properly index the data. For example, OnDemand won't be able to locate triggers, indexes, and fields or extract the correct index values.



Return this portion with your payment.

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

Make check payable to

MOUNTAIN COMMUNICATIONS

TOTAL AMOUNT DUE: $56.97
DATE DUE: JAN 29, 1993

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

TOTAL $56.97

MOUNTAIN COMMUNICATIONS
BOULDER CO 80000-0000

BILL DATE: JAN 11, 1993
ACCOUNT NUMBER: 303-222-3456-6B

PREVIOUS BILL	PAYMENT	ADJUSTMENTS	PAST DUE DISREGARD IF PAID
$66.79	$66.79	$0.00	$0.00

THANK YOU FOR YOUR PAYMENT

CURRENT CHARGES	$56.97
DATE DUE	JAN 29, 1993
AMOUNT DUE	$56.97

SUMMARY OF CURRENT CHARGES

RESIDENCE SERVICE	$25.07
911 SURCHARGE	$0.50
CUSTOMER ACCESS SERVICE	$3.50
WIRING MAINTENANCE PLAN	$0.50
FEDERAL EXCISE TAX	$0.50
STATE TAX	$0.49
LONG DISTANCE CHARGES (ITEMIZED BELOW)	$26.41

LONG DISTANCE CHARGES

NO.	DATE	TIME	TO PLACE	TO AREA NUMBER	MINUTES	AMOUNT
1	DEC 11	7:15P	LOVELAND CO	303 666-7777	006	$0.82
2	DEC 15	9:16A	NIWOT CO	303 555-6666	012	$1.56
3	DEC 24	9:32P	SANTA BARBARA CA	805 999-2222	032	$15.80
4	DEC 25	2:18P	LAS VEGAS NV	702 888-7654	015	$8.23
					TOTAL	$26.41

PAGE 1

Figure 10. Phone Bill

```
1                                                 WILLIAM R. SMITH
                                                  5280 SUNSHINE CANYON DR
                                                  BOULDER CO 80000-0000
-                                                 TOTAL AMOUNT DUE: $56.97
                                                  DATE DUE: JAN 29, 1993
-
-
0      1 BASIC SERVICE. . . . . . . . . . . . . .$30.56
       2 LONG DISTANCE CHARGES. . . . . . . . . .$26.41
0                                 TOTAL . . . .$56.97
-
0                                                 BILL DATE: JAN 11, 1993
                                                  ACCOUNT NUMBER: 303-222-3456-6B
-
-   $66.79           $66.79        $0.00                                  $0.00
                                                                         $56.97
                                                                   JAN 29, 1993
                                                                         $56.97
-
0 SUMMARY OF CURRENT CHARGES
0     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)                            $30.56
0 LONG DISTANCE CHARGES
0  NO.    DATE     TIME       TO PLACE           TO AREA NUMBER  MINUTES  AMOUNT
0  1      DEC 11   7:15P      LOVELAND CO        303 666-7777      006     $0.82
   2      DEC 15   9:16A      NIWOT CO           303 555-6666      012     $1.56
   3      DEC 24   9:32P      SANTA BARBARA CA   805 999-6666      032    $15.80
   4      DEC 25   2:18P      LAS VEGAS NV       702 888-7654      015     $8.23
-                                                      TOTAL . . . . . .$26.41
-
-
0                                                                      PAGE   1
```

Figure 11. Phone Bill Data Stream

```
/* DATA INPUT/OUTPUT CHARACTERISTICS                                    */
CC=YES                                  /* carriage controls present    */
CCTYPE=A                                /* ANSI carriage controls in EBCDIC */
CONVERT=YES                             /* line data to AFP             */
CPGID=500                               /* code page of the input data  */
FILEFORMAT=RECORD,133                   /* fixed length records         */

/* TRIGGER/FIELD/INDEX DEFINITIONS                                      */
TRIGGER1=*,1,X'F1',(TYPE=GROUP)                          /* 1           */
TRIGGER2=12,50,X'C1C3C3D6E4D5E340D5E4D4C2C5D9',(TYPE=GROUP) /* ACCOUNT NUMBER */
FIELD1=0,66,15,(TRIGGER=2,BASE=0)                        /* account number field */
FIELD2=0,50,30,(TRIGGER=1,BASE=0)                        /* name field   */
FIELD3=11,61,12,(TRIGGER=1,BASE=0)                       /* bill date field */
INDEX1=X'818383A36D95A494',field1,(TYPE=GROUP,BREAK=YES)     /* acct_num index */
INDEX2=X'95819485',field2,(TYPE=GROUP,BREAK=NO)              /* cust name index */
INDEX3=X'828993936D8481A385',field3,(TYPE=GROUP,BREAK=NO)    /* bill_date index */

/* INDEXING INFORMATION                                                 */
IMAGEOUT=ASIS                           /* leave image alone            */
INDEXOBJ=GROUP                          /* group-level indexes          */
INDEXSTARTBY=1                          /* must find index by page 1    */

/* RESOURCE INFORMATION                                                 */
CHARS=GT10                              /* coded font for AFP           */
FORMDEF=F1PHBILL                        /* formdef name required for AFP */
PAGEDEF=P1PHBILL                        /* pagedef name required for AFP */
FDEFLIB=/usr/lpp/psf/res/fdeflib        /* formdef directories          */
FONTLIB=/usr/lpp/psf/res/fontlib        /* font directories             */
OVLYLIB=/usr/lpp/psf/res/ovlylib        /* overlay directories          */
PDEFLIB=/usr/lpp/psf/res/pdeflib        /* pagedef directories          */
PSEGLIB=/usr/lpp/psf/res/pseglib        /* pseg directories             */
USERLIB=/tmp/res/phbill                 /* user resources               */
RESTYPE=fdef,pseg,ovly                  /* collect these resources      */
```

Figure 12. ACIF Parameters

Key concepts

Group Trigger. Group triggers identify the beginning of a group. You must define at least one group trigger. Trigger1 must be a group trigger.

Group Index. Indexes generated once for each group. All data stored in OnDemand must be indexed by group (even if a group contains only one page).

Index Break. Indexes that determine when ACIF closes the current group and begins a new group. One of the group indexes determines when ACIF breaks groups. However, a group index based on a floating trigger cannot be used to control the group break.

Convert. Determines whether ACIF converts the input data to AFP. You typically convert line data to AFP to format the data into pages and enhance the appearance of the output with images, graphics, fonts, and bar codes.

Resources. Objects required to load, view, and print AFP data. If the input data is AFP or you convert line data to AFP, you must specify resources and resource paths.

Defining the application — part 1

An application identifies the type of data that is stored in OnDemand, the method used to index the data, and other information that OnDemand uses to load and view reports. This topic provides information about the application we created to process the sample phone bill report.

General

The General page is where we name the application and assign the application to an application group.

We assigned the application to the application group where we plan to store the phone bill report. The application group contains database field definitions for the account number, customer name, and bill date.

View Information

The View Information page is where we specify information needed by OnDemand client programs to display the phone bills. This information is also used by the indexing program.

Even though the phone bill report will be stored in OnDemand as AFP data, we initially set the Data Type to Line to prepare the indexer information. After completing the indexer information, we will reset the Data Type to AFP, which is the format of the data as stored in OnDemand. Other important settings on this page include:

- Code Page. We set the code page to 500. This is the code page of the input data required by ACIF and the graphical indexer.
- RECFM. Records in the input data are fixed length, 133 characters in length.
- CC. The input data contains carriage control characters in column one of the data.
- CC Type. The input data contains ANSI carriage control characters coded in EBCDIC.

Indexer Information

The Indexer Information page is where we need to do the bulk of our work.

First, we change the Indexer to ACIF. Notice the ACIF indexing parameters, options, and data values that the administrative client automatically set, based on our choice of indexer and the settings we chose on the View Information page:

- `CONVERT=NO`. The input file is line data. Although we plan to store the report in OnDemand as AFP data, we first need to process a sample of the source data with the ACIF graphical indexer. After generating the indexing parameters, we'll change CONVERT to YES.
- `CPGID=500`. The code page of the data as stored in OnDemand.
- `FILEFORMAT=RECORD,133`. The fileformat parameter contains information that identifies the format and length of the input records.

The remaining parameters are standard ACIF parameters with default values for processing line data.

We need to define additional ACIF parameters, including those that determine the index data extracted from the report and loaded into OnDemand. We will process sample report data with the OnDemand graphical indexer and define the parameters.

Opening the sample report

First, in the Parameter Source area, select Sample Data. Then click Modify. OnDemand displays the Open dialog box. Select the name of the file that contains the sample data. Click Open. OnDemand loads the file into the report window.

The name of the input file is displayed at the top of the window along with a warning that the data must match the data being loaded.

Note: When you load a report, OnDemand uses the indexing parameters, options, and data values stored in the application definition to index the data. If the data being loaded does not match the data that you use to generate indexing parameters with the graphical indexer, OnDemand cannot properly index the data. For example, OnDemand will not be able to locate triggers, indexes, and fields or extract correct index values.

When processing sample data with the graphical indexer, you typically define triggers first, then fields, and finally, indexes.

Defining triggers

ACIF uses one or more triggers to determine where to begin to locate index values. For the phone bill report, we need to define two triggers:

- A trigger that instructs ACIF to examine the first byte of every input record for the presence of an EBCDIC skip-to-channel one carriage control character (X'F1'). This is the TRIGGER1 record.
- A trigger that instructs ACIF to locate the string ACCOUNT NUMBER starting in column 50 of the 12th record following the TRIGGER1 record.

Together, these triggers uniquely identify the start of a statement in the phone bill report.

Defining TRIGGER1

Since the graphical indexer displays the report as it is viewed in OnDemand, we cannot see the carriage control characters in column one of the data. To define the trigger, select any column in the first record. When you select a column, the graphical indexer highlights the data. Next, click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger1) determines the name of the trigger parameter. Trigger1 must always be defined and establishes a starting point where other group triggers and non-floating fields can be found. The Records to Search area determines the records ACIF searches to locate the trigger. For the phone bill report, we want ACIF to search every record. The Columns to Search area determines the column number of the trigger record where ACIF begins to search for the trigger string value. We set the Columns to Search to Carriage Control so that ACIF searches column one of each record. When we do this, the graphical indexer displays the trigger string value (X'F1') in the Value area.

Click OK to add the trigger and return to the report window.

Locating the account number

To define the trigger used to locate account numbers in the phone bill report, select the string ACCOUNT NUMBER in the report. The graphical indexer highlights the string. Click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger2) determines the name of the trigger parameter. The trigger Type is Group. We want ACIF to create a group index for each account number it finds in the phone bill report. The Records to Search area determines the records ACIF searches to locate the trigger. For a group trigger other than TRIGGER1, the record is based on TRIGGER1. Since the trigger string value can be found in a specific record in each group, only one record will be searched (12 in the sample report). The Columns to Search area determines the columns of the trigger record ACIF searches. The graphical indexer displays the starting column number of the string we selected in the

report. In the sample report, ACIF begins its search in column 50. The Value area contains the trigger string value that ACIF searches for.

Click OK to add the trigger and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the triggers. Note the two trigger parameters with the options and data values we specified in the Add a Trigger dialog box.

Close the Display ACIF Parameters dialog box.

Defining fields

ACIF uses field definitions to determine where to locate index values. For the sample phone bill report, we must define three fields:

- A field that instructs ACIF to locate account number values beginning in column 66 of the TRIGGER2 record.
- A field that instructs ACIF to locate customer name values beginning in column 50 of the TRIGGER1 record.
- A field that instructs ACIF to locate the date of the phone bill beginning in column 61 of the 11th record following the TRIGGER1 record.

Defining the account number field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected the account number on the first page in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field1) determines the name of the field parameter. The Trigger determines the name of the trigger parameter that ACIF uses to locate the field. By default, the trigger is TRIGGER1. Select TRIGGER2 from the list, so that ACIF uses TRIGGER2 to locate the field. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger. The Columns to Search area determines the column number (66) where ACIF locates the beginning of the field. The Size area determines the length (15) of the field. The Reference String area lists the field value we selected.

Click OK to add the field and return to the report window.

Defining the customer name field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected the customer name on the first page in the sample report. We included a number of blank columns (to the right of the name) in the string we selected. We want to make sure that the length of the field that we define contains enough characters to hold the longest name ACIF will encounter in the report. For example, if the field can include values up to 30 characters in length and the sample value is only 17 characters in length, we must select an additional 13 columns in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field2) determines the name of the field parameter. The Trigger (Trigger1) determines the name of the trigger parameter that ACIF uses to locate the field. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger. The Columns to Search area determines the column number (50) where ACIF locates the beginning of the field. The Size area determines the length (30) of the field. The Reference String area lists the field value we selected.

We don't need to make any changes to the information extracted by the graphical indexer. Click OK to add the field and return to the report window.

Defining the bill date field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected the billing date on the first page in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field3) determines the name of the field parameter. The Trigger (Trigger1) determines the name of the trigger parameter that ACIF uses to locate the field. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger. The Columns to Search area determines the column number (61) where ACIF locates the beginning of the field. The Size area determines the length (12) of the field. The Reference String area lists the field value we selected.

We don't need to make any changes to the information extracted by the graphical indexer. Click OK to add the field and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing

parameters. We can see the result of defining the fields. Note the field parameters with the options and data values we specified in the Add a Field dialog box.

Close the Display ACIF Parameters dialog box.

Defining indexes

Our next task in defining indexing parameters for the phone bill report is to define indexes. The index definitions determine attribute names and values stored in the database and the type of index ACIF creates. For the phone bill report, we must define three indexes. ACIF extracts three index values for each group in the report.

Defining the account number index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index1) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, `acct_num`. When we load data into the application group, the account number index values will be stored into this database field. The Type of Index determines the type of index ACIF generates. For each index that we define in this example, we want ACIF to extract a value for each group in the report. The Break area determines whether ACIF closes the current group and begins a new group when the index value changes. Since we want ACIF to use the account number index to control the group break, we set Break to Yes. The Fields area lists the field parameters that have been defined for the report. We need to identify the field parameter that ACIF uses to locate the index. For the account number index values, that is Field1. Select Field1 in the Fields list and click Add. OnDemand moves Field1 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

Defining the customer name index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index2) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, `name`. When we load the report into the application group, the customer name index values will be stored into this database field. The Type of Index determines the type of index ACIF generates. For all indexes in this example, we want

ACIF to generate an index for each group in the report. The Break areas determines whether ACIF closes the current group and begins a new group when the index value changes. Since we do not want ACIF to use the customer name index to control the group break, we set Break to No. The Fields area lists the field parameters that have been defined (in the Fields list) for the report. We need to identify the field parameter that ACIF uses to locate the index. For the customer name index values, that is Field2. Select Field2 in the Fields list and click Add. OnDemand moves Field2 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

Defining the bill date index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index3) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, `bill_date`. When we load the report into the application group, the billing date index values will be stored into this database field. The Type of Index determines the type of index ACIF generates. For each index that we define in this example, we want ACIF to extract a value for each group in the report. The Break area determines whether ACIF closes the current group and begins a new group when the index value changes. Since we do not want ACIF to use the billing date index to control the group break, we set Break to No. The Fields area lists the field parameters that have been defined for the report. We need to identify the field parameter that ACIF uses to locate the index. For the billing data index values, that is Field3. Select Field3 in the Fields list and click Add. OnDemand moves Field3 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the indexes. Note the index parameters with the options and data values we specified in the Add an Index dialog box.

Close the Display ACIF Parameters dialog box.

Displaying triggers, fields, and indexes

Click the Display and Add Parameters icon on the toolbar to verify the indexing information. When you click the icon, the graphical indexer changes to display mode (note the status bar). The triggers and fields appear highlighted. Scroll through pages of the report to verify that they appear in the correct location on all pages. When you have finished, toggle the Display and Add Parameters icon to add mode.

Click the Select Trigger, Index, Field Parameters icon on the toolbar. The Select dialog box appears. Using this dialog box, you can display and maintain trigger, index, and field information. For example, click Field1. OnDemand highlights the area of the report where we defined the field. Click Field1 again. OnDemand highlights the area on the next page. Click Field2. OnDemand highlights the field in the report. Click Index1 and then click Properties. OnDemand displays the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties. OnDemand displays the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

ACIF Indexer Properties

After defining the View Information, triggers, fields, and indexes, we can complete the indexing information for the phone bill report by setting values in the ACIF Indexer Properties dialog box. This is a central place to maintain parameters that describe the format of the data, resources required to view and print the data, indexes generated by ACIF, and optional user-defined exit programs to process input, output, and index records and resources. Many of the parameter values are based on the choices made on the View Information page and trigger, field, and index definitions. We won't go over every field on every page; you can click Help on each page to display information about the fields. You can also refer to "Chapter 3. ACIF Parameter Reference" on page 61 for details.

Review the parameter values on the Data Format page. Since we want to store the line data input in OnDemand as AFP data, we must set Data Conversion to Yes. When we do this, the administrative client automatically changes the Data Type to AFP on the View Information page. The file format and carriage control areas retain the original settings we made on the View Information page. We don't need to make any changes to the values in the Font Information area.

Click the Resource Information tab. Since we're converting the input data to AFP, we must specify values on this page. For the sample phone bill report, we specify the name of a form definition and page definition. We want ACIF to collect form definitions, overlays, and page segments, so we check the appropriate boxes in the Resource File Contents area. Finally, in the Search

Paths area, we identify where ACIF can find the resources. Enter the names of the directories where the resources reside. For example, in the User Directories field, we entered `/tmp/res/phbill` so that ACIF can find our user-defined form definition and page definition required to process the input data.

Review the information on the Output Information page. We don't have to make any changes for the sample phone bill report.

We're not providing user exits for ACIF to process the data, index, or resources so we don't need to specify values on the Exit Information page.

Click OK to save the changes and return to the report window. Since we're finished defining indexing information for the report, we can close the report window. OnDemand asks us if we want to save our changes. Click Yes. We return to the Indexer Information page.

Defining the application — part 2

View Information

The View Information page is where we specify information needed by OnDemand client programs to display the phone bills.

We initially set the Data Type to Line to prepare the indexer information. After generating the indexing parameters, we set Convert to Yes (in the ACIF Indexer Parameters dialog box). When we did that, the administrative client automatically set the Data Type to AFP, which is the format of the data as stored in OnDemand.

Indexer parameters

The Indexer Parameters area now contains all of the indexing parameters required for ACIF to process the phone bill report. Use the scroll bars to review the parameters, including those added based on our settings in the ACIF Indexer Properties dialog box.

Load Information

The Load Information page is where we map the index attribute names we defined to hold the attribute values ACIF extracts from the report to application group database field names. For the sample phone bill report, we named the attributes the same as the database fields. Therefore, we do not need to map the attributes names on the Load Information page. To verify this, select each name in the Application Group DB Name list. The corresponding index attribute name appears in the Load ID Name field. The names should be the same.

The Load Information page also contains other values used when OnDemand stores index data in the database. For example, if the appearance of the date field in the report is different than the default date format for the application group, you can identify the date format for the report. Verify the date format for the bill date field. Select `bill_date` in the Application Group DB Name list. The Format field should contain the format specifier that describes the appearance of the date in the report. If it does not, select the format specifier that correctly describes the appearance of the date in the report.

Adding the application

We've completed the minimum requirements for adding an application to the database, including defining the indexer information. Click OK to add the application to OnDemand and return to the Administrative Tasks window.

Example three

About the report

This example describes how to create index information for a sample income statement report. An income statement report typically contains hundreds of pages of line data. The report is logically segmented into statements. The beginning of a statement occurs when two conditions exist: a record that contains a skip-to-channel one carriage control and a record that contains the string *Income Statement*. Each statement can contain one or more pages. Figure 13 on page 39 shows an income statement viewed with one of the OnDemand client programs.

For ease of retrieval, we'll segment the report into groups of pages, one statement in each group. We'll create one index row for each group. The row contains three user-defined indexes: the account number, the statement date, and the total income. In addition, we will create page-level indexes to allow users to locate the type of income and the subtotal when they view a statement. Because page-level indexes are only supported by AFP, we must convert the input line data to AFP. The page-level indexes are stored with the AFP document, not in the database (you cannot use page-level indexes to search for documents). Figure 14 on page 40 shows the ACIF indexer parameters required to process the data.

Accessing the sample report

This example provides instructions about using the OnDemand graphical indexer to process a sample report and create indexing information. The graphical indexer is part of the OnDemand administrative client, a Windows 32-bit application. To process a sample report, you typically create or extract a subset of a complete report. The report used in this example was generated on

an MVS system. We transferred the report to the PC as a binary file. If you download reports from an MVS system to an OnDemand server, you can mount the filesystem (or share the directory) that contains the reports on the PC. You can then load the sample data into the graphical indexer.

It is extremely important that the sample data used to create the indexing information matches the actual data to be indexed and loaded into the database. When you load a report, OnDemand uses the indexing parameters, options, and data values stored in the application definition to index the data. If the data being loaded does not match the data that you use to generate indexing parameters with the graphical indexer, OnDemand cannot properly index the data. For example, OnDemand won't be able to locate triggers, indexes, and fields or extract the correct index values.

```
                                                          Page 1 of 5
Income Statement# 123-45-6789                             Date: 09/1994

                           Eugene & Pearl Aardvark
                             18005 Le May Street
                             West Hills PA 12345

                           Total Income - $  2,931.26
    Type of Income - W2 Wages
    Subtotal:       1,015.00
                                          Source                Amount
                                          The Pastry Shoppe       1,015.00
    Type of Income - Interest
    Subtotal:         491.35
                                          Source                Amount
                                          Big Bank                  123.45
                                          TPS Credit Union          367.90
    Type of Income - SEP/IRA
    Subtotal:          50.00
                                          Source                Amount
                                          LOTTO                      50.00
    Type of Income - Dividend
    Subtotal:          53.91
                                          Source                Amount
                                          XVT Railroad               53.91
    Type of Income - Farm
    Subtotal:       1,321.00
                                          Source                Amount
                                          CRP                     1,321.00
```

Figure 13. Income Statement

```
/* DATA INPUT/OUTPUT CHARACTERISTICS                                    */
CC=YES                                  /* carriage controls present    */
CCTYPE=A                                /* ANSI controls in EBCDIC      */
CONVERT=YES                             /* line data to AFP because we  */
                                        /*   need to generate page-level */
                                        /*   index information          */
CPGID=500                               /* code page ID                 */
TRC=NO                                  /* table ref chars not present  */
FILEFORMAT=RECORD,133                   /* Fixed length input file      */

/* TRIGGER/FIELD/INDEX DEFINITIONS                                      */
TRIGGER1 = *,1,X'F1',(TYPE=GROUP)                    /* 1               */
FIELD1   = 1,73,7,(TRIGGER=1,BASE=0)                 /* sdate field     */
INDEX1   = X'A28481A385',FIELD1,(TYPE=GROUP,BREAK=YES)  /* sdate index */
TRIGGER2 = 1,2,X'C9958396948540E2A381A385948595A3',(TYPE=GROUP) /* Income Statement */
FIELD2   = 0,20,11,(TRIGGER=2,BASE=0)               /* incstmt field    */
INDEX2   = X'899583A2A394A3',field2,(TYPE=GROUP,BREAK=YES)  /* incstmt index */
/*                                                     /* Total Income  */
TRIGGER3 = *,31,X'E396A3819340C99583969485',(TYPE=GROUP,RECORDRANGE=(7,8))
FIELD3   = 0,46,10,(TRIGGER=3,BASE=0)                  /* totinc field  */
INDEX3   = X'A396A3899583',FIELD3,(TYPE=GROUP,BREAK=NO)  /* totinc index */
TRIGGER4 = *,5,X'E3A8978540968640C99583969485',(TYPE=FLOAT)  /* Type of Income */
FIELD4   = 0,22,12,(TRIGGER=4,BASE=0)               /* Type of Income field */
INDEX4   = X'E3A8978540968640C99583969485',field4,(TYPE=PAGE)  /* Type of Income index */
TRIGGER5 = *,5,X'E2A482A396A38193',(TYPE=FLOAT)         /* Subtotal      */
FIELD5   = 0,17,10,(TRIGGER=5,BASE=0)                  /* Subtotal field */
INDEX5   = X'E2A482A396A38193',field5,(TYPE=PAGE)       /* Subtotal index */

/* INDEXING INFORMATION                                                 */
IMAGEOUT=ASIS                           /* leave images alone           */
INDEXOBJ=ALL                            /* group and page indexes       */
INDEXSTARTBY=1                          /* must find index by page 1    */

/* RESOURCE INFORMATION                                                 */
CHARS=GT10                              /* coded font for AFP           */
FORMDEF=F1A10110                        /* formdef name required for AFP */
PAGEDEF=P1A08682                        /* pagedef name required for AFP */
FDEFLIB=/usr/lpp/psf/res/fdeflib        /* formdef directories          */
PDEFLIB=/usr/lpp/psf/res/pdeflib        /* pagedef directories          */
RESTYPE=fdef                            /* collect these resources      */
```

Figure 14. ACIF Parameters

Key concepts

Group Trigger. Group triggers identify the beginning of a group. You must define at least one group trigger. Trigger1 must be a group trigger.

Group Index. Indexes generated once for each group. All data stored in OnDemand must be indexed by group (even if a group contains only one page).

Recordrange Trigger. Triggers that can be found in a range of records. For example, the trigger string value is Total Income. The string may appear in record seven or eight, depending on whether the address contains three or four lines. Define a recordrange trigger to cause ACIF to search records seven and eight for the trigger string value.

Float Trigger. Triggers that do not necessarily occur in the same location on each page, the same page in each group, or each group. For example, customer statements contain one or more accounts. Not every statement contains all types of accounts. The location of the account type does not appear on the same line or page in every statement. Define a float trigger to locate each type of account, regardless of where it appears in the statement.

Page Index. Indexes that may be created zero or more times for each page in the group. A page index identifies one and only one field. The field must be based on a floating trigger. Since the field is based on a floating trigger, the page index may or may not occur. Page indexes are only allowed within group indexes. Page indexes cannot break a group index. 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 from the server, you can use the page indexes to navigate pages of the document with the Go To command.

Index Break. Indexes that determine when ACIF closes the current group and begins a new group. One or more group indexes can be used to determine when ACIF breaks groups. However, a group index based on a floating trigger cannot be used to control group breaks. A page index cannot be used to control group breaks.

Convert. Determines whether ACIF converts the input data to AFP. You must convert input line data to AFP to generate page-level indexes and store them with the AFP documents.

Resources. Objects required to load, view, and print AFP data. If the input data is AFP or you convert line data to AFP, you must specify resources and resource paths.

Defining the application — part 1

An application identifies the type of data that is stored in OnDemand, the method used to index the data, and other information that OnDemand uses to load and view reports. This topic provides information about the application we created to process the sample income statement report.

General

The General page is where we name the application and assign the application to an application group.

We assigned the application to the application group where we plan to store the income statement report. The application group contains database field definitions for the statement number, statement date, total income, types of income, and subtotals.

View Information

The View Information page is where we specify information needed by OnDemand client programs to display the income statements. This information is also used by the indexing program.

Even though the income statement report will be stored in OnDemand as AFP data, we initially set the Data Type to Line to prepare the indexer information. After completing the indexer information, we will reset the Data Type to AFP, which is the format of the data as it is stored in OnDemand. Other important settings on this page include:

- Code Page. We set the code page to 500. This is the code page of the data as stored in OnDemand and viewed using programs such as the graphical indexer.
- RECFM. Records in the input data are fixed length, 133 characters in length.
- CC. The input data contains carriage control characters in column one of the data.
- CC Type. The input data contains ANSI carriage control characters coded in EBCDIC.

Indexer Information

The Indexer Information page is where we need to do the bulk of our work.

First, we change the Indexer to ACIF. Notice the ACIF indexing parameters, options, and data values that the administrative client automatically set, based on our choice of indexer and the settings we chose on the View Information page:

- `CONVERT=NO`. The input file is line data. Although we plan to store the report in OnDemand as AFP data, we first need to process a sample of the source data with the ACIF graphical indexer. After generating the indexing parameters, we will change CONVERT to YES.
- `CPGID=500`. The code page of the data as stored in OnDemand.
- `FILEFORMAT=RECORD,133`. The fileformat parameter contains information that identifies the format and length of the input records.

The remaining parameters are standard ACIF parameters with default values for processing line data.

We need to define additional ACIF parameters, including those that determine the index data extracted from the report and loaded into OnDemand. We will process sample report data with the OnDemand graphical indexer and define the parameters.

Opening the sample report

First, in the Parameter Source area, select Sample Data. Then click Modify. OnDemand displays the Open dialog box. Select the name of the file that contains the sample data. Click Open. OnDemand loads the file into the report window.

The name of the input file is displayed at the top of the window along with a warning that the data must match the data being loaded.

Note: When you load a report, OnDemand uses the indexing parameters, options, and data values stored in the application definition to index the data. If the data being loaded does not match the data that you use to generate indexing parameters with the graphical indexer, OnDemand cannot properly index the data. For example, OnDemand will not be able to locate triggers, indexes, and fields or extract correct index values.

When processing sample data with the graphical indexer, you typically define triggers first, then fields, and finally, indexes.

Defining triggers

ACIF uses one or more triggers to determine where to begin to locate index values. For the income statement report, we need to define five triggers:

- A trigger that instructs ACIF to examine the first byte of every input record for the presence of an EBCDIC skip-to-channel one carriage control character (X'F1'). This is the TRIGGER1 record.
- A trigger that instructs ACIF to examine every input record for the string `Income Statement` beginning in column two of the record following the TRIGGER1 record. This trigger, along with TRIGGER1, uniquely identifies the start of a statement in the report.
- A trigger that instructs ACIF to locate the string `Total Income` starting in column 31 of the seventh or eighth record following the TRIGGER1 record. The trigger string value can occur in one of two records because it follows the address, which may contain three or four lines.
- A trigger that instructs ACIF to locate the string `Type of Income` starting in column 5 of any record in the group. Since a statement can contain one or

more types of income, there may be several records that contain this trigger string value. We want ACIF to collect all income types for each group.

- A trigger that instructs ACIF to locate the string `Subtotal` starting in column 5 of any record in the group. A subtotal value is associated with a type of income, so we want ACIF to collect all the subtotals for each group.

Defining TRIGGER1

Since the graphical indexer displays the report as it is viewed in OnDemand, we cannot see the carriage control characters in column one of the data. To define the trigger, select any column in the first record. When you select a column, the graphical indexer highlights the data. Next, click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger1) determines the name of the trigger parameter. Trigger1 must always be defined and establishes a starting point where other group triggers and non-floating fields can be found. The Records to Search area determines the records ACIF searches to locate the trigger. For the income statement report, we want ACIF to search every record. The Columns to Search area determines the column number of the trigger record where ACIF begins to search for the trigger string value. We set the Columns to Search to Carriage Control so that ACIF searches column one of each record. When we do this, the graphical indexer displays the trigger string value (X'F1') in the Value area.

Click OK to add the trigger and return to the report window.

Locating the statement number

To define the trigger used to locate the income statement number, select the string `Income Statement` in the report. The graphical indexer highlights the string. Click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger2) determines the name of the trigger parameter. Trigger2 is a group trigger that, along with Trigger1, establishes the beginning of an income statement. The Records to Search area shows that ACIF can locate this trigger in the first record after the TRIGGER1 record. The Columns to Search area determines the columns of the trigger record ACIF searches. The graphical indexer displays the starting column number of the string we selected in the report. ACIF begins its search in this column (2 in the sample report). The Value area contains the trigger string value that ACIF searches for.

Click OK to add the trigger and return to the report window.

Locating the total income

To define the trigger used to locate the total income values in the income statement report, select the string Total Income in the report. The graphical indexer highlights the string. Click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger3) determines the name of the trigger parameter. The trigger Type is Group. We want ACIF to create a total income index for each group in the report. The Records to Search area determines the records ACIF searches to locate the trigger. Since we know that the total income value can occur in one of two records (depending on the number of address lines in a statement), we must specify a Record Range. The Start value is the first record that can contain the total income value. In our example, the Start value is the number of the record (7, seven) that contains the trigger string value we selected in the report. The End value is the last record that ACIF searches for the trigger. In our example, the End value is 8 (eight). The Columns to Search area determines the column of the trigger record where ACIF begins to search for the trigger string value. The graphical indexer displays the starting column number of the string we selected in the report. In the sample report, ACIF begins its search in column 31. The Value area contains the trigger string value that ACIF searches for.

Click OK to add the trigger and return to the report window.

Locating the type of income

To define the trigger used to locate the type of income, select the string Type of Income in the report. The graphical indexer highlights the string. Since we plan to collect index values for all the income types found in each group, it doesn't matter which Type of Income string we select (if there is more than one on the page currently displayed in the report window). Click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger4) determines the name of the trigger parameter. An income statement can contain one or more income types. We need ACIF to search every record in the group. A float trigger is how this is accomplished. Change the Type to Float. When we change the Type to Float, the graphical indexer automatically sets the Records to Search to Every Record. The Columns to Search area determines the columns of the trigger record ACIF searches. The graphical indexer displays the starting column number of the string we selected in the report. ACIF begins its search in this column (5 in the sample report). The Value area contains the trigger string value that ACIF searches for.

Click OK to add the trigger and return to the report window.

Locating the subtotal

To define the trigger used to locate the subtotal, select the string Subtotal in the report. The graphical indexer highlights the string. Since we plan to collect index values for all the subtotals found in each group, it doesn't matter which Subtotal string we select (if there is more than one on the page in the report window). Click the Trigger icon on the toolbar. OnDemand displays the Add a Trigger dialog box.

The Identifier (Trigger5) determines the name of the trigger parameter. An income statement can contain one or more subtotals (one for each income type). We need ACIF to search every record in the group. A float trigger is how this is accomplished. Change the Type to Float. When we change the Type to Float, the graphical indexer automatically sets the Records to Search to Every Record. The Columns to Search area determines the columns of the trigger record ACIF searches. The graphical indexer displays the starting column number of the string we selected in the report. ACIF begins its search in this column (5 in the sample report). The Value area contains the trigger string value that ACIF searches for.

Click OK to add the trigger and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the triggers. Note the five trigger parameters with the options and data values we specified in the Add a Trigger dialog box.

Close the Display ACIF Parameters dialog box.

Defining fields

ACIF uses field definitions to determine where to locate index values. For the sample income statement report, we must define five fields:

- A field that instructs ACIF to locate statement date values beginning in column 73 of the record following the TRIGGER1 record.
- A field that instructs ACIF to locate income statement number values beginning in column 20 of the TRIGGER2 record.
- A field that instructs ACIF to locate total income values beginning in column 46 of the TRIGGER3 record.
- A field that instructs ACIF to locate type of income values beginning in column 22 of the TRIGGER4 record.

- A field that instructs ACIF to locate subtotal values beginning in column 17 of the TRIGGER5 record.

Defining the statement date field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected the statement date on the first page in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field1) determines the name of the field parameter. The Trigger determines the name of the trigger parameter that ACIF uses to locate the field. By default, ACIF uses TRIGGER1. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger (in our example, one). The Columns to Search area determines the column number (73) where ACIF locates the beginning of the field. The Size area determines the length (7) of the field. The Reference String area lists the field value we selected.

Click OK to add the field and return to the report window.

Defining the statement number field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected the statement number on the first page in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field2) determines the name of the field parameter. The Trigger determines the name of the trigger parameter that ACIF uses to locate the field. By default, ACIF uses TRIGGER1. Since we want ACIF to locate the statement number field using TRIGGER2, we must select TRIGGER2 from the Trigger list. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger (in our example, zero). The Columns to Search area determines the column number (20) where ACIF locates the beginning of the field. The Size area determines the length (11) of the field. The Reference String area lists the field value we selected.

Click OK to add the field and return to the report window.

Defining the total income field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected the total income value on the first page in the sample report. We included two blank columns (to the left of the value) in the string we selected. We want to make sure that the

length of the field that we define contains enough characters to hold the largest total income value ACIF will encounter in the report. For example, if the field can include values up to ten characters in length and the sample value is only eight characters in length, we must select an additional two columns in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field3) determines the name of the field parameter. The Trigger determines the name of the trigger parameter that ACIF uses to locate the field. By default, ACIF uses TRIGGER1. Since we want ACIF to locate the total income field using TRIGGER3, we must select TRIGGER3 from the Trigger list. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger (in our example, zero). The Columns to Search area determines the column number (46) where ACIF locates the beginning of the field. The Size area determines the length (10) of the field. The Reference String area lists the field value we selected.

Click OK to add the field and return to the report window.

Defining the type of income field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected one of the type of income values on the first page in the sample report. We included several blank columns (to the right of the value) in the string we selected. We want to make sure that the length of the field that we define contains enough characters to hold the longest type of income value ACIF will encounter in the report. For example, if the field can include values up to twelve characters in length and the sample value is only eight characters in length, we must select an additional four columns in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field4) determines the name of the field parameter. The Trigger determines the name of the trigger parameter that ACIF uses to locate the field. By default, ACIF uses TRIGGER1. Since we want ACIF to locate the type of income field using TRIGGER4, we must select TRIGGER4 from the Trigger list. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger (in our example, zero). The Columns to Search area determines the column number (22) where ACIF locates the beginning of the field. The Size area determines the length (12) of the field. The Reference String area lists the field value we selected.

Click OK to add the field and return to the report window.

Defining the subtotal field

Select a field by clicking on the area in the report that contains the field data. The graphical indexer highlights the string. We selected one of the subtotal values on the first page in the sample report. We included two blank columns (to the left of the value) in the string we selected. We want to make sure that the length of the field that we define contains enough characters to hold the largest subtotal value ACIF will encounter in the report. For example, if the field can include values up to ten characters in length and the sample value is only eight characters in length, we must select an additional two columns in the sample report. Click the Define a Field icon on the toolbar. The graphical indexer displays the Add a Field dialog box.

The Identifier (Field5) determines the name of the field parameter. The Trigger determines the name of the trigger parameter that ACIF uses to locate the field. By default, ACIF uses TRIGGER1. Since we want ACIF to locate the total income field using TRIGGER5, we must select TRIGGER5 from the Trigger list. The Records to Search area contains the number of the record where ACIF can find the field, offset from the trigger (in our example, zero). The Columns to Search area determines the column number (19) where ACIF locates the beginning of the field. The Size area determines the length (10) of the field. The Reference String area lists the field value we selected.

Click OK to add the field and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the fields. Note the field parameters with the options and data values we specified in the Add a Field dialog box.

Close the Display ACIF Parameters dialog box.

Defining indexes

Our next task in defining indexing parameters for the income statement report is to define indexes. The index definitions determine attribute names and values of the group indexes stored in the database and the page indexes stored with the AFP document and the type of index ACIF creates. For the income statement report, we must define five indexes. ACIF extracts a minimum of five index values for each group in the report. ACIF can extract additional page-level index values if there is more than one type of income present in a statement.

Defining the statement date index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index1) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, `sdate`. When we load the report into the application group, the date index values will be stored into this database field. The Type of Index determines the type of index ACIF generates. Since we want ACIF to extract one date index value for each group in the report, we accept the default Type of Group. The Break areas determines whether ACIF closes the current group and begins a new group when the index value changes. Since we want ACIF to use the date index to control the group break, we set Break to Yes. The Fields area lists the field parameters that have been defined (in the Fields list) for the report. We need to identify the field parameter that ACIF uses to locate the index. For the statement date index values, that is Field1. Select Field1 in the Fields list and click Add. OnDemand moves Field1 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

Defining the statement number index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index2) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, `incstmt`. When we load data into the application group, the statement number index values will be stored into this database field. The Type of Index determines the type of index ACIF generates. Since we want ACIF to extract one income statement value for each group in the report, we accept the default Type of Group. The Break area determines whether ACIF closes the current group and begins a new group when the index value changes. Since we want ACIF to use the statement number index to control the group break, we set Break to Yes. The Fields area lists the field parameters that have been defined for the report. We need to identify the field parameter that ACIF uses to locate the index. For the statement number index values, that is Field2. Select Field2 in the Fields list and click Add. OnDemand moves Field2 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

Defining the total income index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index3) determines the name of the index parameter. The Attribute is the name of the index. We accept the suggested default, `totinc`. When we load the report into the application group, the total income index values will be stored into this database field. The Type of Index determines the type of index ACIF generates. Since we want ACIF to extract one total income value for each group in the report, we accept the default Type of Group. The Break area determines whether ACIF closes the current group and begins a new group when the index value changes. Since we do not want ACIF to use the total income index to control the group break, we set Break to No. The Fields area lists the field parameters that have been defined for the report. We need to identify the field parameter that ACIF uses to locate the index. For the total income index values, that is Field3. Select Field3 in the Fields list and click Add. OnDemand moves Field3 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

Defining the type of income index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index4) determines the name of the index parameter. The Attribute is the name of the index. For AFP documents, OnDemand displays the attribute names and values of page indexes in the Go To dialog box, allowing users a way to navigate pages of a statement. Attribute names should be meaningful to the user. Enter `Type of Income`. The Type determines the type of index ACIF generates. Since we want ACIF to extract type of income indexes for each page in a statement, we set the Type to Page. The page indexes are stored with the AFP document. A page index cannot be used to break a group. The Fields area lists the field parameters that have been defined for the report. We need to identify the field parameter that ACIF uses to locate the index. For the type of income index values, that is Field4. Select Field4 in the Fields list and click Add. OnDemand moves Field4 from the Fields list to the Order list.

Click OK to save the index information and return to the report window.

Defining the subtotal index

To define an index, first clear any selected triggers or fields by clicking a blank area of the report. Then click the Add an Index icon on the toolbar. OnDemand displays the Add an Index dialog box.

The Identifier (Index5) determines the name of the index parameter. The Attribute is the name of the index. For AFP documents, OnDemand displays the attribute names and values of page indexes in the Go To dialog box, allowing users a way to navigate pages of a statement. Attribute names should be meaningful to the user. Enter `Subtotal`. The Type of Index determines the type of index ACIF generates. Since we want ACIF to extract subtotal indexes for each page in a statement, we set the Type to Page. The page indexes are stored with the AFP document. A page index cannot be used to break a group. The Fields area lists the field parameters that have been defined for the report. We need to identify the field parameter that ACIF uses to locate the index. For the subtotal index values, that is Field5. Select Field5 in the Fields list and click Add. OnDemand moves Field5 from the Fields list to the Order list.

Click OK to add the index and return to the report window.

ACIF parameters

If we were to click the Display ACIF Parameters icon on the toolbar, the graphical indexer would display a window that contains the indexing parameters. We can see the result of defining the indexes. Note the index parameters with the options and data values we specified in the Add an Index dialog box.

Close the Display ACIF Parameters dialog box.

Displaying triggers, fields, and indexes

Click the Display and Add Parameters icon on the toolbar to verify the indexing information. When you click the icon, the graphical indexer changes to display mode (note the status bar). The triggers and fields appear highlighted. Scroll through pages of the report to verify that they appear in the correct location on all pages. When you have finished, toggle the Display and Add Parameters icon to add mode.

Click the Select Trigger, Index, Field Parameters icon on the toolbar. The Select dialog box appears. Using this dialog box, you can display and maintain trigger, index, and field information. For example, click Field1. OnDemand highlights the area of the report where we defined the field. Click Field1 again. OnDemand highlights the area on the next page. Click Field2. OnDemand highlights the field in the report. Click Index1 and then click

Properties. OnDemand displays the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties. OnDemand displays the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

ACIF Indexer Properties

After defining the View Information, triggers, fields, and indexes, we can complete the indexing information for the income statement report by setting values in the ACIF Indexer Properties dialog box. This is a central place to maintain parameters that describe the format of the data, resources required to view and print the data, indexes generated by ACIF, and optional user-defined exit programs to process input, output, and index records and resources. Many of the parameter values are based on the choices made on the View Information page and trigger, field, and index definitions. We won't go over every field on every page; you can click Help on each page to display information about the fields. You can also refer to "Chapter 3. ACIF Parameter Reference" on page 61 for details.

Review the parameter values on the Data Format page. Since we want to store the input line data report in OnDemand as AFP data (to support the page-level indexes), we must set Data Conversion to Yes. When we do this, the administrative client automatically changes the Data Type to AFP on the View Information page. The file format and carriage control areas retain the original settings we made on the View Information Page. We don't need to make any changes to the values in the Font Information area.

Click the Resource Information tab. Since we plan to convert the input data to AFP, we must specify values on this page. For the income statement report, we specify the name of a form definition and page definition. We want ACIF to collect form definitions, so we check the appropriate box in the Resource File Contents area. Finally, in the Search Paths area, we identify where ACIF can find the resources. Enter the name of the directories where the resources reside. For example, in the Form Definitions field, we entered /usr/lpp/psf/res/fdeflib.

You can review the parameter values on the Output Information page (although we don't need to change any of them).

We're not providing user exits for ACIF to process the data, index, or resources. Therefore, we don't need to specify values on the Exit Information page.

Click OK to save the changes and return to the report window. Since we're finished defining indexing information for the report, we can close the report window. OnDemand asks us if we want to save our changes. Click Yes. We return to the Indexer Information page.

Defining the application — part 2

View Information

The View Information page is where we specify information needed by OnDemand client programs to display the income statements.

We initially set the Data Type to Line to prepare the indexer information. After generating the indexing parameters, we set Convert to Yes (in the ACIF Indexer Parameters dialog box). When we did that, the administrative client automatically set the Data Type to AFP, which is the format of the data as stored in OnDemand.

Indexer parameters

The Indexer Parameters area now contains all of the indexing parameters required for ACIF to process the income statement report. Use the scroll bars to review the parameters.

Load Information

The Load Information page is where we map the index attribute names we defined to hold the attribute values ACIF extracts from the report to application group database field names. For the sample income statement report, we named the attributes the same as the database fields. Therefore, we do not need to map the attributes names on the Load Information page. To verify this, select each name in the Application Group DB Name list. The corresponding index attribute name appears in the Load ID Name field. The names should be the same.

The Load Information page also contains other values used when OnDemand stores index data in the database. For example:

- If the appearance of the date field in the report is different than the default date format for the application, you can specify the correct date format found in the report. Verify the date format for the billing date field. Select `sdate` in the Application Group DB Name list. The Format field should contain the format specifier that describes the appearance of the date in the report (%m/%Y, for the MM/YYYY format dates found in the report). If it does not, select the format specifier that correctly describes the appearance of the date in the report.
- If the field contains special characters, you can specify that you want OnDemand to remove them from the data before storing index values in the database. For decimal fields, such as the Total Income and Subtotal, you probably want to remove the blank, $ (dollar), **,** (comma), and **.** (decimal) characters from the data. To do so, select the field in the Application Group

DB Name list. In the Embedded field, enter the comma and period characters. In the Leading field, enter the blank and dollar characters.

Adding the application

We've completed the minimum requirements for adding an application to the database, including defining the indexer information. Click OK to add the application to OnDemand and return to the Administrative Tasks window.

Example four

About the report

This example describes how to create indexing information for an input AFP file that contains Tagged Logical Element (TLE) structured fields. The TLEs in the file contain group-level and page-level index tags. The group-level index information is stored in the database and used to search for and retrieve the file. The page-level index information is stored with the file. The page identifiers can be used to navigate pages of the file with one of the OnDemand client programs. The IBM Document Composition Facility (DCF) is an example of an application that can create AFP files that contain TLE structured fields. The OnDemand publications are examples of AFP files generated by DCF that contain TLE structured fields. Figure 15 on page 56 shows a page of the document viewed with one of the OnDemand client programs.

For faster retrieval, we plan to store the file in OnDemand large objects in groups of 20 pages. Figure 16 on page 57 shows the indexer parameters required for ACIF to process the AFP file.

Accessing the sample report

Because we don't need to specify TRIGGER, FIELD, and INDEX parameters for the sample document, we are not going to process the document with the graphical indexer.

Contents

Figure 15. AFP Document

```
/* DATA INPUT/OUTPUT CHARACTERISTICS                                    */
CC=YES                                  /* carriage controls present   */
CCTYPE=A                                /* ANSI controls in EBCDIC     */
CONVERT=YES                             /* AFP data in OD              */

/* TRIGGER/FIELD/INDEX DEFINITIONS                                      */
/* None when ACIF processes AFP input data containing TLEs             */

/* INDEXING INFORMATION                                                 */
DCFPAGENAMES=YES                        /* unique page names           */
UNIQUEBNQS=YES                          /* unique group names          */
IMAGEOUT=ASIS                           /* leave images alone          */
INDEXOBJ=ALL                            /* required for large object   */

/* RESOURCE INFORMATION                                                 */
FORMDEF=F1A10110                        /* default formdef             */
USERLIB=/usr/lpp/ars/pubs/reslib        /* resource library            */
RESTYPE=NONE                            /* do not collect resources    */
```

Figure 16. ACIF Parameters

Key concepts

Large Object. Provides enhanced usability and better retrieval performance for reports that contain very large logical items (for example, statements that exceed 500 pages) and files that contain lots of images, graphics, fonts, and bar codes. OnDemand segments data into groups of pages, compressed inside a large object. You determine the number of pages in a group. When the user retrieves an item, OnDemand retrieves and uncompresses the first group of pages. As the user navigates pages of the item, OnDemand automatically retrieves and uncompresses the appropriate groups of pages. To enable large object support, you must index the input data with ACIF and specify INDEXOBJ=ALL.

Page Identifiers. For AFP data, identifies each page in a report and provides another way to navigate pages of a report. Typically extracted from page-level TLEs contained in the document.

Convert. Determines the type of output produced by ACIF. When you process AFP input data with ACIF, you must specify CONVERT=YES.

Resources. Objects required to load, view, and print AFP data. If the input data is AFP, you must specify resources and resource paths, even if the input data contains all of the required resources.

Defining the application — part 1

An application identifies the type of data that is stored in OnDemand, the method used to index the data, and other information that OnDemand uses to load and view reports. This topic provides information about the application we created to process the sample AFP document.

General

The General page is where we name the application and assign the application to an application group.

We assigned the application to the application group where we plan to store AFP documents. The application group contains database field definitions for the document date and number.

View Information

The View Information page is where we specify information needed by OnDemand client programs to display the documents. This information is also used by the indexing program.

Since the documents will be stored in OnDemand as AFP data, we set the Data Type to AFP.

Indexer Information

The Indexer Information page is where we specify ACIF as the Indexer and create additional ACIF parameters, including those that identify the format of the AFP document, the type of indexes ACIF generates, and the resources ACIF requires to process the document. We will create the parameters using the keyboard option on the Indexer Information page.

Creating ACIF parameters

Make sure that the Add an Application window is turned to the Indexer Information page. In the Parameter Source area, select Keyboard. Click Modify. OnDemand opens the ACIF Parameters window, a standard edit window where you can enter, modify, and delete ACIF parameters.

Note: OnDemand does not verify the parameters, options, or data values that you enter in the ACIF Parameters window.

Defining the data format

We need to add the following parameters that describe the format of the AFP document. Whenever you process input data with ACIF and store AFP data in OnDemand, you must specify CONVERT=YES.

- `CC=YES`
- `CCTYPE=A`
- `CONVERT=YES`

Defining indexing information

We need to add the following parameter that describes the type of index information ACIF generates. This parameter causes ACIF to generate page-level indexes (in addition to group-level indexes). OnDemand uses the page-level indexes to determine the object that contains the page the user wants to view. Because we plan to store the document as an OnDemand large object, we must specify INDEXOBJ=ALL.

- `INDEXOBJ=ALL`

In addition, because the sample AFP document was generated by DCF, we need to change the value of the DCFPAGENAMES parameter:

- `DCFPAGENAMES=YES`

Defining resource information

We need to add the following parameters so that ACIF can process the AFP document. These are the minimum parameters required by ACIF to process AFP data. We specify the name of a standard form definition and the directory that contains resources. We also specify that we do not want ACIF to collect resources (our sample AFP document contains the required resources).

- `FORMDEF=F1A10110`
- `USERLIB=/usr/lpp/ars/pubs/reslib`
- `RESTYPE=NONE`

We're finished defining indexing information for the AFP document. Close the ACIF Parameters window. OnDemand asks us if we want to save our changes. Click Yes. We return to the Indexer Information page.

Defining the application — part 2

Indexer parameters

The Indexer Parameters area now contains all of the indexing parameters required for ACIF to process the AFP document. Use the scroll bars to review the parameters.

Load Information

The Load Information page is where we map index attribute names to application group database field names. For the sample AFP document, we named the attributes the same as the database fields. Therefore, we do not need to map the attributes names on the Load Information page. To verify this, select each name in the Application Group DB Name list. The corresponding index attribute name appears in the Load ID Name field. The names should be the same.

The Load Information page also contains other values used when OnDemand stores index data in the database. For example, if the appearance of the date field in the document is different than the default date format for the application, you can specify the correct date format found in the report. Verify the date format for the document date field. Select `pubdate` in the Application Group DB Name list. The Format field should contain the format specifier that describes the appearance of the date in the document. If it does not, select the format specifier that correctly describes the appearance of the date in the document.

Adding the application

We've completed the minimum requirements for adding an application to the database, including defining the indexer information. Click OK to add the application to OnDemand and return to the Administrative Tasks window.

Chapter 3. ACIF Parameter Reference

This parameter reference assumes that you will run ACIF on an OnDemand server. If you run ACIF on an MVS or OS/390 system, the defaults for certain parameters and values change from ASCII to EBCDIC. Refer to the CCTYPE (page 62) and CPGID (page 64) parameters, the MASK subparameter of the TRIGGER parameter (page 99), and the USERMASK parameter (page 104) for details.

This parameter reference also assumes that you will use OnDemand data loading programs (arsload and arsadmin) to process files. When you use OnDemand data loading programs to process files, the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, PARMDD, and RESOBJDD parameters are ignored. If you run ACIF from the command prompt or on an MVS or OS/390 system, verify the values of the INDEXDD (page 80), INPUTDD (page 83), MSGDD (page 86), OUTPUTDD (page 88), PARMDD (page 91), and RESOBJDD (page 96) parameters.

CC

Required	Default Value	Data Type	
		AFP	Line
No	YES	X	X

Determines whether the input contains carriage-control characters.

Generic syntax

CC=*value*

Options and values

The *value* can be:

YES

The input contains carriage control characters. When the input data is AFP, you should set CC=YES and CCTYPE=A.

NO

The input does not contain carriage control characters.

Related parameters

CCTYPE parameter on page 62.

CCTYPE

Required	Default Value	Data Type	
		AFP	Line
No	Z	X	X

If the data contains carriage control characters, determines the type of carriage-control characters. ACIF supports ANSI carriage-control characters in either ASCII or EBCDIC and machine code carriage-control characters. ACIF does not allow a mixture of ANSI and machine carriage-control characters within a file. If you specify CC=YES and you do not specify the CCTYPE parameter, ACIF assumes that the input contains ANSI carriage-control characters encoded in ASCII. If you are running ACIF on an MVS or OS/390 system, ACIF assumes that the carriage-controls are encoded in EBCDIC.

Generic syntax

CCTYPE=*value*

Options and values

The *value* can be:

Z

The input contains ANSI carriage-control characters that are encoded in ASCII. The carriage-control characters are the ASCII values that directly relate to ANSI carriage-controls, which cause the action of the carriage-control character to occur *before* the line is printed.

A

The input 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. If the input data is AFP, you should set CCTYPE=A and CC=YES.

M

The input contains machine code carriage-control characters. 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.

Related parameters

CC parameter on page 61.

CHARS

Required	Default Value	Data Type	
		AFP	Line
No		X	

When converting line data to AFP and the input data contains TRCs, the CHARS parameter is required if the specified page definition does not name a font. The CHARS parameter identifies from one to four fonts referenced in the data. If the fonts will be saved in a resource group, the CHARS parameter also provides the names of the fonts ACIF saves in the resource group. The CHARS command can also be used to specify the font used for the entire report when the input data does not contain TRCs and the specified page definition does not name a font.

Generic syntax

CHARS=*fontlist*

Options and values

The *fontlist* is a comma-separated string of one to four valid coded font names. For example:

```
CHARS=GT10,GT12,GT24
```

The font name is limited to four alphanumeric or national characters and should not include the two-character prefix of the coded font name (X0 through XG). For example, the coded font X0GT10 would be specified as GT10. On UNIX servers, the font name is case sensitive. If you use the ASCII fonts that are supplied with PSF for AIX, use the font names listed in “Chapter 6. ACIF and the IBM AFP Fonts for ASCII Data” on page 165 (excluding the two-character prefix).

Related parameters

PAGEDEF parameter on page 89.

CONVERT

Required	Default Value	Data Type	
		AFP	Line
No	YES	X	X

Determines whether ACIF converts the input data to AFP. If the input data is line data and you need to generate page-level indexes, you must convert the line data to AFP. Otherwise, OnDemand cannot write the page-level index information to the output file.

Generic syntax

CONVERT=*value*

Options and values

The *value* can be:

YES

ACIF converts the input data to AFP. If the input data is AFP, the CONVERT parameter is optional, but the default is YES.

NO

ACIF does not convert the input data to AFP.

CPGID

Required	Default Value	Data Type	
		AFP	Line
No	850 (ASCII) 500 (EBCDIC)	X	X

Identifies the code page of the index data. Typically, the CPGID is the same as the code page of the input data.

Generic syntax

CPGID=*value*

Options and values

The *value* can be:

850

The default IBM code page.

code page identifier

Any valid code page. A three to five character identifier of an IBM-registered or user-defined code page.

DCFPAGENAMES

Required	Default Value	Data Type	
		AFP	Line
No	NO	X	X

Determines whether ACIF generates page names using an eight-byte counter or uses structured field tokens found in the input data stream.

Generic syntax

DCFPAGENAMES=*value*

Options and values

The *value* can be:

NO

ACIF generates page names using an eight-byte counter.

YES

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

FDEFLIB

Required	Default Value	Data Type	
		AFP	Line
No		X	

Identifies directories in which form definitions are stored. Specify any valid search path. ACIF searches for the form definition in the following order:

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the FDEFLIB parameter, if any.
3. The paths you specified with RESLIB parameter, if any.
4. On AIX servers, the paths you specified with the PSFPATH environment variable (if it is set).
5. On AIX servers, the directory /usr/lpp/psf/reslib (if it exists).

Generic syntax

FDEFLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
FDEFLIB=/tmp:/usr/resources:/usr/lpp/ars/fdeflib
```

ACIF searches the paths in the order specified. Delimit UNIX path names with the colon (:) character. Delimit NT path names with the semicolon (;) character.

Related parameters

RESLIB parameter on page 95.

USERLIB parameter on page 103.

FIELD

Required	Default Value	Data Type	
		AFP	**Line**
Yes		X	X

Identifies the location of index data and can provide default and constant index values. You must define at least one field. You can define up to 32 fields. There are three types of fields. A *trigger field* is based on the location of a trigger string value. A *constant field* provides the actual index value stored in the database. A *transaction field* is used to index reports that contain one or more columns of sorted data and it is not practical to store every value in the database (ACIF extracts the beginning and ending sorted values in each group).

Trigger field syntax

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

Options and values

n

The field parameter identifier. When adding a field parameter, use the next available number, beginning with 1 (one).

record

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

The relative column number from the BASE. This is the column number where ACIF begins to search for the field. A value of 1 (one) 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 one), consider defining the value of the carriage-control character as one of the TRIGGER parameters. If you specify BASE=0, the *column* value can be 1 to 32756. If you specify BASE=TRIGGER, the *column* value can be –32756 to 32756. If the specified value exceeds the physical length of the record, ACIF reports an error condition and terminates processing.

length

The number of contiguous bytes (characters) that compose the field. The supported range of values are 1 to 250. The field can extend outside the record length, if the column where it begins lies within the record length. In this case, ACIF adds padding blanks to fill out the record. If the field begins outside the maximum length of the record, ACIF reports an error condition and terminates processing, unless you specify a DEFAULT value.

TRIGGER=*n*

Identifies the trigger parameter ACIF uses to locate the field. This is an optional parameter, but the default is TRIGGER1. Replace *n* with the number of a defined TRIGGER parameter.

BASE={**0** | **TRIGGER**}

Determines whether ACIF uses the starting column number of the trigger string value to locate the field data. Choose from 0 (zero) or TRIGGER. If BASE=0, ACIF adds zero to the field column offset. If BASE=TRIGGER, ACIF adds the starting column number of the trigger string value to the field column offset. You should use BASE=0 if the field data always starts in a specific column. You should use BASE=TRIGGER if the field data doesn't always start in a specific column, but is always offset from the trigger string value a specific number of columns. For example, a trigger occurs in the second record on a page. The trigger string value can begin in any column in the record. A field based on this trigger occurs in the trigger record. The starting column number of the field is always ten bytes from the starting column number of the trigger. Specify BASE=TRIGGER and a column offset of ten so that ACIF correctly locates the field, regardless of the starting column of the trigger string value.

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

Defines the default index value, when a record is not long enough to contain the field data. Specify the value in hexadecimal.

The DEFAULT keyword supports the use of MVS Download and Download for OS/390, products that can be used to transmit data from the JES Spool to OnDemand servers. To conserve space and reduce transmission times, Download truncates records that contain one or more

blank characters at the end of the record. For example, a report contains fixed length records, each 80 bytes in length. Columns 77 through 80 of the records contain audit data generated by the application program. If a record has not been audited, the columns contain blanks. During file transmission, Download eliminates the blanks from the end of the records. You want to store a default value in the database for unaudited records. Given the following definition:

```
FIELD2=1,77,4,(DEFAULT=X'D5D6D5C5')
```

ACIF assigns the index associated with FIELD2 the value D5D6D5C5 (NONE), if a record is not 77 bytes in length.

Examples

The following field parameter 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. We specify BASE=0 because the field data always starts in the same column.

```
TRIGGER1=*,1,X'F1',(TYPE=GROUP)
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
```

The following field parameter causes ACIF to locate field values that begin ten columns offset from the trigger string value. The trigger string value can start in any column in any record. Basing the field on TRIGGER2 and specifying BASE=TRIGGER allows ACIF to locate the field by adding ten to the starting column offset of the trigger string value.

```
TRIGGER2=*,*,X'E2A482A396A38193',(TYPE=FLOAT)
FIELD2=0,10,12,(TRIGGER=2,BASE=TRIGGER)
```

Constant field syntax

FIELD*n*=*constant*

Options and values

n

The field parameter identifier. When adding a field parameter, use the next available number, beginning with 1 (one).

constant

The literal (constant) string value of the field. This is the index value stored in the database. If the input data contains unformatted ASCII data, the constant can be specified either as character data or hexadecimal data. Specify a hexadecimal value using the format X'*constant*', where *constant* is hexadecimal data. If the input data contains EBCDIC data, the constant must be specified as hexadecimal data. The constant value can be 1 to 250 bytes in length. ACIF does not validity check the actual content of the supplied data.

Examples

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

```
FIELD3=X'F0F0F0F0F0F0F0F0'
INDEX3=X'D5D6D6D7',FIELD3,(TYPE=GROUP,BREAK=NO)
```

The following field parameters cause ACIF to concatenate a constant value with the index value extracted from the data. ACIF concatenates the constant 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=X'F0F0F0F060'
FIELD4=0,66,14
INDEX3=X'818383A36D95A494',FIELD3,FIELD4,(TYPE=GROUP,BREAK=YES)
```

Transaction field syntax

FIELD*n*=*****,*****,***length*,**(OFFSET=(***start1*:*end1*[,*...start8*:*end8*]),**MASK='***@#=¬^%'*
[,**ORDER=**{**BYROW** | **BYCOL**}])

Options and values

n

The field parameter identifier. When adding a field parameter, use the next available number, beginning with 1 (one).

*****

The record number where ACIF begins searching for the field. A transaction trigger must specify an asterisk, meaning ACIF searches every record in the group.

*****

The column number where ACIF begins searching for the field. A transaction trigger 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 the OFFSET keyword of the FIELD parameter, ACIF always uses the starting column number(s) from the OFFSET keyword to determine the location of the field value(s).

length

The number of contiguous bytes (characters) that compose the field. The supported range of values are 1 to 250. The field can extend outside the record length, if the column where it begins lies within the record length.

In this case, ACIF adds padding blanks to fill out the record. If the field begins outside the maximum length of the record, ACIF reports an error condition and terminates processing.

OFFSET=(*start***:***end***)**

Determines 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 the OFFSET keyword, you must also specify the MASK keyword. The implied length of an OFFSET must be the same as the number of characters in the MASK or ACIF will not detect a match.

MASK='*@#=¬^%'

Determines the pattern of symbols that ACIF matches with data located in the field columns. When you specify the MASK keyword, you must also specify the OFFSET keyword. 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 pagegrange indexes. Valid mask symbols include the following:

***** Not literal; matches a user-defined mask. Refer to the USERMASK (page 104) parameter.

@ Matches alphabetic characters.

Matches numeric characters.

¬ Matches any non-blank character.

^ Matches any non-blank character.

% Matches the blank character and numeric characters.

= Matches any character.

Code page 850 is the default code page for the symbols in the MASK. If you specify a different code page (on 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. For example, the following definitions:

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

Cause ACIF to search columns ten through seventeen for a hexadecimal C1 followed by four numeric characters (hexadecimal F0-F9), a hexadecimal 60, and two numeric characters (hexadecimal F0-F9).

ORDER={**BYROW** | **BYCOL**}

Identifies 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. The default ORDER is BYROW.

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 may appear as follows:

```
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 may appear as follows:

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

Examples

The following field parameter causes ACIF to locate a ten 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)
```

Related parameters

CPGID parameter on page 64.

INDEX parameter on page 76.

TRIGGER parameter on page 99.

FILEFORMAT

Required	Default Value	Data Type	
		AFP	**Line**
No	STREAM	X	X

Identifies the format of the input file.

Generic syntax

FILEFORMAT={**STREAM**[**,(NEWLINE=X'***value***')**] | **RECORD**[**,***n*]}

Options and values

The values are:

STREAM[**,(NEWLINE=X'***value***')**]

The input file has no length information; it is a stream of data separated by a newline character. Files with STREAM format typically come from a workstation operating system such as AIX, HP-UX, OS/2, Solaris, or Windows NT.

The NEWLINE keyword identifies the hexadecimal character that delimits records in the data stream. If the NEWLINE keyword is not specified, ACIF examines the first six bytes of the first record in the input file, to determine whether the file is ASCII or EBCDIC. If ACIF determines that the input file is ASCII, it looks for the ASCII newline character (X'0A') to delimit the end of a record. If ACIF determines that the input file is EBCDIC, it looks for the EBCDIC newline character (X'25') to delimit the end of a record.

RECORD[**,***n*]

The input file is formatted in System/370 record format, where the first two bytes of each line specify the length of the line. RECORD format files typically are MVS files that have a variable record format.

For RECORD,*n* files, the input file is formatted in such a way that each record is fixed length, *n* bytes long. The value of *n* is a number from 1 to 32767.

FONTLIB

Required	Default Value	Data Type	
		AFP	Line
No		X	

Identifies the directories in which fonts are stored. Specify any valid search path. ACIF searches for the fonts in the following order:

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the FONTLIB parameter, if any.
3. The paths you specified with the RESLIB parameter, if any.
4. On AIX servers, the paths specified by the PSFPATH environment variable, if it is set.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists.
6. On AIX servers, the directory /usr/lpp/afpfonts, if it exists.
7. On AIX servers, the directory /usr/lpp/psf/fontlib, if it exists.

Generic syntax

FONTLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
FONTLIB=/tmp:/usr/resources:/usr/lpp/ars/fontlib
```

ACIF searches the paths in the order in which they are specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

Related parameters

RESLIB parameter on page 95.

USERLIB parameter on page 103.

FORMDEF

Required	Default Value	Data Type	
		AFP	**Line**
Yes		X	

Determines the file name of the form definition. A form definition defines how a page of data is placed on a form, the number of copies of a page, any modifications to that group of copies, the paper source, and duplexing. ACIF requires a form definition to process an AFP file or a line data file being converted to AFP.

The form definition can be located inline in the file. To use an inline form definition you must do the following:

- If you specify the FORMDEF parameter, the name of the inline form definition must match the name of the specified form definition, or you must specify FORMDEF=DUMMY. If the name specified for the FORMDEF parameter does not match the name of an inline form definition, ACIF looks for the form definition in the FDEFLIB search path. If you specify FORMDEF=DUMMY and there is no inline form definition, ACIF searches the FDEFLIB search path for a form definition named DUMMY. ACIF reports an error and stops if unable to locate the form definition.
- If a form definition resource is included inline with the data, the file must be identified as containing carriage control characters (you must specify CC=YES).

- 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:
  - On UNIX and Windows NT servers, specify FILEFORMAT=RECORD,n (where n is the logical record length defined for the file).
  - In MVS, specify record format FBA (fixed block with ANSI carriage control characters or FBM (fixed block with machine carriage control characters).
- 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:
  - On UNIX and Windows NT servers, specify FILEFORMAT=RECORD. The first two bytes of each record determine the record length.
  - In MVS, specify record format VBA (variable blocked with ANSI carriage control characters or VBM (variable blocked with machine carriage control characters).

Generic syntax

FORMDEF=*fdefname*

Options and values

The *fdefname* is the name of the form definition, one to eight alphanumeric or national characters, including the two-character prefix, if there is one. On UNIX servers, the *fdefname* is case sensitive. Specify the file name of the form definition, not the file type. However, the file type must be FDEF3820, FDEF38PP, or FDE (or no file type).

Related parameters

FDEFLIB parameter on page 65.

GROUPMAXPAGES

Required	Default Value	Data Type	
		AFP	Line
No		X	X

Determines the maximum number of pages that ACIF puts into a group. Allows ACIF to logically segment a large report into groups of pages and create indexes for each group. You can specify a number from 1 and 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 the

GROUPMAXPAGES parameter, ACIF does not terminate the current group and begin a new group until the value of one of the fields named by an INDEX with BREAK=YES 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.

Generic syntax

GROUPMAXPAGES=*value*

Options and values

The *value* is the number of pages ACIF puts in a group. Enter a number from 1 to 9999.

Related parameters

INDEX parameter on page 76.

GROUPNAME

Required	Default Value	Data Type	
		AFP	**Line**
No	INDEX1	X	

Determines which of the 32 possible index values should be used as the group name for each index group. If you do not specify the GROUPNAME parameter, ACIF uses the value of the INDEX1 parameter. Using the most unique index value for the group name is recommended. The intent is to have a unique group name for every group ACIF produces. The value includes the FIELD definitions from the INDEX parameter but does not include the attribute name. OnDemand displays the value along with the attribute name and index value. After retrieving a document from the server, users can use the group name to select and display a specific group of pages.

Note: When defining the group name, a FIELD cannot be based on a float or recordrange trigger.

Generic syntax

GROUPNAME=*indexParameter*

Options and values

The *indexParameter* can be:

INDEX1

ACIF uses the INDEX1 parameter to determine the group name. INDEX1 is the default.

INDEX*n*

ACIF uses the specified INDEX parameter to determine the group name.

Related parameters

INDEX parameter on page 76.

IMAGEOUT

Required	Default Value	Data Type	
		AFP	Line
No	IOCA	X	X

Determines the format of the image data produced by ACIF.

Generic syntax

IMAGEOUT=*value*

Options and values

The *value* can be:

IOCA

ACIF converts image data to uncompressed Image Object Content Architecture (IOCA) format.

ASIS

ACIF passes all image data through unconverted. We recommend that you select ASIS to reduce the size of the output file and to improve ACIF performance.

INDEX

Required	Default Value	Data Type	
		AFP	Line
Yes		X	X

Identifies 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. ACIF can also generate page indexes for line data that you store in OnDemand large objects, if you set the INDEXOBJ parameter to ALL. You must define at least one index parameter. You can define up to 32 index parameters. When you define a group index, we strongly encourage you to name the index the same as the application group database field name.

Important: 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 navigate pages of the document with the Go To command.

Generic syntax

INDEXn=*name*,**FIELD**nn[,...**FIELD**nn][,**(TYPE=***type***)**]

Options and values

n

The index parameter identifier. When adding an index parameter, use the next available number beginning with 1 (one).

name

Determines the index name associated with the actual index value. For example, assume INDEX1 is to contain account numbers. The string *acct_num* would be a meaningful index name. The index value of INDEX1 would be an actual account number, for example, 000123456789. The index name can be a maximum of 250 bytes in length.

The index name can be specified either as character data or hexadecimal data. If the input file is **anything other than** ASCII, then the index name **must** be specified as hexadecimal data. Specify a hexadecimal value using the format X'name', where name is hexadecimal data, for example, X'95819485'.

FIELD*nn*

The name of the field parameter or parameters ACIF uses to locate the index. A maximum of 32 field parameters can be specified for each index. Separate field parameter names with a comma. The total length of all the specified field parameters cannot exceed 250 bytes.

GROUPRANGE and PAGERANGE indexes must name one and only one transaction field. PAGE indexes must name fields based on floating triggers.

GROUPRANGE, PAGE, and PAGERANGE indexes cannot break a group – you must specify BREAK=NO.

An index that names a field based on a floating trigger must be TYPE=GROUP or TYPE=PAGE and must specify BREAK=NO.

TYPE=*type*

The type of index ACIF generates. You can define group indexes for AFP and line data. You can define page indexes for AFP data. The default index type is GROUP. Valid index types are:

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

Create a group index value. ACIF creates one index value for each group.

You can specify whether ACIF includes or ignores the index when calculating a group break. When BREAK=YES (the default), ACIF begins a new group when the index value changes. For most reports, break should always be set to yes. BREAK=NO is useful when you define two or more indexes and you want ACIF to begin a new group only when a specific index value changes. Specify BREAK=YES for the index that you want ACIF to use to control the group break. Specify BREAK=NO for the other indexes.

A GROUP index that names a field parameter based on a floating trigger must specify BREAK=NO.

TYPE=GROUPRANGE,BREAK=NO

Create group indexes. 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. A GROUPRANGE index cannot name a field parameter that is based on a floating trigger. A GROUPRANGE index 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. 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.

TYPE=PAGE,BREAK=NO

For AFP output data, create zero or more page indexes per page. Page indexes must name fields based on floating triggers. Page indexes cannot be used to break a group – you must specify BREAK=NO.

Page indexes are stored with the document, not in the database, and cannot be used to search for documents. After retrieving the document, you can use the page indexes to navigate pages of the document with the Go To command.

When you define a PAGE index, you must also set the INDEXOBJ parameter to ALL. Otherwise, ACIF will not write the page index data to the index object file.

TYPE=PAGERANGE,BREAK=NO

For AFP output data, create page indexes. 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.

PAGERANGE indexes cannot be used to break a group – you must specify BREAK=NO.

PAGERANGE indexes must name one and only one transaction field. PAGERANGE indexes cannot name a field parameter that is based on a floating trigger.

Page indexes are stored with the document, not in the database, and cannot be used to search for documents. After retrieving the document, you can use the page indexes to navigate pages of the document with the Go To command.

When you define a PAGERANGE index, you must also set the INDEXOBJ parameter to ALL. Otherwise, ACIF will not write the pagerange index data to the index object file.

Examples

Group index

The following index parameter causes ACIF to generate group indexes for date index values. The input data is encoded in EBCDIC. The index type is optional, but defaults to group. When the index value changes, ACIF closes the current group and begins a new group.

```
INDEX1='6C6F61645F64617465',FIELD1,(TYPE=GROUP,BREAK=YES)
```

The following index parameters cause ACIF to generate group indexes for customer name and account number index values. The input data is encoded in EBCDIC. The index type is optional, but defaults to group. 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 may 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='95819485',FIELD1,(TYPE=GROUP,BREAK=YES)
INDEX2='818383A46D95A494',FIELD2,(TYPE=GROUP,BREAK=NO)
```

Grouprange index

The following index parameter causes ACIF to generate 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. A grouprange index must be based on a transaction field. 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='4C6F616E204E756D626572',FIELD2,(TYPE=GROUPRANGE,BREAK=NO)
GROUPMAXPAGES=100
```

Page index

The following index parameter causes ACIF to generate page indexes for subtotal values (the attribute name that appears in the Go To dialog box is Subtotal). The input data (AFP or line data converted to AFP) is encoded in EBCDIC. ACIF extracts the index values from each page. A page index must name a field that is based on a floating trigger. A page index cannot be used to break a group.

```
INDEX3='E2A482A396A38193',FIELD3,(TYPE=PAGE,BREAK=NO)
```

Related parameters

FIELD parameter on page 66.

INDEXOBJ parameter on page 81.

INDEXDD

Note: This parameter is ignored when you use the OnDemand data loading programs (arsload and arsadmin) to process files.

Required	Default Value	Data Type	
		AFP	**Line**
No	INDEX	X	X

Determines the name or the full path name of the index object file, where ACIF writes indexing information. If you specify the file name without a path, ACIF puts the index object file in the current directory. If you do not specify the INDEXDD parameter, ACIF writes indexing information to the file INDEX.

Generic syntax

INDEXDD=*filename*

Options and values

The *filename* is a valid filename or full path name.

INDEXOBJ

Required	Default Value	Data Type	
		AFP	**Line**
No	GROUP	X	X

Determines the level of indexes ACIF includes in the index object file.

Generic syntax

INDEXOBJ=*value*

Options and values

The *value* can be:

GROUP

ACIF includes group-level index entries in the index object file.

Note: If you define page-level indexes and specify INDEXOBJ=GROUP, OnDemand will not be able to write the page-level index data.

ALL

ACIF includes group-level and page-level indexes in the index object file. You must specify INDEXOBJ=ALL for reports that require page-level indexes or OnDemand large object support. The source data must be AFP or line data that you plan to convert to AFP with ACIF.

NONE

ACIF does not create an index object file. Specify none only when you do not want to index the input file.

Related parameters

INDEX parameter on page 76.

INDEXSTARTBY

Required	Default Value	Data Type	
		AFP	Line
No	1	X	X

Determines the page number by which ACIF must find an indexing field. ACIF fails if it does not find an indexing field before the specified page number. This parameter is optional, but the default is that ACIF must find an index on the first page.

This parameter is helpful if the input file contains header pages. For example, if the input file contains two header pages, you can specify a page number one greater than the number of header pages (INDEXSTARTBY=3) so that ACIF will not start indexing until the page after the header pages.

Note: When you use INDEXSTARTBY to skip header pages, ACIF copies the non-indexed pages to the output file. OnDemand stores the non-indexed pages in the application group. For example, if you specify INDEXSTARTBY=3 and ACIF finds the first index on page three, ACIF copies pages one and two to the output file and OnDemand stores them in the application group.

Generic syntax

INDEXSTARTBY=*value*

Options and values

The *value* is the page number of the report by which ACIF must find an indexing field.

INDXEXIT

Required	Default Value	Data Type	
		AFP	Line
No		X	X

Identifies the name or the full path name of the index record exit program. This is the program ACIF calls for every record (line or structured field) it writes in the index object file. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to “Chapter 5. ACIF User Exits and Attributes of the Input Print File” on page 151.

Generic syntax

INDXEXIT=*name*

Options and values

The *name* is the file name or full path name of the index record exit program. On UNIX servers, the program name is case sensitive. If you specify the file name without a path, ACIF searches for the exit program in the paths specified by the PATH environment variable.

INPEXIT

Required	Default Value	Data Type	
		AFP	**Line**
No		X	X

Identifies the name or the full path name of the input record exit program. This is the program ACIF calls for every record (line) it reads from the input file. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to "Chapter 5. ACIF User Exits and Attributes of the Input Print File" on page 151.

Generic syntax

INPEXIT=*name*

Options and values

The *name* is the file name or full path name of the input record exit program. On UNIX servers, the program name is case sensitive. If you specify the file name without a path, ACIF searches for the exit program in the paths specified by the PATH environment variable.

INPUTDD

Note: This parameter is ignored when you use the OnDemand data loading programs (arsload and arsadmin) to process files.

Required	Default Value	Data Type	
		AFP	**Line**
No	stdin	X	X

Identifies the file name or full path name of the input file that ACIF will process. If you do not specify the INPUTDD parameter, ACIF uses standard input.

Generic syntax

INPUTDD=*filename*

Options and values

The *filename* is the file name or full path name of the input file to process. On UNIX servers, the program name is case sensitive. If you specify the file name without a path, ACIF searches in the current directory.

INSERTIMM

Required	Default Value	Data Type	
		AFP	Line
No	NO	X	

Determines whether ACIF inserts an IMM structured field before the first BPG structured field of every named page group.

Generic syntax

INSERTIMM=*value*

Options and values

The *value* can be:

NO

ACIF does not insert IMMs into the output data.

YES

ACIF inserts IMMs into the output data. Specify yes if the form definition names different overlays and multiple copy groups and switches copy groups any place other than on a group boundary. ACIF ensures that an IMM will be present within the named page group. However, ACIF does not guarantee that the correct overlay will be used, especially if the form definition uses enhanced *n*-up processing.

Note: The INSERTIMM parameter should be used with caution. It is helpful in viewing individual groups that require knowledge of the most recently used IMM. However, INSERTIMM=YES results in extra page advances when printing the output produced by ACIF.

Related parameters

The FORMDEF parameter on page 73.

LINECNT

Required	Default Value	Data Type	
		AFP	Line
No			X

For unconverted line data, determines the maximum number of lines per page. This parameter tells ACIF when to create page breaks. The LINECNT parameter is required when you specify CC=NO and CONVERT=NO. This parameter is ignored if CONVERT=YES. Note that page breaks also occur if Skip-to-Channel 1 carriage controls are present in the data.

Generic syntax

LINECNT=*number*

Options and values

The *number* is the maximum number of lines per page. ACIF creates a page break in the output file when this number is reached.

Related parameters

The CC parameter on page 61.

The CONVERT parameter on page 63.

MCF2REF

Required	Default Value	Data Type	
		AFP	Line
No	CPCS	X	

Determines the way that ACIF builds Map Coded Font 2 (MCF2) structured fields in the output file and the resource group file. ACIF can build MCF2 structured fields using coded font names or code page and character set names (the default). If you map the AFP fonts the report was created with to fonts that can be displayed on the PC, you must specify MCF2REF=CPCS (the default).

Generic syntax

MCF2REF=*value*

Options and values

The *value* can be:

CPCS

ACIF builds MCF2 structured fields using the names of the code page and character set by opening and reading the contents of all coded fonts specified in MCF1 and MCF2 structured fields in the input file or input resources. This is the default value. If you map the AFP fonts the report was created with to fonts that can be displayed on the PC, you must specify MCF2REF=CPCS.

CF

ACIF builds MCF2 structured fields using the name of the coded font. This option improves performance, because ACIF does not have to read the coded fonts from the font library.

Related parameters

The RESTYPE parameter on page 96.

MSGDD

Note: This parameter is ignored when you use the OnDemand data loading programs (arsload and arsadmin) to process files.

Required	Default Value	Data Type	
		AFP	**Line**
No	stderr	X	X

Determines the name or the full path name of the file where ACIF writes error messages. If you do not specify the MSGDD parameter, ACIF writes messages to standard error (UNIX) or the console (Windows NT).

Generic syntax

MSGDD=*filename*

Options and values

The *filename* is the file name or full path name where ACIF writes error messages. On UNIX servers, the file name is case sensitive. If you specify the file name without a path, ACIF places the message file in the current directory.

NEWPAGE

Required	Default Value	Data Type	
		AFP	**Line**
No	1		X

Identifies the skip-to-channel number that indicates a new page in the data stream. The NEWPAGE parameter is optional when you specify CC=YES and CONVERT=NO, but the default is 1 (one). You must specify the NEWPAGE parameter when the input is line data and you do not convert it to AFP and the skip-to-channel number is not 1 (one).

Generic syntax

NEWPAGE=*number*

Options and values

The *number* is the skip-to-channel number that indicates a new page in the data stream. Valid numbers are 1 (one) to 12 (twelve).

Related parameters

The CC parameter on page 61.

The CONVERT parameter on page 63.

OUTEXIT

Required	Default Value	Data Type	
		AFP	**Line**
No		X	X

Identifies the name or the full path name of the output record exit program. ACIF calls this program for every output record (every line or structured field) it writes to the output file. For more information about optional

program exits you can use to customize how ACIF handles input and output data, please refer to “Chapter 5. ACIF User Exits and Attributes of the Input Print File” on page 151.

Generic syntax

OUTEXIT=*name*

Options and values

The *name* is the file name or full path name of the output record exit program. On UNIX servers, the file name is case sensitive. If you specify the file name without a path, ACIF searches for the file name in the paths specified by the PATH environment variable.

OUTPUTDD

Note: This parameter is ignored when you use the OnDemand data loading programs (arsload and arsadmin) to process files.

Required	Default Value	Data Type	
		AFP	**Line**
No	stdout	X	X

Identifies the name or the full path name of the output file.

Generic syntax

OUTPUTDD=*name*

Options and values

The *name* is the file name or full path name of the output file. On UNIX servers, the file name is case sensitive. If you specify the file name without a path, ACIF puts the output file in the current directory.

OVLYLIB

Required	Default Value	Data Type	
		AFP	**Line**
No		X	

Identifies the directories in which overlays are stored. ACIF searches for an overlay in the following order:

1. The paths you specified with USERLIB, if any.
2. The paths you specified with OVLYLIB, if any.
3. The paths you specified with the RESLIB parameter, if any.
4. On AIX servers, the paths specified by the PSFPATH environment variable, if it is set.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists.

Generic syntax

OVLYLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
OVLYLIB=/tmp:/usr/resources:/usr/lpp/ars/ovlylib
```

ACIF searches the paths in the order specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

Related parameters

RESLIB parameter on page 95.

USERLIB parameter on page 103.

PAGEDEF

Required	Default Value	Data Type	
		AFP	Line
No		X	

Identifies the file name of the page definition. A page definition defines the page format that ACIF uses to compose the input file into pages. ACIF requires a page definition to convert an input file that contains System/370 line data, mixed-mode data, or unformatted ASCII data into AFP.

The page definition can be located inline in the file. To use an inline page definition you must do the following:

- If you specify the PAGEDEF parameter, the name of the inline page definition must match the name of the specified page definition, or you must specify PAGEDEF=DUMMY. If the name specified on the PAGEDEF parameter does not match the name of an inline page definition, ACIF looks for the page definition in the PDEFLIB search path. If you specify PAGEDEF=DUMMY and there is no inline page definition, ACIF searches

the PDEFLIB search path for a page definition named DUMMY. ACIF reports an error and stops if unable to locate the page definition.

- If a page definition resource is included inline with the data, the file must be identified as containing carriage control characters (you must specify CC=YES).
- 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:
  - On UNIX and Windows NT servers, specify FILEFORMAT=RECORD,n (where n is the logical record length defined for the file).
  - In MVS, specify record format FBA (fixed block with ANSI carriage control characters or FBM (fixed block with machine carriage control characters).
- 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:
  - On UNIX and Windows NT servers, specify FILEFORMAT=RECORD. The first two bytes of each record determine the record length.
  - In MVS, specify record format VBA (variable blocked with ANSI carriage control characters or VBM (variable blocked with machine carriage control characters).

Generic syntax

PAGEDEF=*pdefname*

Options and values

The *pdefname* can be one to eight alphanumeric or national characters, including the two-character prefix, if there is one. On UNIX servers, the *pdefname* is case-sensitive. Specify the file name, not the file extension. However, the file extension must be PDEF3820, PDEF38PP, or PDE (or no file extension).

Related parameters

PDEFLIB parameter on page 91.

PARMDD

Note: This parameter is ignored when you use the OnDemand data loading programs (arsload and arsadmin) to process files.

Required	Default Value	Data Type	
		AFP	Line
No		X	X

Identifies the name or the full path name of the file that contains the ACIF parameters, options, and data values. Specify the PARMDD parameter only when running ACIF from the command prompt. When you index files using the OnDemand data indexing and loading programs, OnDemand automatically retrieves the ACIF parameters from the database.

Generic syntax

PARMDD=*filename*

Options and values

The *filename* is the name or full path name of the file that contains the ACIF parameters. On UNIX servers, the file name is case sensitive. If you specify the file name without a path, ACIF searches for the file name in the current directory.

PDEFLIB

Required	Default Value	Data Type	
		AFP	Line
No		X	

Specifies the directories in which page definitions are stored. ACIF searches for a page definition in the following order:

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the PDEFLIB parameter, if any.
3. The paths you specified in the RESLIB parameter, if any.
4. On AIX servers, the paths specified by the PSFPATH environment variable, if it is set.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists.

Generic syntax

PDEFLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
PDEFLIB=/tmp:/usr/resources:/usr/lpp/ars/pdeflib
```

ACIF searches the paths in the order specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

Related parameters

PAGEDEF parameter on page 89.

RESLIB parameter on page 95.

USERLIB parameter on page 103.

PRMODE

Required	Default Value	Data Type	
		AFP	**Line**
No		X	

If the input data contains shift-in and shift-out codes, determines how ACIF processes them. Shift-in and shift-out codes (X'0E' and X'0F') indicate when the code points in a record change from single byte to double byte or double byte to single byte.

Generic syntax

PRMODE=*value*

Options and values

The *value* can be:

SOSI1

ACIF converts each shift-out and shift-in code to a blank character and a Set Coded Font Local text control. For the SOSI1 process to work correctly, the first font specified in the CHARS parameter (or in a font list in a page definition) must be a single byte font and the second font must be a double byte font.

SOSI2

ACIF converts each shift-out and shift-in code to a Set Coded Font Local text control.

SOSI3

ACIF converts each shift-out code to a Set Coded Font Local text control. ACIF converts each shift-in code to a Set Coded Font Local Text control and two blank characters. The SOSI3 data conversion is the same as the SOSI3 data conversion performed by PSF/MVS.

Related parameters

The CHARS parameter on page 63.

PSEGLIB

Required	Default Value	Data Type	
		AFP	**Line**
No		X	

Identifies the directories in which page segments and BCOCA, GOCA, and IOCA objects are stored. ACIF searches for page segments in the following order:

1. The paths you specified with the USERLIB parameter, if any.
2. The paths you specified with the PSEGLIB parameter, if any.
3. The paths you specified with the RESLIB parameter, if any.
4. On AIX servers, the paths specified by the PSFPATH environment variable, if it is set.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists.

Generic syntax

PSEGLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
PSEGLIB=/tmp:/usr/resources:/usr/lpp/ars/pseglib
```

ACIF searches the paths in the order specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

Related parameters

RESLIB parameter on page 95.

USERLIB parameter on page 103.

RESEXIT

Required	Default Value	Data Type	
		AFP	Line
No		X	

Identifies 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. For more information about optional program exits you can use to customize how ACIF handles input and output data, please refer to "Chapter 5. ACIF User Exits and Attributes of the Input Print File" on page 151.

Generic syntax

RESEXIT=*name*

Options and values

The *name* is the file name or full path name of the resource exit program. On UNIX servers, the file name is case sensitive. If you specify the file name without a path, ACIF searches for the file name in the paths specified by the PATH environment variable.

RESFILE

Required	Default Value	Data Type	
		AFP	Line
No	SEQ	X	

Determines the format of the resource file (MVS and OS/390 systems) ACIF creates. ACIF can create either a sequential data set (SEQ) or a partitioned data set (PDS) from resources it retrieves from the PSF/MVS or PSF for OS/390 resource libraries. If this parameter is not specified, ACIF writes a sequential data set to the DDname specified in the RESOBJDD parameter.

Specifying SEQ creates a resource group that can be concatenated to the document file as inline resources. You may need to concatenate the files created by ACIF before transmitting them to the workstation and processing the data with the OnDemand data loading program. A file created by selecting PDS cannot be concatenated to the document file.

Generic syntax

RESFILE=*type*

Options and values

The *type* can be:

SEQ

Creates a sequential data set that can be concatenated to the document file.

PDS

Creates a partitioned data set that cannot be concatenated to the document file.

RESLIB

Required	Default Value	Data Type	
		AFP	**Line**
No		X	

Determines the paths for the system resource directories. System resource directories typically contain resources that are shared by many users. The directories can contain any AFP resources (fonts, page segments, overlays, page definitions, form definitions, bar code objects, image objects, or graphics objects). ACIF searches for resources in the following order:

1. Paths you specified with the USERLIB parameter, if any.
2. Paths you specified with the FDEFLIB, FONTLIB, PDEFLIB, PSEGLIB, and OVLYLIB parameters, if any, for specific types of resources.
3. Paths you specified with the RESLIB parameter, if any.
4. On AIX servers, paths specified by the PSFPATH environment variable, if it is set.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists.
6. For fonts, on AIX servers, the directory /usr/lpp/afpfonts, if it exists.
7. For fonts, on AIX servers, the directory /usr/lpp/psf/fontlib, if it exists.

Generic syntax

RESLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
RESLIB=/tmp:/usr/resources:/usr/lpp/ars/reslib
```

ACIF searches the paths in the order specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

Related parameters

FONTLIB parameter on page 72.

FDEFLIB parameter on page 65.

OVLYLIB parameter on page 88.

PDEFLIB parameter on page 91.

PSEGLIB parameter on page 93.

USERLIB parameter on page 103.

RESOBJDD

Note: This parameter is ignored when you use the OnDemand data loading programs (arsload and arsadmin) to process files.

Required	Default Value	Data Type	
		AFP	**Line**
No	RESOBJ	X	

Identifies the name or the full path name of the resource file produced by ACIF. The resource file contains all of the resources required to view or reprint pages of the report.

Generic syntax

RESOBJDD=*filename*

Options and values

The *filename* is the file name or full path name of the resource group file. On UNIX servers, the file name is case sensitive. If you specify the file name without a path, ACIF puts the resource group file in the current directory.

RESTYPE

Required	Default Value	Data Type	
		AFP	**Line**
No	NONE	X	

Determines the types of AFP print resources ACIF should collect and include in the resource group file.

Generic syntax

RESTYPE=*value*

Options and values

The *value* can be:

NONE

No resource file is created.

FDEF

The form definition used in processing the file will be included in the resource file.

PSEG

Page segments required to print or view the output file will be included in the resource file.

OVLY

Overlays required to print or view the output file will be included in the resource file.

FONT

Font character sets and code pages required to print or view the output file will be included in the resource file. If you specify MCF2REF=CF, ACIF also includes coded fonts in the resource file.

BCOCA

BCOCA objects required to print or view the output file will be included in the resource file.

GOCA

GOCA objects required to print or view the output file will be included in the resource file.

IOCA

IOCA objects required to print or view the output file will be included in the resource file.

*type***,...***type*

A comma-separated list of two or more specific types of resources to collect. For example, to collect form definitions, page segments, and overlays, use the following format:

```
RESTYPE=fdef,pseg,ovly
```

ALL

All resources required to print or view the output file will be included in the resource file.

Because OnDemand does not use AFP raster fonts when presenting the data on the screen, you may want to specify RESTYPE=FDEF,OVLY,PSEG to prevent fonts from being included in the resource file. This reduces the number of bytes transmitted over the network when the file is transferred to the workstation.

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

Because multiple resource types are now contained in the page segment library and ACIF does not enforce a prefix for the 8-character resource name, you should define a naming convention that identifies each type of resource in the page segment library. We recommend a two-character prefix, for example:

- B1 for BCOCA objects
- G1 for GOCA objects
- I1 for IOCA objects
- S1 for page segments

Related parameters

The MCF2REF parameter on page 85.

The RESLIB parameter on page 95.

The RESOBJDD parameter on page 96.

TRC

Required	Default Value	Data Type	
		AFP	Line
No	NO	X	X

Identifies whether the input file contains table reference characters (TRCs). Some applications may produce output that uses different fonts on different lines of a file by specifying TRCs at the beginning of each line after the carriage-control character if one is present.

Consider the following when you use TRCs:

- The order in which the fonts are specified in the CHARS parameter establishes which number is assigned to each associated TRC. For example, the first font specified is assigned 0, the second font 1, and so on.
- If you specify TRC=YES and the input data does not contain TRCs, ACIF interprets the first character (or second, if carriage-control characters are used) of each line as the font identifier. Consequently, the font used to process each line of the file may not be the one you expect, and one byte of data will be lost from each line.
- If you specify TRC=NO or you do not specify the TRC parameter and the input contains a TRC as the first character (or second if carriage-control characters are used) of each line, ACIF interprets the TRC as a text character in the processed output, rather than using it as a font identifier.

Generic syntax

TRC=*value*

Options and values

The *value* can be:

NO

The input does not contain TRCs.

YES

The input does contain TRCs.

Related parameters

The CHARS parameter on page 63.

TRIGGER

Required	Default Value	Output Data Type	
		AFP	Line
Yes		X	X

Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to eight triggers.

When running ACIF under Windows NT and you define a TRIGGER parameter using structured field data, you must change the order of the length bytes in the trigger string value. The length bytes are bytes two and three of a structured field. Some instructions in the x86 architecture use the

length bytes in the reverse of the order that they appear in the input data. ACIF automatically changes the order of the length bytes in all input structured fields before indexing the data. For example, in the input data, a structured field may appear as follows:

```
5A0010D3EEEE00. . .
```

In the example, the hexadecimal value 0010 represents the length of the structured field (16 bytes following the 5A). To support the x86 architecture, after reading the input data into its storage, ACIF changes the order of the length bytes before indexing the data. The example data would appear in ACIF storage as follows:

```
5A1000D3EEEE00. . .
```

When defining a TRIGGER parameter using structured field data, the order of the length bytes in the trigger string value must be the same as the data in ACIF storage, not the original input data. For example:

```
TRIGGER1=*,1,X'5A1000D3EEEE00. . .'
```

Before writing the output data, ACIF restores the length bytes to their original locations.

Generic syntax

TRIGGERn=*record*,*column*,*value*[**,(TYPE=***type***)**]

Options and values

n

The trigger parameter identifier. When adding a trigger parameter, use the next available number, beginning with 1 (one).

record

The input record where ACIF locates the trigger string value. For TRIGGER1 and float triggers, the input record must be * (asterisk), so that ACIF searches every input record for the trigger string value. For other group triggers, the input record is relative to the record that contains the TRIGGER1 string value. The supported range of record numbers is 0 (the same record that contains the TRIGGER1 string value) to 255.

column

The beginning column where ACIF locates the trigger string value. The supported range of column numbers is 0 to 32756. Specify an * (asterisk) or 0 (zero) for the column, to cause ACIF to scan the record from left to right looking for the trigger string value.

value

The actual string value ACIF uses to match the input data. The string value is case sensitive. If the input data is encoded in EBCDIC, enter the value in hexadecimal. The value can be from 1 to 250 bytes in length.

TYPE=*type*

The trigger type. The default trigger type is group. TRIGGER1 must be a group trigger. Valid trigger types are:

GROUP

Triggers that identify the beginning of a group. Define only as many group triggers as needed to identify the beginning of a group. In many cases, you may need only one group trigger.

GROUP,RECORDRANGE=(*start***,***end***)**

Triggers that identify 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 string value, ACIF stops after matching the value in record six.

FLOAT

Triggers that identify 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 string 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.

Examples

TRIGGER1

The following TRIGGER1 parameter causes ACIF to search column one of every input record for the occurrence of a skip-to-channel one carriage control character. The record value for TRIGGER1 must be an asterisk. The input data is encoded in EBCDIC. The trigger type is optional, but defaults to group. TRIGGER1 must be a group trigger.

```
TRIGGER1=*,1,X'F1',(TYPE=GROUP)
```

The following TRIGGER1 parameter causes ACIF to attempt to match the string value PAGE 1 beginning in column two of every input record. The record value for TRIGGER1 must be an asterisk. The input data is encoded in EBCDIC. The trigger type is optional, but defaults to group. TRIGGER1 must be a group trigger.

```
TRIGGER1=*,2,X'D7C1C7C5404040F1',(TYPE=GROUP)
```

Group trigger

The following trigger parameter causes ACIF to attempt to match the string value Account Number beginning in column fifty of the sixth input record following the TRIGGER1 record. A record number must be specified for a group trigger (other than TRIGGER1 or a recordrange trigger). The input data is encoded in EBCDIC. The trigger type is optional, but defaults to group.

```
TRIGGER2=6,50,X'C1838396A495A340D5A494828599',(TYPE=GROUP)
```

Recordrange trigger

The following trigger parameter causes ACIF to attempt to locate the string value Account Number beginning in column fifty within a range of records (the trigger string value can occur in records six, seven, or eight following TRIGGER1) in each group. An asterisk must be used for record number (ACIF uses the recordrange to determine which records to search for the trigger string value). The input data is encoded in EBCDIC. The trigger type is optional, but must be group for a recordrange trigger.

```
TRIGGER2=*,50,X'C1838396A495A340D5A494828599',(TYPE=GROUP,RECORDRANGE=(6,8))
```

Float trigger

The following trigger parameter causes ACIF to attempt to match the string value Type of Income, beginning in column five of every record in the group. An asterisk must be specified for the record number. The input data is encoded in EBCDIC. The trigger type is float and must be specified.

```
TRIGGER3=*,5,X'E3A89785409686 40C99583969485',(TYPE=FLOAT)
```

Trigger using structured field data

Note: This example is for ACIF running under Windows NT

The following trigger parameter shows how to specify structured field data. Because ACIF changes the order of the length bytes in all structured fields before indexing the data, you must make sure that the order of the length bytes in the trigger string value is the same as the data in ACIF storage. This example could be used to index mixed mode data, such as a line data report that contains NOP structured fields, allowing ACIF to logically segment the report into documents.

```
TRIGGER1=*,1,X'5A1000D3EEEE000000000000000000000000',(TYPE=GROUP)
```

Related parameters

The FIELD parameter on page 66.

UNIQUEBNGS

Required	Default Value	Output Data Type	
		AFP	Line
No	YES	X	X

Determines 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 contains an index value and a sequence number.

Generic syntax

UNIQUEBNGS=*value*

Options and values

The *value* can be:

YES

ACIF generates an eight-character numeric string and appends the string to the group name. The default, if you specify DCFPAGENAMES=NO.

NO

ACIF does not generate the string. The default, if you specify DCFPAGENAMES=YES. Specify no if you use the AFP API to generate group names. Specify no if you use DCF to generate the input data.

Related parameters

The DCFPAGENAMES parameter on page 65.

USERLIB

Required	Default Value	Output Data Type	
		AFP	Line
No		X	

Identifies 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, bar code objects, image objects, or graphics objects). 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, OVLYLIB, PDEFLIB, or PSEGLIB parameters. ACIF searches for resources in the following order:

1. Paths you specify with the USERLIB parameter, if any.
2. Paths you specify with the FDEFLIB, FONTLIB, OVLYLIB, PDEFLIB, or PSEGLIB parameters, for specific types of resources, if any.
3. Paths you specify with the RESLIB parameter, if any.
4. On AIX servers, paths specified by the PSFPATH environment variable, if it is set.
5. On AIX servers, the directory /usr/lpp/psf/reslib, if it exists.
6. For fonts, on AIX servers, the directory /usr/lpp/afpfonts, if it exists.
7. For fonts, on AIX servers, the directory /usr/lpp/psf/fontlib, if it exists.

Generic syntax

USERLIB=*pathlist*

Options and values

The *pathlist* is a string of one or more valid path names. For example:

```
USERLIB=/tmp:/usr/resources:/usr/lpp/ars/userlib
```

ACIF searches the paths in the order specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

USERMASK

Required	Default Value	Output Data Type	
		AFP	**Line**
No		X	X

Identifies a symbol and string used to match a field. The symbol can represent one or more characters, including the characters reserved for the field mask. The string contains the character or characters you want to match to the field data.

Generic syntax

USERMASK=*number***,***symbol***,'***string***'**

Options and values

number

The number of the user mask. You can define up to four user masks, using the numbers 1 (one) through 4 (four).

symbol

The *symbol* that represents the characters in the *string*. You can use any printable character, except those reserved for the field mask (#@=¬^%). The character that you specify does not match its literal value in the field. That is, if you specify an * (asterisk) as the symbol, ACIF will not match an asterisk character in the field.

string

One or more characters that you want to match in the field data.

Examples

A typical use of a USERMASK is to match specific characters that may appear in a field column. For example, the following definitions:

```
USERMASK=1,'*','AaBbCc'
FIELD3=*,*,15,(OFFSET=(10:24),MASK='*@@@@@@@@@@@@@@',ORDER=BYROW)
```

Cause ACIF to match an upper or lower case A, B, or C in the first position of a 15 character string, such as a name.

A user mask can also be used to 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, the following definitions:

```
USERMASK=2,'*','%'
FIELD4=*,*,3,(OFFSET=(10:12),MASK='##*',ORDER=BYROW)
```

Cause ACIF to match a three-character string that contains two numerics and the percent sign, for example 85%.

Related parameters

The FIELD parameter on page 66.

Chapter 4. Message Reference

Introduction

ACIF creates a message list at the end of each compilation. A return code of 0 (zero) means that ACIF completed processing without any errors.

ACIF detects a number of error conditions that can be logically grouped into three categories:

- **Severe**

  ACIF issues an error message and return code of 8 (eight) or 16 (sixteen) and terminates processing the current input file. Most error conditions detected by ACIF fall into this category. The exact method of termination may vary. For certain severe errors, ACIF may fail with a segment fault. This is generally the case when some system service fails. In other cases, ACIF terminates with the appropriate error messages written either to standard error or to a file. When ACIF is invoked by the OnDemand data indexing and loading programs, error messages are written to the system log. If you invoke ACIF from the command prompt, you can specify the name or the full path name of the file to contain processing messages with the MSGDD parameter.

- **Warning**

  ACIF issues a warning message and a return code of 4 (four) when the fidelity of the document (assuming it is reprinted) may be in question.

- **Informational**

  When ACIF processes a file, it issues informational messages that allow the user to determine if the correct processing parameters have been specified. These messages can assist in providing an audit trail.

Multiple Message Scenarios

ACIF may issue more than one error message as a result of a single error condition. These situations are limited to the parsing of the Advanced Function Presentation (AFP) structured fields (for example, determining the length and type of the structured field). Some possible scenarios include:

0425-105, 0425-108, 0425-109, 0425-103

0425-105, 0425-108, 0425-110, 0425-103

0425-106, 0425-108, 0425-109, 0425-103

0425-106, 0425-108, 0425-110, 0425-103

Any subset of the listed messages is also possible, provided you start with the first one (for example, 0425-105, 0425-108, 0425-109; or 0425-105, 0425-108; or 0425-105, 0425-110). The first message accurately describes the error condition; any subsequent messages provide additional information. Additional error messages may not always be accurate.

Message 0425-101 may occur after many error conditions, because ACIF attempts to locate the end of the resource containing the error as part of its recovery procedure.

Messages

0425-031 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 dataset may have been created by a program that creates inline medium maps, but a data set that contains inline medium maps cannot be printed.

System Action: ACIF stops processing the input file.

User Response: Correct the error and resubmit ACIF.

0425-102 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 input file.

User Response: Correct the error and resubmit ACIF.

0425-103 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field name1* **STRUCTURED FIELD WAS FOUND WHERE** *structured field name2* **STRUCTURED FIELD WAS EXPECTED.**

Explanation: *Structured field name1* is incorrect at the present point in the input file or resource. The structured-field type expected at this point is *structured field name2*. The required structured field is missing or out of sequence.

Subsequent error messages give additional information about the processing environment when the error occurred.

System Action: ACIF stops processing the input file.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-104 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID:** *structured field* **STRUCTURED FIELD IS NOT ALLOWED OR FORMS AN INVALID SEQUENCE.**

Explanation: The structured field identified in this message is either out of sequence or is invalid in an object. If inline resources are used with header pages, multiple resource groups might be present.

System Action: ACIF stops processing the input file.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-105 **THE ERROR REPORTED ABOVE OCCURRED IN LOGICAL RECORD NUMBER** *record number***, WHOSE SEQUENCE NUMBER IS** *sequence number***.**

Explanation: This message is given in addition to the message that describes the error. It identifies the specific invalid input record.

The record number specified is relative to the user file and is different for multiple transmissions of the file. However, the record number may be inaccurate if the file is using a page definition that performs conditional processing.

The sequence number may print as NOT AVAILABLE in the message. For example, an AFP 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.

0425-106 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: NAME** *token name* **IN** *Begin-type structured field* **DOES NOT MATCH NAME** *token name* **IN** *End-type structured field***.**

Explanation: The Token Name parameters in the Begin-type and End-type structured fields identified in this message do not match. Structured fields may be out of sequence in the input file.

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 file or resource. ACIF issues additional messages identifying the processing environment when the error was found.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-108 **THE ERROR REPORTED ABOVE WAS DETECTED WITHIN OBJECT TYPE** *object type* **WITH TOKEN NAME** *token name***.**

Explanation: This message is issued in addition to the message that describes the error. The objects that were being processed are listed to identify the location of the error in the input file 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.

0425-109 THE ERROR REPORTED ABOVE WAS CAUSED BY THE RESOURCE *resource name* IN AN EXTERNAL LIBRARY OR AN INLINE RESOURCE.

Explanation: This message is issued in addition to the message that describes the error. The object identified in the accompanying message was either a resource being processed from an external directory (or library in an MVS environment) or an inline resource. The previous error message, 0425-108, identified the object as a page definition, form definition, font, code page, font character set, page segment, barcode object, graphics object, image object, or an overlay. The combined information from these two messages can be used to identify the directory defined to ACIF in the appropriate *LIB* parameter (for example, **ovlylib** for overlays) if appropriate. In the case of an inline form definition or page definition, the resource does not reside in a directory but is included at the beginning of the file.

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.

0425-110 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE LENGTH SPECIFIED IN THE SELF-DEFINING PARAMETER *identifier* OF THE STRUCTURED FIELD *structured field* IS INCORRECT.

Explanation: Insufficient data was present in the structured field for the length given in the self-defining parameter. The structured field can be contained in a bar code object, a graphics object, or an image object. The bar code object may be contained in an overlay or an AFP print file, or it may be imbedded in a file containing line data. The graphics object may be contained in an AFP print file or an overlay, or it may be imbedded in a file containing line data. The image object may be contained in an AFP print file, an overlay, or a page segment, or it may be imbedded in a file containing line data.

System Action: ACIF stops processing the input and issues a message identifying the position of the structured field in the file 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 local problem-reporting procedures to report this message.

0425-112 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 input.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-113 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-114 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-116 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-117 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: LENGTH INDICATED IN THE STRUCTURED FIELD INTRODUCER IS INCORRECT FOR *structured field name* **STRUCTURED FIELD.**

Explanation: The length indicated by the structured-field introducer specifies an invalid number of bytes for the structured field identified in this message. This error is caused by one of the following:

- The Extension or Padding Indicator flags in the structured-field introducer are set incorrectly.
- One or more of the parameters in the invalid structured field contain too many bytes of data.

In some cases, the length of a structured field is specified in a parameter located in another structured field. For example, the length of the 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a

resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-118 UNSUPPORTED STRUCTURED FIELD *structured field code* **WAS IGNORED, AND, IF IT BEGAN AN OBJECT, THE OBJECT WAS IGNORED.**

Explanation: The Identifier parameter in the structured-field introducer for the invalid structured field specified a structured-field code that was not recognized as a valid structured-field code.

System Action: If the structured field began an object, the object was ignored. Otherwise, only the structured field was ignored, and processing of the rest of the file continues as usual.

ACIF issues a message identifying the position of the structured field in the input file or containing resource. ACIF issues additional messages identifying the processing environment when the error was found.

User Response: If the printed output was unacceptable, use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-120 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field name1* **STRUCTURED FIELD CONTAINS AN INCORRECT VALUE FOR THE SIZE OF THE** *structured field name2* **REPEATING GROUP.**

Explanation: *Structured field name1* specifies the length of each repeating group found in *structured field name2*. Either the value specified in *structured field name1* for the size of the repeating group is too small, or the actual length of the repeating-group data is not a multiple of the size specified.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-130 DATA IN AN INPUT RECORD IS INVALID: *structured field name* **STRUCTURED FIELD IS NOT ACCEPTABLE AT THE START OF A DATA STREAM.**

Explanation: The structured-field type identified in this message is not valid at the start of the file. Subsequent error messages give additional information about the processing environment when the error occurred.

System Action: ACIF stops processing the input file.

User Response: Correct the process used to create the input file. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-135 DATA IN A FORMDEF RESOURCE IS INVALID: DUPLICATE OVERLAY LOCAL IDENTIFIER WAS FOUND IN THE *structured field* **STRUCTURED FIELD.**

Explanation: The same local identifier was found assigned to more than one Overlay Local Identifier parameter in the Map Medium Overlay (MMO) or Map Page Overlay (MPO) structured field repeating groups. The MMO structured field is contained in the form definition. The MPO is

contained in the page definition or the input file.

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 process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-138 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: OVERLAY LOCAL IDENTIFIER VALUE IS NOT ACCEPTABLE IN THE *structured field* **STRUCTURED FIELD.**

Explanation: An invalid Overlay Local Identifier was encountered in the Map Medium Overlay (MMO), Map Page Overlay (MPO), or Medium Modification Control (MMC) structured field repeating groups. The MMO and MMC structured fields are contained in the form definition. The MPO is contained in the page definition or the input file.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-139 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-140 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-141 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition

with the error, use local problem-reporting procedures to report this message.

0425-143 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-145 DATA IN A FORMDEF RESOURCE IS INVALID: THE FORMS-FLASH VALUE IN MMC STRUCTURED FIELD, ID *identifier***, IS NOT ACCEPTABLE.**

Explanation: The Medium Modification Control (MMC) structured field contains an invalid value for the repeating group that contains forms-flash modification. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-146 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-147 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-152 **DATA IN A FORMDEF RESOURCE IS INVALID: MMC STRUCTURED FIELD WAS NOT FOUND TO COMPARE WITH IDENTIFIER** *identifier value* **IN MCC STRUCTURED FIELD.**

Explanation: The Medium Modification Control Identifier parameter in the Medium Copy Count (MCC) structured field contains a value that did not match the Medium Modification Control Identifier parameter in any Medium Modification Control (MMC) structured field in the form environment group. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-154 **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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-155 **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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition to either reduce the number of copy groups in the Medium Copy Count (MCC) structured field or to 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.

0425-156 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: NULL NAME IS NOT ACCEPTABLE IN** *structured field name* **STRUCTURED FIELD.**

Explanation: All Begin-type and End-type structured fields can include an 8-byte token name. A null token name is not allowed for the listed structured field.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field

causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-157 **MISMATCH BETWEEN PRINT DATA SET AND FORMDEF RESOURCE: MEDIUM MAP** *medium map* **SPECIFIED IN IMM STRUCTURED FIELD WAS NOT FOUND IN FORMDEF** *form definition name***.**

Explanation: The Token Name parameter in the Invoke Medium Map (IMM) structured field specifies the token name used to locate a medium map in the form definition. This parameter must match the Token Name parameter specified in bytes 0–7 in one of the Begin Medium Map (BMM) structured fields in the current form definition. The IMM structured field is contained in the input file (referred to as a data set in an MVS environment).

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: Ensure that the correct form definition was specified. If it was, and if you added the Invoke Medium Map structured field to the input file, change the Token Name in the IMM structured field and rerun ACIF. Refer to *Mixed Object Document Content Architecture Reference* for more information about the BMM and IMM structured fields. If the correct form definition was specified and if you used a program to imbed the IMM structured field in the input file, verify that the copy group name that you gave the program is valid for the form definition you have specified.

0425-158 **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 input file (referred to as a data set in an MVS environment) contains at least one record of ASCII or line data. A file containing ASCII or line data cannot be printed without an active page definition. No **pagedef** parameter was provided for this job.

This error can also occur if an AFP page in the print file contains a structured field without the required X'5A' control character preceding the structured-field introducer. The missing control character makes the record appear to be line data. A page definition is necessary to process line data.

This error can also occur if a Presentation-Text Data (PTX) structured field or a Begin Image (BIM) structured field is encountered outside of a page.

System Action: ACIF stops processing the input file.

User Response: If you intended to process ASCII or line data, define the PAGEDEF parameter in the application (or, if you are running ACIF from the command prompt, include the parameter in the command syntax).

If you did not intend to print ASCII or line data, correct the process used to create the input file to ensure that all AFP print data records begin with the X'5A' control character.

If you used an IBM licensed program to create the AFP print file, use local problem-reporting procedures to report this message.

0425-159 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 file ended before a corresponding End-type structured field was found.

System Action: ACIF stops processing the input file.

User Response: Correct the process used to create the input file. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-162 MISMATCH BETWEEN PRINT DATA SET AND PAGEDEF RESOURCE: DATA MAP *data map* **SPECIFIED IN IDM STRUCTURED FIELD WAS NOT FOUND IN PAGEDEF** *page definition***.**

Explanation: The Token Name parameter in the Invoke Data Map (IDM) structured field specifies the token name used to locate a data map in the page definition. The name must match the value specified in the Token Name parameter in the Begin Data Map (BDM) structured field in the current page definition. The IDM structured field is contained in the input file (referred to as a data set in an MVS environment).

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: Ensure that the correct page definition was specified. If it was, and if you added the Invoke Data Map structured field to the input file, change the Token Name in the IDM structured field and rerun ACIF. Refer to *Advanced Function Printing: Programming Guide and Line Data Reference* for more information about the BDM and IDM structured fields. If the correct page definition was specified, and if you used a program to imbed the IDM structured field in the input file, verify that the data map name that you supplied the program is one that is valid for the page definition you have specified.

0425-163 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE SCALE FACTOR VALUE IN THE IOC STRUCTURED FIELD IS NOT ACCEPTABLE.

Explanation: The Image Block Scale Factor parameter in the Image Output Control (IOC) structured field is invalid. The image block or image cell may be contained in an overlay, a page segment, or an AFP print file. It may also be imbedded in an input file containing line data using a Begin Image (BIM) structured field.

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 process used to create the image. If you used an IBM licensed program to create the image with the error, use local problem-reporting procedures to report this message.

0425-166 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: AN ENTRY IN A MCF STRUCTURED FIELD CONTAINS AMBIGUOUS IDENTIFICATION.

Explanation: Two ways to identify a font in the Map Coded Font (MCF) structured field are either with a Coded Font Name parameter or with a combination of the Font Character Set Name parameter and the Code Page Name parameter. One of the repeating groups in an MCF structured field specified both a Coded Font Name parameter and at least a Font Character Set Name or a Code Page Name parameter. The MCF structured field is contained in an AFP print file, an overlay, or a page definition.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-167 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 an AFP print file, an overlay, or a page definition.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-170 DATA IN A FORMDEF RESOURCE IS INVALID: THE SIMPLEX/DUPLEX VALUE IN MMC STRUCTURED FIELD, ID *identifier*, IS NOT ACCEPTABLE.

Explanation: In the Medium Modification Control (MMC) structured field with the specified identifier, either the simplex or the duplex keyword-parameter value is invalid. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-171 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: FONT LOCAL IDENTIFIER VALUE IS NOT ACCEPTABLE IN THE *structured field* STRUCTURED FIELD.

Explanation: The Map Coded Font (MCF) structured field consists of repeating groups. In one of the groups, the value of the Coded Font Local Identifier parameter for the font (section) being mapped is not valid. The MCF structured field is contained in an AFP print file, an overlay, or a page definition.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-172 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-178 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-179 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-181 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition

with the error, use local problem-reporting procedures to report this message.

0425-188 **DATA IN A FORMDEF RESOURCE IS INVALID: THE SET OF MODIFICATIONS SPECIFIED IN THE MCC STRUCTURED FIELD SELECTS MORE THAN ONE INPUT SOURCE.**

Explanation: The Medium Copy Count (MCC) structured field refers to one or more Medium Modification Control (MMC) structured fields, which include requests for the primary input source and the alternate source. You cannot specify both the primary input source and the alternate source for one copy group. The MCC and MMC structured fields are contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-190 **DATA IN A FORMDEF RESOURCE IS INVALID: THE BIN-SELECTION VALUE IN MMC STRUCTURED FIELD, ID** *identifier***, IS NOT ACCEPTABLE.**

Explanation: In the Medium Modification Control (MMC) structured field with the identifier specified in the message text, the bin-selection parameter value was invalid. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-191 **DATA IN A FORMDEF RESOURCE IS INVALID: THE SUPPRESSION LOCAL IDENTIFIER VALUE IN MMC STRUCTURED FIELD, ID** *identifier***, IS NOT ACCEPTABLE.**

Explanation: The Medium Modification Control Identifier parameter in a Medium Modification Control (MMC) structured field is invalid. The MMC structured field is contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-210 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: A REQUIRED SELF-DEFINING PARAMETER WITH ID** *id* **WAS MISSING FROM A** *structured field name* **STRUCTURED FIELD.**

Explanation: The self-defining parameter specified in the message was not found in the indicated structured field. This is a required self-defining parameter. The structured field is contained in an image object. The image object may be contained in an AFP print file or an overlay, or it may be imbedded in an input file containing line data.

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 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.

0425-212 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE UNIT BASE PARAMETER IN THE** *structured field name* **STRUCTURED FIELD IS INVALID.**

Explanation: An invalid Unit Base value was encountered in the structured field identified in this message. The structured field is contained in the Object Environment Group of an image object. The image object may be contained in an AFP print file or an overlay, or it may be imbedded in an input file containing line data.

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 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.

0425-213 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE L-UNITS PER UNIT BASE PARAMETER IN THE** *structured field name* **STRUCTURED FIELD IS INVALID.**

Explanation: An invalid L-Units value was encountered in the structured field identified in this message. The structured field is contained in the Object Environment Group of an image object. The image object may be contained in an AFP print file or an overlay, or it may be imbedded in an input file containing line data.

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 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.

0425-217 **DATA IN AN INPUT RECORD IS INVALID: PARAMETER IN A BR STRUCTURED FIELD CONTAINS UNACCEPTABLE DATA.**

Explanation: One of the parameters in the Begin Resource (BR) structured field is invalid. The BR structured field is contained in the input file.

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 process used to place the resource in the input file. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-221 **DATA IN A FORMDEF RESOURCE IS INVALID: THE ORIENTATION VALUE** *value* **IN THE MDD STRUCTURED FIELD IS UNACCEPTABLE.**

Explanation: The Medium Descriptor (MDD) structured field has an invalid orientation value. The MDD structured field is contained in the form definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-246 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: A REQUIRED TRIPLET WITH ID X'4B' WAS MISSING FROM AN IOB STRUCTURED FIELD.**

Explanation: The x or y axis origin for object content or an object area size (X'4C') triplet was specified on an IOB but no measurement units (X'4B') triplet was specified. The structured field

is contained in a print file or overlay.

System Action: ACIF stops processing the input file.

User Response: If you placed the IOB structured field in the input file or overlay, correct the error and resubmit the ACIF job. Refer to the *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 file or overlay, contact your system programmer.

0425-247 DATA IN AN INPUT RECORD IS INVALID: A PARAMETER IN AN IOB STRUCTURED FIELD CONTAINS UNACCEPTABLE DATA.

Explanation: One of the parameters in the Include Object (IOB) structured field is invalid. The object type specified is not supported or is invalid or 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'. The IOB structured field is contained in the print file or an overlay.

System Action: ACIF stops processing the input file.

User Response: If you placed the IOB structured field in the input file or overlay, correct the error and resubmit the ACIF job. Refer to the *Mixed Object Document Content Architecture Reference* for more information about the structured field. If you used an IBM licensed program to place the IOB structured field in the print file or overlay, contact your system programmer.

0425-248 DATA IN A PAGE SEGMENT IS INVALID: *structured field name* **STRUCTURED FIELD IS NOT ALLOWED IN A PAGE SEGMENT INCLUDED WITH AN IOB.**

Explanation: Only MODCA page segments are allowed to be included with an IOB structured field. MODCA page segments cannot contain IM1 image or PTOCA data.

System Action: ACIF stops processing the input file.

User Response: If you placed the IOB structured field in the input file or overlay, correct the error and resubmit the ACIF job. Refer to the *Mixed Object Document Content Architecture Reference* for more information about the structured field. If you used an IBM licensed program to place the IOB structured field in the print file or overlay, contact your system programmer.

0425-250 DATA IN A PAGE OR RESOURCE IS MISSING: THE REQUIRED STRUCTURED FIELD *structured field name* **COULD NOT BE FOUND TO COMPLETE THE PROCESSING OF A PAGE OR RESOURCE.**

Explanation: The structured field identified in this message is required to complete the processing of a page or resource. This structured field was not found before the end of the page or resource was encountered.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-251 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 input

file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-253 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-254 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-255 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE FONT RESOLUTION AND METRIC TECHNOLOGY TRIPLET SPECIFIES AN INCORRECT VALUE.

Explanation: There is an invalid 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 an input data set or overlay.

System Action: ACIF stops processing the input.

User Response: Correct the error in the input data set or overlay and rerun ACIF. Refer to the *Mixed Object Document Content Architecture Reference* for more information about the X'84' triplet.

0425-256 DATA IN A FORMDEF RESOURCE IS INVALID: THE FONT FIDELITY TRIPLET SPECIFIES AN INCORRECT VALUE.

Explanation: There is an invalid value specified for the font unavailable exception continuation action in the Font Fidelity Triplet (X'78') on the Presentation Fidelity Control (PFC) structured field. The PFC is contained in the form definition resource.

System Action: ACIF stops processing the input.

User Response: Correct the error in the formdef and resubmit the job. Refer to the *Mixed Object Document Content Architecture Reference* for more information about the X'78' triplet.

0425-258 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: *structured field* STRUCTURED FIELD IS NOT ALLOWED BETWEEN OBJECTS.**

Explanation: The structured field identified in this message is not allowed at the point in the input file or resource at which it was found.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-259 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE X-DIRECTION AND Y-DIRECTION L-UNITS PER UNIT BASE VALUES SPECIFIED IN STRUCTURED FIELD *structured field* DO NOT MATCH.**

Explanation: The X-direction and Y-direction L-Units per Unit Base values in the structured field identified in the message are not identical.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-261 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD *structured field* CONTAINED A CODED-FONT-LOCAL-IDENTIFIER VALUE THAT WAS USED IN A PREVIOUS FONT MAPPING STRUCTURED FIELD.**

Explanation: One or more font mapping structured fields in the same active environment group or object environment group used the same coded font local identifier for different coded fonts. The Map Coded Font (MCF) structured field that attempted to use the already-mapped coded font local identifier is identified in the message. The MCF structured field is contained in an AFP print file, an overlay, or a page definition.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-262 **DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: STRUCTURED FIELD *structured field* CONTAINS AN INVALID ROTATION VALUE.**

Explanation: The rotation value specified in the named structured field is invalid.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error,

use local problem-reporting procedures to report this message.

0425-263 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: OVERLAY *overlay name* **NAMED IN AN IPO STRUCTURED FIELD IS NOT NAMED IN AN MPO STRUCTURED FIELD.**

Explanation: An Include Page Overlay (IPO) structured field names a page overlay, but the overlay was not previously defined in the Map Page Overlay (MPO) structured field in the Active Environment Group (AEG) of the page, which contains the IPO. The MPO may be contained in the AEG of a composed-text page or a page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-264 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: A CODED FONT NAMED IN AN OBJECT ENVIRONMENT GROUP IS NOT NAMED IN THE ACTIVE ENVIRONMENT GROUP OF THE PAGE OR RESOURCE.

Explanation: A Map Coded Font (MCF) structured field in an object environment group names a coded font, but that coded font is not defined in the MCF structured field in the active environment group of the page or resource containing the object environment group.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource. ACIF stops processing.

User Response: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-267 EITHER NO ENVIRONMENT GROUP WAS SPECIFIED FOR THE PAGE OR AN ERROR OCCURRED IN THE ENVIRONMENT GROUP.

Explanation: Either no environment group was specified, or an error occurred in one of the structured fields in the environment group. If an environment group was present but contained an error, a previous ACIF message identifies the error. The environment group causing this error may be contained in an AFP print file, an overlay, or a page definition.

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: Use the accompanying messages to determine if the structured field causing the error is in the input file or in a resource. Correct the process used to create the input file or resource. If you used an IBM licensed program to create the file with the error, use local problem-reporting procedures to report this message.

0425-268 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 input file and issues a message identifying the position of the structured field in the file 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 local problem-reporting procedures to report this message.

0425-270 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 file containing ASCII or line data.

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 process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-271 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-272 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-273 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-274 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-275 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 simplexed page. Another medium map must be invoked to allow processing of the remaining print 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the form definition. If you used an IBM licensed program to create the form definition with the error, use local problem-reporting procedures to report this message.

0425-278 DATA IN AN INPUT RECORD OR RESOURCE IS INVALID: THE OUTPUT OPTION SPECIFIED IN THE *structured field name* **IS INVALID OR UNSUPPORTED.**

Explanation: The structured field in error contained an invalid Output Option value or the printer does not support the Output Option value. The structured field can be contained in a bar code object, graphics object, image object, or it was specified with an IOB structured field. The bar code object, graphics object, image object, or OIB can be contained in an overlay, MODCA page, or it can be imbedded in line data.

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 process used to create the object. If you used an IBM licensed program to create the object with the error, use local problem-reporting procedures to report this message.

0425-300 DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SKIPPING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* **IS 0.**

Explanation: The current record contains a control character that indicates a skip to a Line Descriptor (LND) structured field with a specific channel control. However, the LND structured field identified in this message had a value of 0 in its Next Line Descriptor If Skipping parameter. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting

procedures to report this message.

0425-301 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SKIPPING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **IS** *parameter value***. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF** *parameter value***.**

Explanation: In the Line Descriptor (LND) structured field identified in this message, the value of the next LND IF SKIPPING parameter is greater than the total number of LND structured fields in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-307 **DATA IN A PAGEDEF RESOURCE IS INVALID: IN LND STRUCTURED FIELD NUMBER** *structured field number***, THE REUSE RECORD FLAG WAS SET BUT THE NEXT LINE DESCRIPTOR IF REUSING DATA PARAMETER WAS 0.**

Explanation: In the Line Descriptor (LND) structured field identified in this message, the Reuse Record flag had a value of B'1', indicating that the 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-309 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE REPEATING GROUP LENGTH PARAMETER VALUE IN CCP STRUCTURED FIELD** *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. Either the Length of Repeating Groups parameter is zero, or the length of the repeating group data is not a multiple of the size specified in that parameter. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-310 **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 in the data map of the page definition had a value of zero. The LNC structured field is contained in the page definition.

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 process used to

create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-312 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-314 DATA IN A PAGEDEF RESOURCE IS INVALID: THE NUMBER OF REPEATING GROUPS PARAMETER VALUE IN CCP STRUCTURED FIELD *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. Either the Number of Repeating Groups parameter contained in the CCP structured field is zero, or the number of repeating groups does not match the number specified in the parameter. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-315 DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SPACING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER *structured field number* **IS 0.**

Explanation: The logical-record control character indicates that the Next Line Descriptor If Spacing parameter should be followed. However, in the Line Descriptor (LND) structured field identified in this message, the Next Line Descriptor If Spacing parameter value was zero. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-316 DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SPACING PARAMETER IN LND STRUCTURED FIELD NUMBER *structured field number* **IS** *parameter value***. THIS VALUE IS TOO LARGE.**

Explanation: The logical record control character indicates that the Next Line Descriptor If Spacing parameter in the Line Descriptor (LND) structured field should be followed. However, in the Line Descriptor (LND) structured field identified in this message, the Next Line Descriptor If Spacing parameter value was greater than the total number of line descriptors in the data map. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to

create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-317 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE LENGTH OF COMPARISON STRING PARAMETER VALUE IN CCP STRUCTURED FIELD** *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. Either the Length of Comparison String parameter is zero, or the length of the comparison string data does not match the length of a repeating group minus the fixed lengths of the remaining fields of the repeating group. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-319 **DATA IN A PAGEDEF RESOURCE IS INVALID: SUPPRESSION TOKEN NAME =** *token name* **IS INVALID IN LND STRUCTURED FIELD NUMBER =** *structured field number***.**

Explanation: The Suppression Token Name parameter in the Line Descriptor (LND) structured field in the page definition has a null value. A null value is any value that contains X'FFFF' in the first two bytes. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-320 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE IDENTIFIER** *identifier1* **SPECIFIED IN THE NEXT CCP IDENTIFIER PARAMETER IN CCP STRUCTURED FIELD** *identifier2* **WAS NOT FOUND.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Next Conditional Processing Control Identifier parameter in the CCP structured field specifies the identifier used to locate a CCP if the CCP structured fields are chained. The identifier must match a value specified in the CCP Identifier parameter of another CCP within the same page definition. The identifier specified in the Next CCP Identifier parameter did not match the CCP Identifier of any CCPs in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-321 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE TIMING OF ACTION PARAMETER VALUE** *value* **IN CCP STRUCTURED FIELD** *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Timing of Action parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input

file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-322 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE MEDIUM MAP ACTION PARAMETER VALUE** *value* **IN CCP STRUCTURED FIELD** *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Medium Map Action parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-323 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA MAP ACTION PARAMETER VALUE** *value* **IN CCP STRUCTURED FIELD** *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Data Map Action parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-324 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE COMPARISON PARAMETER VALUE** *value* **IN CCP STRUCTURED FIELD** *CCP identifier* **IS INVALID.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Comparison parameter in one of the repeating groups of the CCP structured field contains an invalid value. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-326 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA MAP** *data map name* **SPECIFIED IN THE DATA MAP NAME PARAMETER OF CCP STRUCTURED FIELD** *CCP identifier* **WAS NOT FOUND.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Data Map Name parameter in one of the repeating groups of the CCP structured field specifies the token name of a data map used to locate a data map in the page definition. The name must match the value specified in the Token Name parameter in one of the Begin Data Map (BDM) structured fields in the current page definition. No data map with name *data map name* was found in the page definition.

System Action: ACIF stops processing the input

file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-327 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF REUSING DATA PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **WILL CAUSE AN INFINITE LOOP.**

Explanation: The Next Line Descriptor If Reusing Data parameter in the Line Descriptor (LND) structured field identified in this message caused an infinite-loop condition. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-329 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF REUSING DATA PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **IS** *parameter value1***. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF** *parameter value2***.**

Explanation: The Next Line Descriptor If Reusing Data parameter in the Line Descriptor (LND) structured field identified in this message has an invalid value. The value is greater than the Count parameter in the Line Descriptor Count (LNC) structured field in the current data map. The LNC and LND structured fields are contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-330 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA START POSITION PARAMETER VALUE WHEN ADDED TO THE DATA LENGTH PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **EXCEEDS THE FDS STRUCTURED FIELD SIZE PARAMETER VALUE OF** *parameter value***.**

Explanation: The Use Fixed Data flag in byte 0 in the Line Descriptor (LND) structured field was set to B'1'. This indicates that data from Fixed Data Text (FDX) structured fields is to be added to the data placed within the page by the LND structured field The FDX and LND structured fields are contained in the page definition.

The Data Start Position parameter in the LND structured field indicates the offset of the first byte of data. The Data Length parameter indicates the number of bytes of FDX data to be placed within the page. This error was caused when these two parameters specified more data than was contained in the FDX structured fields. The number of bytes of data in the FDX structured fields can be found in the Size parameter of the Fixed Data Size (FDS) structured field.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM

licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-334 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 input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-335 DATA IN A PAGEDEF RESOURCE IS INVALID: THE MEDIUM MAP *medium map name* **SPECIFIED IN THE MEDIUM MAP NAME PARAMETER OF CCP STRUCTURED FIELD** *CCP identifier* **WAS NOT FOUND.**

Explanation: The Conditional Processing Control (CCP) structured field has an invalid value. The Medium Map Name parameter in one of the repeating groups of the CCP structured field specifies the token name of a medium map used to locate a medium map in the form definition. The name must match the value specified in the Token Name parameter in one of the Begin Medium Map (BMM) structured fields in the current form definition. No medium map with name *medium map name* was found in the form definition. The CCP structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-337 DATA IN A PAGEDEF RESOURCE IS INVALID: IN LND STRUCTURED FIELD NUMBER *structured field number***, THE CONDITIONAL PROCESSING FLAG WAS SET BUT THE CONDITIONAL PROCESSING CONTROL IDENTIFIER PARAMETER WAS ZERO.**

Explanation: In the Line Descriptor (LND) structured field, the Conditional Processing flag had a value of B'1', indicating that the data to be processed by this LND structured field is to be compared with a value specified in a Conditional Processing Control (CCP) structured field. The CCP Identifier parameter in the LND structured field is used to locate one of the CCP structured fields in the current page definition. This Identifier parameter was set to 0, which is not a valid value if the Conditional Processing flag is on. The LND and CCP structured fields are contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-339 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE IDENTIFIER** *identifier* **SPECIFIED IN THE CONDITIONAL PROCESSING CONTROL IDENTIFIER PARAMETER IN LND STRUCTURED FIELD NUMBER** *structured field number* **WAS NOT FOUND.**

Explanation: In the Line Descriptor (LND) structured field, the Conditional Processing flag had a value of B'1', indicating that the data to be processed by this LND structured field is to be compared with a value specified in a Conditional Processing Control (CCP) structured field. The CCP Identifier parameter in the LND structured field is used to locate one of the CCP structured fields in the current page definition. The identifier specified in *identifier* does not match the value specified in the CCP Identifier parameter in any of the CCP structured fields in the current page definition. The LND and CCP structured fields are contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-340 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **IS** *value1***. THIS EXCEEDS THE LNC STRUCTURED FIELD COUNT VALUE OF** *value2***.**

Explanation: The Next Line Descriptor If Conditional Processing parameter in the Line Descriptor (LND) structured field has an invalid value. The value is greater than the Count parameter in the Line Descriptor Count (LNC) structured field in the current data map. The LNC and LND structured fields are contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-342 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF CONDITIONAL PROCESSING PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **WILL CAUSE AN INFINITE LOOP.**

Explanation: The Next Line Descriptor If Conditional Processing parameter in the Line Descriptor (LND) structured field will cause an infinite-loop condition. The LND structured field is contained in the page definition.

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 process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-343 **DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS INVALID: AN LND STRUCTURED FIELD THAT USES RELATIVE POSITIONING ATTEMPTED TO PLACE DATA OUTSIDE OF THE LOGICAL PAGE IN THE Y DIRECTION. THE PRIOR AND CURRENT LND NUMBERS ARE:** *prior LND number* **AND** *current LND number***.**

Explanation: When relative positioning is being used on a Line Descriptor (LND) structured field, the relative position specified for the Y direction can be a negative value. The current LND is trying to position data outside of the logical page using a negative value. The prior LND position defines the baseline position from which the relative offset of the current LND is measured.

System Action: ACIF stops processing the input file and issues a message.

User Response: Correct the ACIF input file or page definition and submit the job to ACIF again. For information about structured fields of a page definition, refer to the *Advanced Function Presentation Programming Guide and Line Data Reference*.

0425-344 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NUMBER OF LND STRUCTURED FIELDS DOES NOT MATCH THE VALUE SPECIFIED IN THE LNC STRUCTURED FIELD.**

Explanation: The number of Line Descriptor (LND) structured fields found in a page definition is either greater than or less than the value specified in the Line Descriptor Count (LNC) structured field. The LND and LNC structured fields are contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-345 **DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS INVALID: AN LND STRUCTURED FIELD THAT USES RELATIVE POSITIONING CONTAINS AN ORIENTATION THAT IS DIFFERENT THAN THE LAST LND USED FOR POSITIONING. THE PRIOR AND CURRENT LND NUMBERS ARE:** *prior LND number* **AND** *current LND number***.**

Explanation: When relative positioning is being used on a Line Descriptor (LND) structured field, the text orientation field of the current LND must match the text orientation field of the LND that was last used for positioning data. The prior LND position defines the baseline position from which the relative offset of the current LND is measured.

System Action: ACIF stops processing the input file and issues a message.

User Response: Correct the ACIF input file or page definition and submit the job to ACIF again. For information about structured fields of a page definition, refer to the *Advanced Function Presentation Programming Guide and Line Data Reference*.

0425-346 **DATA IN AN INPUT RECORD OR PAGEDEF RESOURCE IS INVALID: A SKIP TO A NONEXISTENT CHANNEL =** *channel* **ON RECORD NUMBER =** *record number* **WAS DETECTED WITHIN THE LND STRUCTURED FIELDS. OUTPUT WAS FORCED TO SINGLE SPACING, WHICH MAY CAUSE BLANK PAGES.**

Explanation: An attempt was made to skip to a channel not defined in the current data map. The

Line Descriptor (LND) structured fields in the page definition are invalid. During scanning, the entire Next Line Descriptor If Skipping parameter could not be followed because an LND had the End Page If Skipping flag set. This created an infinite loop on the same input record. The LND structured field is contained in the page definition.

System Action: The record containing the error was forced to single spacing. When forced single spacing occurs, the carriage control character on the record is ignored. The record is treated as if a X'09' machine control character or a X'40' ANSI control character was specified in the record that caused the error.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-348 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE NEXT LINE DESCRIPTOR IF SPACING PARAMETER VALUE** *value* **IN LND STRUCTURED FIELD NUMBER** *structured field number* **IS INVALID.**

Explanation: The logical-record control character had indicated that the Next Line Descriptor If Spacing parameter should be followed. However, in the Line Descriptor (LND) structured field identified by *structured field number*, the Next Line Descriptor If Spacing parameter value was either zero or greater than the total number of Line Descriptors in the data map. The LND structured field is contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-353 **DATA IN A PAGEDEF RESOURCE IS INVALID: THE DATA LENGTH PARAMETER VALUE IN LND STRUCTURED FIELD NUMBER** *structured field number* **DOES NOT MATCH THE LENGTH OF COMPARISON STRING PARAMETER VALUE IN CCP STRUCTURED FIELD** *CCP identifier***.**

Explanation: In the Line Descriptor (LND) structured field, the value of the Data Length parameter is used in identifying the field of the current input record for which conditional processing is to be performed. This field is to be compared with the Comparison String specified in the Conditional Processing Control (CCP) structured field. The length specified in the Data Length parameter in the LND structured field does not match the length specified in the Length of Comparison String parameter of the CCP structured field. The LND and CCP structured fields are contained in the page definition.

System Action: ACIF stops processing the input file and issues a message identifying the position of the structured field in the file or resource.

User Response: Correct the process used to create the page definition. If you used an IBM licensed program to create the page definition with the error, use local problem-reporting procedures to report this message.

0425-400 **THE** *parameter* **NUMBER VALUE IS NOT NUMERIC.**

Explanation: A numeric value must be specified after the parameter.

System Action: ACIF terminates.

User Response: Use a numeric value after the parameter and rerun ACIF.

0425-401 **THE** *parameter* **NAME MUST BE DELIMITED WITH QUOTES.**

Explanation: The attribute name of a parameter must begin and end with single quotes.

System Action: ACIF terminates.

User Response: Use single quotes before and after the attribute name in the parameter.

0425-402 **THE PARAMETER** *'xxxxxxxx'* **IS INVALID.**

Explanation: A parameter that is not valid for ACIF was specified.

System Action: ACIF terminates.

User Response: Correct the parameter and rerun ACIF.

0425-403 **THE REQUESTED RESOURCE** *type* **IS UNKNOWN.**

Explanation: A resource I/O has been requested, but the resource *number* is unknown to ACIF. This condition is caused by an ACIF logic error. The resource type codes are listed below:

Type	Resource Name
1	Print input file
2	FORMDEF file
3	PAGEDEF file
4	OVERLAY file
5	SEGMENT file
6	Coded FONT file
7	Code PAGE file
8	FONT Character Set file
9	FONT Metric file
10	FONT Shape file
20	Print output file
21	Messages output file
22	SPOOL file
23	Dummy input file
24	Dummy output file
25	Parameter file
26	Resource Object File

System Action: ACIF terminates.

User Response: Use local problem-reporting procedures to report this message.

0425-404 **THE ATTRIBUTE NAME USED IN** *indexn* **HAS AN IMPROPER USE OF QUOTES.**

Explanation: An unpaired set of quotes was found in the attribute name for an **index** parameter.

System Action: ACIF terminates.

User Response: Correct the **index** parameter and rerun ACIF.

0425-405 **A VALUE OF** *'xxxxxxxx'* **IS INVALID FOR PARAMETER** *'xxxxxxxx'***.**

Explanation: The value supplied for a parameter is invalid.

System Action: ACIF terminates.

User Response: Correct the parameter value and rerun ACIF.

0425-407 **A RESTYPE PARAMETER OF '***xxxx***' IS NOT VALID.**

Explanation: Either an unsupported value was specified, or a resource type of **none** was found with another value in the **restype** parameter. Examples of other values are: **font**, **ovly**, **fdef**, or **pseg**. A resource type of **none** cannot be specified with another value.

System Action: ACIF terminates.

User Response: Correct the **restype** parameter and rerun ACIF.

0425-408 **A VIRTUAL STORAGE REQUEST WAS UNSUCCESSFUL - REQUEST SIZE** *storage request size, request type* **RETURN CODE** *return code***.**

Explanation: A GETMAIN/FREEMAIN macro made an unsuccessful attempt to obtain virtual storage. This message indicates the storage size and the return code from the system GETMAIN macro.

System Action: ACIF terminates.

User Response: Decrease the load on the system and rerun ACIF.

0425-410 **AN ACIF STORAGE REQUEST WAS UNSUCCESSFUL - REQUEST SIZE** *storage request size*, *request type* **RETURN CODE** *return code*.

Explanation: An unsuccessful attempt has been made to obtain/free ACIF storage. This error message returns the following information:

- Storage request size
- Request type
- Return code

System Action: ACIF terminates.

User Response: Use the information provided in the message to correct the error and rerun ACIF.

0425-412 **MODULE** *module name* **HAS RETURNED WITH RETURN CODE** *return code.*

Explanation: A non-zero return code has been returned from the called module. This message indicates that an abnormal occurrence has taken place in the called module. This message is informational and further action will take place in higher level modules if required.

System Action: None; this message is for information only.

User Response: See the accompanying message to determine a response.

0425-413 **ATTEMPTED {OPEN|CLOSE|READ|WRITE} OF RESOURCE FILE** *'filename'*, **RESOURCE NAME** *'resname'* **FAILED. RETURN CODE** *return code.*

Explanation: An attempt to open, close, read, or write a resource failed. This message indicates that an abnormal occurrence has taken place in the called module. This message is informational and further action will take place in higher level modules if required.

Return Codes	Meaning
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Invalid data buffer address
4	Address not word aligned
6	Invalid FILE_CB@
7	Invalid MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Invalid record number
12	End of file detected
13	Disk is full
14	RECFM invalid
20	Invalid file id
28	File not found
51	Length exceeds maximum
310	File format invalid. The user should ensure that the CC and FILEFORMAT parameters are coded correctly. This return code indicates that ACIF was unable to determine where one record ended and another one began. Refer to "Chapter 7. ACIF Helpful Hints" on page 167 for more information.

System Action: None; this message is for information only.

User Response: See the accompanying message to determine a response.

0425-415 *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.

0425-418 THE MAXIMUM RECORD ID WAS EXCEEDED.

Explanation: The current report file contains more than 99,999,999 documents.

System Action: ACIF terminates.

User Response: Break the report file up into a smaller number of documents.

0425-419 USER {INPUT|OUTPUT|RESOURCE} EXIT *program* RETURNED CODE *nnnn*.

Explanation: The indicated user exit program has returned a non-zero return code.

System Action: ACIF terminates.

User Response: Correct the error in the exit program and rerun ACIF.

0425-420 AN ERROR OCCURRED WHILE ATTEMPTING TO OPEN *'file'* RETURN CODE *nnn*

Explanation: An attempt to open a file failed. This message is informational and further action will take place in higher level modules if required.

Return Codes	Meaning
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Invalid data buffer address
4	Address not word aligned
6	Invalid FILE_CB@
7	Invalid MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Invalid record number
12	End of file detected
13	Disk is full
14	RECFM invalid
20	Invalid file id
28	File not found
32	On UNIX, the ACIF message catalog was not found in paths specified by the NLSPATH environment variable. ACIF will use the default message catalog file /usr/lib/nls/msg/C/arsacif.cat.
36	On UNIX, the default message catalog is not accessible. Check permissions.
51	Length exceeds maximum
310	File format invalid

System Action: ACIF terminates.

User Response: Use the information provided in the return code to correct the problem.

0425-421 AN ERROR OCCURRED WHILE ATTEMPTING TO CLOSE *'file'* RETURN CODE *nnn*

Explanation: An attempt to close a file failed. This message is informational and further action will take place in higher level modules if required.

Return Codes	Meaning
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Invalid data buffer address
4	Address not word aligned
6	Invalid FILE_CB@
7	Invalid MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Invalid record number

Return Codes	Meaning
12	End of file detected
13	Disk is full
14	RECFM invalid
20	Invalid file id
28	File not found
32	On UNIX, the ACIF message catalog was not found in paths specified by the NLSPATH environment variable. ACIF will use the default message catalog file /usr/lib/nls/msg/C/arsacif.cat.
36	On UNIX, the default message catalog is not accessible. Check permissions.
51	Length exceeds maximum
310	File format invalid

System Action: ACIF terminates.

User Response: Use the information provided in the return code to correct the problem.

0425-422 AN ERROR OCCURRED WHILE ATTEMPTING TO READ *'file'* RETURN CODE *nnn*

Explanation: An attempt to read a file failed. This message is informational and further action will take place in higher level modules if required.

Return Codes	Meaning
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Invalid data buffer address
4	Address not word aligned
6	Invalid FILE_CB@
7	Invalid MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type
10	Storage allocation/deallocation failed
11	Invalid record number
12	End of file detected
13	Disk is full
14	RECFM invalid
20	Invalid file id
28	File not found
32	On UNIX, the ACIF message catalog was not found in paths specified by the NLSPATH environment variable. ACIF will use the default message catalog file /usr/lib/nls/msg/C/arsacif.cat.
36	On UNIX, the default message catalog is not accessible. Check permissions.
51	Length exceeds maximum
310	File format invalid

System Action: ACIF terminates.

User Response: Use the information provided in the return code to correct the problem.

0425-423 AN ERROR OCCURRED WHILE ATTEMPTING TO WRITE *'file'* RETURN CODE *nnn*

Explanation: An attempt to write a file failed. This message is informational and further action will take place in higher level modules if required.

Return Codes	Meaning
0	Successful
1	Permanent I/O error
2	Specified number of bytes is zero or negative
3	Invalid data buffer address
4	Address not word aligned
6	Invalid FILE_CB@
7	Invalid MODE parameter
8	Data record longer than LRECL or buffer
9	File is not supported type

Return Codes	Meaning
10	Storage allocation / deallocation failed
11	Invalid record number
12	End of file detected
13	Disk is full
14	RECFM invalid
20	Invalid file id
28	File not found
32	On UNIX, the ACIF message catalog was not found in paths specified by the NLSPATH environment variable. ACIF will use the default message catalog file /usr/lib/nls/msg/C/arsacif.cat.
36	On UNIX, the default message catalog is not accessible. Check permissions.
51	Length exceeds maximum
310	File format invalid

System Action: ACIF terminates.

User Response: Use the information provided in the return code to correct the problem.

0425-424 PARAMETER 'RESFILE=PDS' IS ONLY VALID UNDER MVS, DEFAULTING TO 'RESFILE=SEQ'.

Explanation: The supplied value for the RESFILE parameter is incorrect for the operating system.

System Action: ACIF produces a sequential resource file.

User Response: No response is necessary.

0425-425 USER *xxxxxx* EXIT *'xxxxxxxx'* WAS NOT FOUND.

Explanation: The user exit program named on the exit's DD parameter does not exist.

System Action: ACIF terminates.

User Response: Verify the name of the user exit program and that the user exit program resides in a directory searched by ACIF.

0425-428 A *'resource'* HAS BEEN REQUESTED, BUT NO NAME WAS GIVEN.

Explanation: The resource listed in the message was requested to be handled by ACIF, but the name to get was not passed to ACIF. This condition is caused by an ACIF logic error.

System Action: ACIF terminates.

User Response: Use local problem-reporting procedures to report this message.

0425-435 THE *ddname* DD STATEMENT SPECIFIED FOR *parameter* IS MISSING.

Explanation: An ACIF DD parameter specified a DDname that was not specified in the JCL (MVS and VSE) or FILEDEF statement (VM).

System Action: ACIF terminates.

User Response: Ensure that the ACIF parameter specifies a DDname that is defined in the job commands.

0425-436 THE GROUPNAME VALUE *'value'* IS NOT WITHIN THE ALLOWABLE RANGE.

Explanation: ACIF processing has encountered the GROUPNAME parameter with an invalid INDEX number specified. The INDEX range is 1–32.

System Action: ACIF terminates.

User Response: Correct the parameter and rerun ACIF.

0425-437 '(TYPE=FLOAT)' MAY NOT BE SPECIFIED FOR TRIGGER1.

Explanation: The 'TYPE=FLOAT' subparameter is invalid for TRIGGER1.

System Action: ACIF terminates.

User Response: Correct the parameter and rerun ACIF.

0425-440 ACIF HAS COMPLETED PROCESSING NORMALLY WITH RETURN CODE *nn*.

Explanation: ACIF processing has completed with the return code shown.

System Action: This message is for information only.

User Response: See any accompanying messages to determine a response.

0425-441 ACIF HAS COMPLETED PROCESSING ABNORMALLY WITH RETURN CODE *nn*, REASON CODE nn.

Explanation: ACIF processing has completed with the return code and reason code shown. The reason codes are for the use of IBM service personnel only. The return codes for ACIF as follows:

Return Code	Meaning	ACIF Action
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.

0425-443 A BEGIN COLUMN SPECIFICATION FOR FIELDnn IS <= 0. SUCH A SPECIFICATION IS ONLY VALID WHEN 'BASE=TRIGGER' IS ALSO SPECIFIED.

Explanation: FIELDnn 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 terminates.

User Response: Correct the ACIF FIELDnn parameter specification and resubmit the job.

0425-444 MULTIPLE COLUMNS WERE SPECIFIED FOR FIELDnn WHICH IS DEFINED WITH 'BASE=TRIGGER.' ONLY ONE COLUMN MAY BE SPECIFIED WHEN A FIELD IS DEFINED WITH 'BASE=TRIGGER.'

Explanation: FIELDnn was specified with multiple columns and BASE=TRIGGER. Only one column may be specified for a field that is also specified with BASE=TRIGGER.

System Action: ACIF terminates.

User Response: Correct the ACIF FIELDnn parameter specification and resubmit the job.

0425-445 INDEXnn WHICH IS DEFINED AS EITHER 'TYPE=PAGERANGE' OR 'TYPE=GROUPRANGE' INCLUDES FIELDnn WHICH IS DEFINED AS 'BASE=TRIGGER.' THIS COMBINATION IS INVALID.

Explanation: INDEXnn was specified as TYPE=PAGERANGE or TYPE=GROUPRANGE with a FIELDnn that was defined as BASE=TRIGGER. This combination is not supported.

System Action: ACIF terminates.

User Response: Correct the ACIF parameters and resubmit the job.

0425-446 **USE OF FIELDnn BY INDEXnn 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 INDEXnn which is defined as either TYPE=PAGERANGE or TYPE=GROUPRANGE. This is invalid.

System Action: ACIF terminates.

User Response: Correct the ACIF parameters and resubmit the job.

0425-447 **THE LENGTH, *x*, OF OFFSET PAIR *n* FOR FIELD*m* DOES NOT EQUAL THE LENGTH, *y*, SPECIFIED FOR FIELD*m*.**

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 invalid. The lengths must be equal.

System Action: ACIF stops processing.

User Response: Correct the ACIF parameters and resubmit the job to ACIF.

0425-448 **INDEXING WAS REQUESTED, BUT NEITHER 'TRIGGER1' NOR ANY 'FIELD' WAS SATISFIED WITHIN THE PAGE RANGE SPECIFIED BY THE INDEXSTARTBY PARAMETER.**

Explanation: Indexing was requested, but TRIGGER1 was outside the range of pages specified in the **indexstartby** parameter.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-449 **INDEX FIELDS REFERENCE OUTSIDE OF THE RECORD, FIELD# *nn* INPUT RECORD# *nnnnnn***

Explanation: The **field** value specified on the **index** parameter references an area that is outside the length of the requested record.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-450 **A REQUIRED ACIF PARAMETER *parameter name* WAS NOT FOUND IN THE PARAMETER FILE.**

Explanation: A required ACIF parameter was not found in the parameter file.

System Action: ACIF terminates.

User Response: Add the missing parameter to the parameter file and rerun ACIF.

0425-452 **A TRIGGER NUMBER OF *nnn* IS INVALID FOR field*n*.**

Explanation: The trigger number specified in the field parameter is invalid.

System Action: ACIF terminates.

User Response: Triggers used in field definitions must be defined. Correct the parameter and rerun ACIF.

0425-453 **THE *xxxxxxxxnn* LENGTH OF *nnnn* IS GREATER THAN THE ALLOWED MAXIMUM OF *nnnn*.**

Explanation: The combined length of all of the **field** values on an **index** parameter is too long.

System Action: ACIF terminates.

User Response: Check the **field** and **index** parameters to find where this happens. Correct the parameter and rerun ACIF.

0425-454 **A VALUE OF *nnn* IS INVALID FOR *xxxxxnn***

Explanation: A parameter value contains invalid characters.

System Action: ACIF terminates.

User Response: Correct the parameter value and rerun ACIF.

0425-455 **FIELDnn USED BY INDEXnn WAS NOT DEFINED.**

Explanation: An INDEX parameter referred to a FIELD that was not defined in the parameter file.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-456 **THE TRIGGER1 RELATIVE RECORD NUMBER IS NOT EQUAL TO ASTERISK.**

Explanation: The record number associated with the **trigger1** parameter was not an asterisk.

System Action: ACIF terminates.

User Response: Correct the parameter and rerun ACIF.

0425-457 **TRIGGER1 WAS NOT DEFINED, BUT SECONDARY TRIGGERS ARE PRESENT.**

Explanation: **TRIGGER1** must be specified if secondary triggers are present.

System Action: ACIF terminates.

User Response: If no indexing is required, delete all **trigger** parameters from the parameter file, otherwise supply a **trigger1** parameter for this ACIF job.

0425-458 **A NON-LITERAL VALUE OF *xxxxxxxx* HAS BEEN SUPPLIED FOR *xxxxxxnn***

Explanation: The supplied **TRIGGER** value was not a literal.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-460 **TRIGGERS SATISFIED, BUT INDEXES WERE INCOMPLETE AT END-OF-FILE.**

Explanation: The **trigger** parameters specified in the parameter file were met, but the end-of-file was reached before the **index** parameters were located.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-461 **TRIGGER SUPPLIED, BUT ALL INDEX VALUES WERE LITERALS.**

Explanation: A **trigger** value has been supplied, but all **index** values were literals.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-462 **A TRIGGER PARAMETER WAS SPECIFIED, BUT THE INPUT FILE IS ALREADY INDEXED.**

Explanation: The parameter file included a **trigger** parameter, but the input file contains indexing structured fields. ACIF cannot index a file that is already indexed.

System Action: ACIF terminates.

User Response: If you want to create an index object file for the input file, remove all **trigger** parameters from the ACIF parameter file and rerun ACIF.

0425-463 **THE INDEX SPECIFIED BY THE GROUPNAME PARAMETER WAS NOT DEFINED OR WAS INVALID.**

Explanation: The INDEX specified by the GROUPNAME parameter was not defined or the INDEX contained a FIELD that was based on either a floating or recordrange trigger. This is invalid.

System Action: ACIF terminates.

User Response: Correct the parameter and resubmit ACIF.

0425-464 *'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 resubmit the ACIF.

0425-465 **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 resubmit ACIF.

0425-466 **A SUB-PARAMETER OF** *'sub-parameter'* **IS NOT SUPPORTED ON THE** *'parameter'* **PARAMETER.**

Explanation: The named sub-parameter 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 resubmit ACIF.

0425-467 **THE NUMBER** *'number'* **IS NOT SUPPORTED FOR {FIELD | INDEX | TRIGGER}.**

Explanation: An invalid number was specified on a FIELD, INDEX, or TRIGGER 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 resubmit ACIF.

0425-468 **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 terminates.

User Response: Use your local problem reporting system to report the error.

0425-469 **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.

0425-470 WHICH BEGINS AT OFFSET *offset* FOR A LENGTH OF *length*.

Explanation: This message is issued following a message which contains the cause of the error.

System Action: See the preceding message.

User Response: See the preceding message.

0425-471 THE NUMBER OF FIELD VALUES ON THE INDEX PARAMETER EXCEEDED THE MAXIMUM ALLOWED.

Explanation: There were too many field values specified for the INDEX 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 FIELDnn values from the INDEX parameter and resubmit ACIF.

0425-472 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 resubmit ACIF.

0425-473 RECORDRANGE SUB-PARAMETER ALLOWED ONLY IF RECORD VALUE IS '*'.

Explanation: The RECORDRANGE sub-parameter is only valid on a TRIGGER parameter if the record value was specified as '*'.

System Action: ACIF continues processing the parameter file, but does not process the report file.

User Response: Either specify an '*' for the record value or remove the RECORDRANGE sub-parameter from the TRIGGER parameter.

0425-474 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 terminates.

User Response: Ensure the literal value is enclosed in quotes and resubmit ACIF.

0425-475 THE HEX STRING *'hex string'* 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 resubmit ACIF.

0425-476 MESSAGE TEXT NOT AVAILABLE FOR MESSAGE NUMBER: nnnnnnnn

Explanation: ACIF attempted to write a message that is not defined in the catalog.

System Action: ACIF continues processing, depending on the significance of the undefined message.

User Response: Contact IBM service and inform them that ACIF attempted to write an undefined message. The situation should be corrected by IBM.

0425-499 INTERNAL ERROR IN MODULE _____ AT FUNCTION _____.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error. Make a note of the module and function specified in the message.

0425-532 A FORM/PAGE DEFINITION WITH A MEMBER NAME nnnnnnnn WAS NOT FOUND - RETURN CODE NN, REASON CODE NN.

Explanation: The requested page or form definition does not exist in any of the available paths.

System Action: ACIF terminates.

User Response: Correct the parameters and rerun ACIF.

0425-900 MISSING DAT POINTER IN CCM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-901 MISSING FORMDEF POINTER IN CCM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-902 MISSING PAGEDEF POINTER IN CCM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-903 MISSING OBJECT STACK POINTER IN CCM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-904 MISSING CODE PAGE POINTER IN CCM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-905 MISSING FONT METRIC POINTER IN CCM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-906 UNEXPECTED OTHERWISE STATEMENT ENCOUNTERED.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-907 CCM CANNOT FIND REQUESTED MEDIUM MAP.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-908 CCM CANNOT FIND REQUESTED DATA MAP.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-909 CCM CANNOT FIND REQUESTED MEG.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-910 INPUT BIN LIST CHANGED DURING PROCESSING.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-911 DAT DID NOT SPECIFY ANY INPUT BIN INFORMATION.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-912 OVERLAY LOCAL ID HAS BEEN CHANGED IN LIST.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-913 STARTING COPY COUNT EXCEEDS TOTAL COPIES IN MM.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-914 CONDITIONAL PROCESSING INFORMATION PASSED TO CCM AT DOCUMENT INTERFACE BUT PAGEDEF DOES NOT REQUEST CONDITIONAL PROCESSING.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-915 ACIF REQUESTED CODE PAGE DEALLOCATION AS WELL AS CODE PAGE PROCESSING.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-916 ACIF REQUESTED ACTIVATION OF AN OUTLINE FONT CHARACTER SET, BUT DOES NOT SUPPORT OUTLINE FONTS.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-917 ACIF REQUESTED ACTIVATION OF A FONT RESOURCE BUT THE GLOBAL NAME WAS NOT PROVIDED OR HAD AN INCORRECT LENGTH.

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-918 NO FREQUENT FONT TABLE OR FGID LOOK ASIDE TABLE WAS PROVIDED TO *modulename***.**

Explanation: An internal error occurred in ACIF.

System Action: ACIF terminates.

User Response: Contact IBM service and inform them that you have received this message indicating an internal error.

0425-919 THE CCM COMPONENT OF ACIF HAS USED UP ITS OBJECT STACK IN *modulename***.**

Explanation: The 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 terminates.

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 / end structured field requirements, this message indicates an internal logic error. Contact IBM service and inform them that you have received this message indicating an internal error.

Chapter 5. ACIF User Exits and Attributes of the Input Print File

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

This section describes the following topics:

User programming exits

Non-zero return codes

Attributes of the input print file

User Programming Exits

OnDemand provides several sample programming exits to assist you in customizing ACIF. Use of the programming exits is optional. You specify the names of the exit programs with the **inpexit**, **indxexit**, **outexit**, and **resexit** parameters. Each of these parameters is described in "Chapter 3. ACIF Parameter Reference" on page 61.

Important: OnDemand provides compiled versions of the sample user exit programs. If you make changes to the sample user exit programs or create your own user exit programs, you must compile them before the programs can be used by ACIF.

Because the header files for the user exit programs can change between releases and PTFs, we strongly encourage you to recompile all user exit programs after upgrading the ACIF component of OnDemand. Failure to do so may cause unexpected results when indexing data with ACIF.

See the ACIF README file on the OnDemand server CD-ROM for more information about supported compiliers and files provided by OnDemand to help you with programming the user exits.

OnDemand provides the following ACIF sample exits:

apkinp.c
Input record exit

apkind.c
Index record exit

apkout.c
Output record exit

apkres.c
Resource exit

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

apka2e.c
Converts ASCII stream data to EBCDIC stream data.

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

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

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

apkexits.h

Input Record Exit

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

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

In a MO:DCA-P document, indexing information can be passed in the form of a Tag Logical Element (TLE) structured field. For more information about the TLE structured field, see “Chapter 8. ACIF Data Stream Information” on page 177. The exit program can create these structured fields while ACIF is processing the print file. This is an alternative to modifying the application in cases where the indexing information is not consistently present in the application output.

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

Figure 17 contains a sample C language header that describes the control block that is passed to the exit program.

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

Figure 17. Sample Input Record Exit C Language Header

Figure 18 contains a sample DSECT that describes the control block for the exit program.

```
PARMLIST DSECT          Parameters for the input record exit
WORK@    DS    A        Address of 16-byte static work area
PFATTR@  DS    A        Address of print-file-attribute information
RECORD@  DS    A        Address of the input record
         DS    A        Reserved for future use
RECORDLN DS    H        Length of the input record
         DS    H        Reserved for future use
REQUEST  DS    X        Add, delete, or process the record
EOF      DS    C        EOF indicator
```

Figure 18. Sample Input Record Exit DSECT

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

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 163 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)
A pointer to the first byte of the input record including the carriage control character. The record resides in a buffer that resides in storage allocated by ACIF, but the exit program is allowed to modify the input record.

reserved1 (Bytes 13–16)
These bytes are reserved for future use.

recordln (Bytes 17–18)
Specifies the number of bytes (length) of the input record. If the input record is modified, this parameter must also be updated to reflect the actual length of the record.

reserved2 (Bytes 19–20)
These bytes are reserved for future use.

request (Byte 21)
Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00', X'01', or X'02', where:

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

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

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

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

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

eof (Byte 22)
An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that specifies whether an **eof** condition has been encountered. When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the input exit, and the input file has been closed. The record pointer is no longer valid. Records may not be inserted when **eof** is signaled. The following are the only valid values for this parameter:

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

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

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

Using the ACIF User Input Record Exits

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

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

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

Also, ensure that the directory where the **apka2e** input record exit program resides is included in the **PATH** environment variable (or specify the full path name).

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

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

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

Also, ensure that the directory where the **asciinp** input record exit program resides is included in the **PATH** environment variable (or specify the full path name).

The **asciinpe** input record exit program combines both user input record exits described above. To execute, specify `inpexit=asciinpe` and follow the directions specified for both **apka2e** and **asciinp**. Also, ensure that the

directory where the **asciinpe** input record exit program resides is included in the **PATH** environment variable (or specify the full path name).

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

backspacing (X'08')

horizontal tabs (X'09')

vertical tabs (X'0B')

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

Index Record Exit

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

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

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

```
typedef struct _INDXEXIT_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 index record                  */
   char           request;    /* Delete or process the record                */
   char           eof;        /* Last call indicator to ACIF                 */
} INDXEXIT_PARMS;
```

Figure 19. Sample Index Record Exit C Language Header

Figure 20 on page 157 contains a sample DSECT that describes the control block that is passed to the exit program.

```
PARMLIST DSECT          Parameters for the index 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 index record
REQUEST  DS    X        Delete or process the record
EOF      DS    C        Last call indicator to ACIF
```

Figure 20. Sample Index Record Exit DSECT

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

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. See "Attributes of the Input Print File" on page 163 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)
A pointer to the first byte of the index record including the carriage control character. The record resides in a 32KB (where KB equals 1024 bytes) buffer. The buffer resides in storage allocated by ACIF, but the exit program is allowed to modify the index record.

recordln (Bytes 13–14)
Specifies the length, in bytes, of the index record. If the index record is modified, this parameter must also be updated to reflect the actual length of the record.

request (Byte 15)
Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01' where:

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

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

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

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

eof (Byte 16)
An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that signals when ACIF has finished processing the index object file.

When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the index exit. The record pointer is no longer valid. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

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

Output Record Exit

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

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

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

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

Figure 21. Sample Output Record Exit C Language Header

Figure 22 on page 159 contains a sample DSECT that describes the control block passed to the exit program.

```
PARMLIST DSECT          Parameters for the output record exit
WORK@    DS    A        Address of 16-byte static work area
PFATTR@  DS    A        Address of print-file-attribute information
RECORD@  DS    A        Address of the record to be written
RECORDLN DS    H        Length of the output record
REQUEST  DS    X        Delete or process the record
EOF      DS    C        Last call indicator
```

Figure 22. Sample Output Record Exit DSECT

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

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. See "Attributes of the Input Print File" on page 163 for more information on the format of this data structure and the information contained in it.

record (Bytes 9–12)
A pointer to the first byte of the output record. The record resides in a 32KB (where KB equals 1024 bytes) buffer. The buffer resides in storage allocated by ACIF, but the exit program is allowed to modify the output record.

recordln (Bytes 13–14)
Specifies the length, in bytes, of the output record. If the output record is modified, this parameter must also be updated to reflect the actual length of the record.

request (Byte 15)
Specifies how the record is to be processed by ACIF. On entry to the exit program, this parameter is X'00'. When the exit program returns control to ACIF, this parameter must have the value X'00' or X'01', where:

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

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

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

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

eof (Byte 16)
An End-Of-File (**eof**) indicator. This indicator is a 1-byte character code that signals when ACIF has finished writing the output file.

When **eof** is signaled (**eof** value='Y'), the last record has already been presented to the output exit. The record pointer is no longer valid. The following are the only valid values for this parameter:

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

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

Resource Exit

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

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

Figure 23 contains a sample C language header that describes the control block that is passed to the exit program.

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

Figure 23. Sample Resource Exit C Language Header

Figure 23 contains a sample DSECT that describes the control block that is passed to the exit program.

```
PARMLIST DSECT          Parameters for the resource exit
WORK@    DS    A        Address of 16-byte static work area
PFATTR@  DS    A        Address of print-file-attribute information
RESNAME  DS    CL8      Name of requested resource
RESTYPE  DS    X        Type of resource
REQUEST  DS    X        Ignore or process the resource
EOF      DS    X
```

Figure 24. Sample Resource Exit DSECT

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

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. See “Attributes of the Input Print File” on page 163 for more information on the format of this data structure and the information presented.

resname (Bytes 9–16)
Specifies the name of the requested resource. This value cannot be modified (changed) by the exit program.

restype (Byte 17)
Specifies the type of resource the name refers to. This is a 1-byte hexadecimal value where:

X’03’ Specifies a GOCA (graphics) object.

X’05’ Specifies a BCOCA (barcode) object.

X'06' Specifies an IOCA (IO image) object.

X'40' Specifies a font character set.

X'41' Specifies a code page.

X'FB' Specifies a page segment.

X'FC' Specifies an overlay.

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

- Page definition

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

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

  If you specify MCF2REF=CF, ACIF includes coded fonts. If MCF2REF=CPCS (the default), ACIF processes coded fonts to determine the names of the code pages and font character sets they reference. This is necessary in creating Map Coded Font-2 (MCF-2) structured fields.

request (Byte 18)

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

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

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

A value of X'00' on entry to the exit program specifies that the resource be processed. If you want to ignore the resource, change the **request** byte value to X'01'. Any value other than X'01' will cause ACIF to process the resource.

Non-Zero Return Codes

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

Attributes of the Input Print File

ACIF provides information about the attributes of the input print file in a data structure available to ACIF's user exits. Figure 25 shows the format of this data structure.

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

Figure 25. Sample Print File Attributes C Language Header

Figure 26 shows the format of the data structure in MVS.

```
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 26. Sample Print File Attributes DSECT

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

cc (Bytes 1–3)
The value of the **cc** parameter as specified to the arsacif command. ACIF uses the default value if this parameter is not explicitly specified.

cctype (Byte 4)
The value of the **cctype** parameter as specified to the arsacif command. ACIF uses the default value if this parameter is not explicitly specified.

chars (Bytes 5–24)
The value of the **chars** parameter as specified to the arsacif command, including any commas that separate multiple font specifications. Because the **chars** parameter has no default value, this field contains blanks if no values are specified.

formdef (Bytes 25–32)
The value of the **formdef** parameter as specified to the arsacif command. Because the **formdef** parameter, has no default value, this field contains blanks if no value is specified.

pagedef (Bytes 33–40)
The value of the **pagedef** parameter as specified to the arsacif command. Because the **pagedef** parameter has no default value, this field contains blanks if no value is specified.

prmode (Bytes 41–48)
The value of the **prmode** parameter as specified to the arsacif command. Because the **prmode** parameter has no default value, this field contains blanks if no value is specified.

trc (Bytes 49–51)
The value of the **trc** parameter as specified to the arsacif command. ACIF uses the default value if this parameter is not explicitly specified.

Notes:

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

Chapter 6. ACIF and the IBM AFP Fonts for ASCII Data

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

Table 2 provides a list of the IBM Core Interchange Fonts for use with unformatted ASCII input data. Because these fonts have eight-character names, the table also provides a list of six-character short names. These coded fonts are installed in the /usr/lpp/psf/reslib directory when PSF for AIX is installed. The installation program also creates the symbolic links of the eight-character names that correspond to the six-character names. You may use these short names, *without* the X0 prefix, to satisfy the four-character limitation for specifying font names with the CHARS parameter.

Table 2. Font Mapping Table for Use with the CHARS Parameter

Type Family	Point Size	Coded Font Name	Linked Short Name (for chars Parameter)
Courier	7	X0423072	X04272
Courier	8	X0423082	X04282
Courier	10	X0423002	X04202
Courier	12	X04230B2	X042B2
Courier	14	X04230D2	X042D2
Courier	20	X04230J2	X042J2
Helvetica	6	X0H23062	X0H262
Helvetica	7	X0H23072	X0H272
Helvetica	8	X0H23082	X0H282
Helvetica	9	X0H23092	X0H292
Helvetica	10	X0H23002	X0H202
Helvetica	11	X0H230A2	X0H2A2
Helvetica	12	X0H230B2	X0H2B2
Helvetica	14	X0H230D2	X0H2D2
Helvetica	16	X0H230F2	X0H2F2
Helvetica	18	X0H230H2	X0H2H2
Helvetica	20	X0H230J2	X0H2J2
Helvetica	24	X0H230N2	X0H2N2

Table 2. Font Mapping Table for Use with the CHARS Parameter (continued)

Type Family	Point Size	Coded Font Name	Linked Short Name (for chars Parameter)
Helvetica	30	X0H230T2	X0H2T2
Helvetica	36	X0H230Z2	X0H2Z2
Times New Roman	6	X0N23062	X0N262
Times New Roman	7	X0N23072	X0N272
Times New Roman	8	X0N23082	X0N282
Times New Roman	9	X0N23092	X0N292
Times New Roman	10	X0N23002	X0N202
Times New Roman	11	X0N230A2	X0N2A2
Times New Roman	12	X0N230B2	X0N2B2
Times New Roman	14	X0N230D2	X0N2D2
Times New Roman	16	X0N230F2	X0N2F2
Times New Roman	18	X0N230H2	X0N2H2
Times New Roman	20	X0N230J2	X0N2J2
Times New Roman	24	X0N230N2	X0N2N2
Times New Roman	30	X0N230T2	X0N2T2
Times New Roman	36	X0N230Z2	X0N2Z2

Chapter 7. ACIF Helpful Hints

This chapter contains General-use Programming Interface and Associated Guidance Information.

When using ACIF, the following topics may prove to be helpful to you:

- Working with control statements that contain numbered lines
- Understanding how ACIF processes unbounded box fonts (3800)
- Placing TLEs in named groups
- Working with file transfer
- Understanding how ANSI and machine carriage controls are used
- Understanding common methods of transferring files from other systems to OnDemand servers:
  - Physical media such as tape
  - PC file transfer program
  - FTP
  - MVS Download and Download for OS/390
- Invoke Medium Map (IMM) structured field
- Indexing considerations
- Concatenating the resource file and the document.

Working with control statements that contain numbered lines

Note: This topic contains information relevant to running ACIF on MVS or OS/390 systems.

You sometimes can receive unexpected results when data set names are continued and the control statements have line numbers in columns 73-80, because ACIF reads all 80 columns of the control statements for processing purposes. (ACIF attempts to use the line number as a data set name, and issues MSGAPK451S and MSGAPK417I with a numeric value.) To resolve this problem, remove any line numbers from the control statements and rerun the job, or use a comment indicator ("/*") before each line number.

Understanding how ACIF processes unbounded box fonts (3800)

Note: This topic contains information relevant to running ACIF on MVS or OS/390 systems.

When ACIF reads a font object (coded font, character set) from a library, it searches for the bounded box version of the font (X0, C0 prefix), regardless of the specifications in the page definition or document MCF record. ACIF does not know what device might be displaying or printing the document. ACIF also does not know whether the correct version of fonts in the library is bounded-box or unbounded-box. The reason ACIF searches the bounded box version is that all printer models except the d/t3800 accept bounded-box fonts. The resulting print file will print correctly, regardless of whether the document contains bounded or unbounded font names, because PSF knows the destination printer type and uses the corresponding fonts.

You should be aware that if only unbounded-box fonts are present in the libraries defined for FONTLIB and USERLIB, ACIF will be unable to locate a bounded-box equivalent, in which case, ACIF issues MSGAPK413 and terminates the job. Therefore, use a bounded-box font library if it is available. If, however, a bounded-box font library is not available, you can allow ACIF to find the correct font and build the output page correctly, but prevent an attempt to access C0 character sets by doing the following:

- Make copies of the unbounded-box CODED fonts
- Rename the copies to begin with X0
- Omit the collection of fonts from the **RESTYPE** parameter. (Do *not* specify **RESTYPE**=ALL or **RESTYPE**=FONT.)

Note: The archiving of unbounded-box fonts is *not* recommended. No current or future MO:DCA receiver (printer or viewer, for example) will support unbounded-box fonts.

Placing TLEs in named groups

To avoid having your job terminated by ACIF, IBM recommends that you place page-level TLEs inside named groups, using one named group per page.

You should be aware that if you specify INDEXOBJ=ALL and the input data set contains composed (AFPDS) pages, page-level TLEs (TLE records after the AEG), and no named groups (BNG/ENG), your job may end with error message 425-410 or 425-408. The reason for this action is that no named groups are present, and the page-level TLE records must be collected in memory until the end of the input document or file. MO:DCA index structures contain the extent (size) of the object being indexed. Indexed objects are delimited by a named group (or end document-EDT). If no named groups are present, ACIF will continue to build the index in memory. If the input file is large enough, there will not be enough memory, and ACIF will terminate. The ACIF memory manager currently limits the number (but not the size) of

memory blocks that can be allocated; therefore, increasing the memory available to the indexing process may not alleviate the problem.

Working with file transfer

ACIF needs to know two things about a file in order to print it:

- How long is each print record
- What kind of carriage control is used

As simple as this sounds, it is the source of most of the difficulty people have printing with ACIF in a workstation environment.

ACIF processes print records. A record is a sequence of contiguous characters, usually representing a printed line or a MO:DCA (AFPDS) structured field.[1] Each record has a defined boundary or length. Some files contain information in each record that describes the record's length; these are called variable-length files. Other files require an external definition of length; these are called fixed-length files.

- **Variable-length files**

  Variable-length files may use a length prefix, which means they contain a prefix that identifies the length of the record in the file. Each record contains a field that gives the length of the record. If the record contains a length, that length must be a prefix for each record and it must be a 16-bit binary number that includes the length of the 2-byte length prefix. Use the **FILEFORMAT**=RECORD control statement to identify files with length prefixes.

  Variable-length files may use a separator or delimiter to indicate the end of a record, instead of using a length prefix. All of the bytes up to, but not including, the delimiter are considered to be part of the record. In UNIX, the delimiter is X'0A' (In NT, the delimiter is X'0A0D'). If the file uses EBCDIC encoding, the newline character is X'25'. Use the **FILEFORMAT**=STREAM control statement to designate files that use newlines to indicate record boundaries.

  ACIF reads the first six bytes and tests for all ASCII characters[2], to determine if a file is encoded in ASCII or EBCDIC. If no non-ASCII characters are found, ACIF assumes the file uses the ASCII newline character, X'0A'. Otherwise, ACIF assumes the file uses the EBCDIC newline character, X'25'. Because an input file can misguide ACIF, either intentionally or by accident, a set of rules has been established to determine how ACIF will interpret how a file will be processed. The following combinations are possible:

1. Structured fields are similar to print commands.
2. Code points from X'00' to X'7F'

Data Type	Newline Character
All EBCDIC	EBCDIC X'25'
All EBCDIC	ASCII X'0A' (Note 1)
All ASCII	EBCDIC X'25' (Note 1)
All ASCII	ASCII X'0A'

Note: These combinations are possible only if a file contains a prefix with a string that indicates a different code set than actually exists. For EBCDIC data with ASCII newlines, use X'0320202020200A'. For ASCII data with EBCDIC newlines, use X'03404040404025'.

- **Fixed-length files**

  Fixed-length files contain records that are all the same length. No other separators or prefixes or self-identifying information exists that indicates the record length. You must know the record length and use the **FILEFORMAT**=RECORD,*nnn* control statement, where *nnn* represents the length of each record.

For variable- and fixed-length files using length prefixes, MO:DCA structured fields are treated as a special case. All such structured fields are self-identifying and contain their own length. They need not contain a length prefix to be correctly interpreted, but will be processed correctly if there is a length prefix.

Understanding how ANSI and machine carriage controls are used

In many environments (including IBM mainframes and most minicomputers), printable data normally contains a carriage control character. The carriage control character acts as a vertical tab command to position the paper at the start of a new page, at a specified line on the page, or to control skipping to the next line. The characters can be one of two types: ANSI carriage control or machine carriage control.

- **ANSI carriage control characters**

  The most universal carriage control is ANSI, which consists of a single character that is a prefix for the print line. The standard ANSI characters are:

ANSI	Command
space	Single space the line and print
0	Double space the line and print
-	Triple space the line and print
+	Don't space the line and print
1	Skip to channel 1 (the top of the form, by convention)
2-9	Skip to hardware-defined position on the page
A,B,C	Defined by a vertical tab record or FCB

All ANSI controls perform the required spacing before the line is printed. ANSI controls may 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 may be present in a print file or not, but every record in the file must contain a carriage control if the controls are to be used. If the file contains carriage controls, but **CC**=NO is specified to ACIF, the carriage controls will be treated as printing characters. If no carriage controls are specified, the file will be printed as though it were single spaced.

Understanding common methods of transferring files to OnDemand servers

You can transfer files from other systems to OnDemand servers 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 OnDemand servers and produce output that ACIF can use are:

- Physical media (such as tape)

- PC file transfer program
- FTP
- MVS Download and Download for OS/390

Note: MVS Download and Download for OS/390 are commonly referred to as Download throughout this book.

Physical media

Normally, you can copy fixed-length files without any transformation using a physical media, such as tape.

PC file transfer program

You may transfer files from other systems to OnDemand servers by using an implementation of the most common PC file transfer program (IND$FILE). You may also transfer files from a host to a personal computer. The variety of possible parameters that can affect printing are host-dependent. IBM recommends the following:

- For MVS, OS/390, and VM/CMS files, the default is binary.
- For CICS and VSE, binary is recommended.
- For files with fixed-length records, binary is recommended (you must know the record length).
- For files with variable-length records that contain only printable characters and either ANSI carriage control characters, or no carriage control characters:
  - Use ASCII and CRLF
  - Specify the control statement **INPEXIT**=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 newlines and carriage returns. You must, however, “trick” ACIF by using a prefix of X'0320202020200A'.

FTP

From most systems, FTP works similarly to PC file transfer, and most of the same options are provided. Also, when executing FTP on an OnDemand server, 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

You can use Download to transmit a print dataset from the JES Spool to file systems on OnDemand servers. The MVS (or OS/390) component of Download operates as one or more JES writers. You configure the writers to interpret JCL parameters, such as CLASS and DEST, and route spool files to an OnDemand server. You can use other JCL parameters, such as FORMS and DATASET to determine the application group and application to load. Download transmits data in binary format.

To conserve space and increase transmission speed, Download truncates a record if it contains one or more blank characters (X'40') at the end of the record. As a result, after transmitting a report to the server, some records may contain fewer characters than the assumed record length. If the location of a FIELD begins outside the actual length of a record, ACIF fails unless you specify a DEFAULT value. For example, a report on the OS/390 system contains fixed length records, each 133 bytes in length. Columns 129 through 133 of the records contain audit data generated by the application program. You define an audit field, to extract the values of columns 129 through 133 and store them in the database. If a record has not been audited, the columns contain blank characters. During transmission of the file, Download eliminates the blank characters from the end of all records that contain X'40' in columns 129 through 133. To prevent ACIF from failing, you must define a DEFAULT value for the field. For example:

```
FIELD2=1,129,4,(DEFAULT=X'D5D6D5C5')
```

In the example, if a record is not 129 bytes in length, ACIF generates the value NONE (X'D5D6D5C5') for FIELD2.

Other Considerations for Transferring Files

Conventional file transfer programs cannot correctly handle the combination of variable-length files, which contain bytes that cannot be translated from their original representation to ASCII, and may also contain machine control characters, mixed line data and structured fields, or special code points that have no standard 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 may have to use special parameters.)

---

3. When ASCII is specified, for example, the file transfer program may destroy the data in translation. When binary is specified, the file transfer program may not be able to indicate record lengths.

Note: Some NFS systems do not supply the binary record length for fixed-length files.

ACIF treats a file that contains only structured fields (MO:DCA or AFPDS or LIST3820) as a special case. You can always transfer such a file as binary with no special record separator, and ACIF can always read it because structured fields are self-defining, containing their own length; ACIF handles print files and print resources (form definitions, fonts, page segments, overlays, and so on) in the same way.

Invoke Medium Map (IMM) Structured Field

Retrieval programs must be able to detect which medium map is active, to ensure that pages are reprinted (or viewed) using the correct medium map. To ensure that the correct medium map is used, use the Active Medium Map triplet and the Medium Map Page Number triplet (from the appropriate Index Element [IEL] structured field in the index object file), which designate the name of the last explicitly invoked IMM structured field and the number of pages produced since the IMM was invoked. The retrieval system can use this information to dynamically create IMM structured fields at the appropriate locations when it retrieves a group of pages from the archived document file.

Indexing Considerations

The index object file contains Index Element (IEL) structured fields that identify the location of the tagged groups in the print file. The tags are contained in the Tagged Logical Element (TLE) structured fields.

The structured field offset and byte offset values are accurate at the time ACIF creates the output document file. However, if you extract various pages or page groups for viewing or printing, you will have to dynamically create from the original a temporary index object file that contains the correct offset information for the new file. For example, assume the following:

- ACIF processed all the bank statements for 6 branches, using the account number, statement date, and branch number.
- The resultant output files were archived using a system that allowed these statements to be retrieved based on any combination of these three indexing values.

If you wanted to view all the bank statements from branch 1, your retrieval system would have to extract all the statements from the print file ACIF created (possibly using the IELs and TLEs in the index object file) and create

another document for viewing. This new document would need its own index object file containing the correct offset information. The retrieval system would have to be able to do this.

Under some circumstances, the indexing that ACIF produces may not be what you expect, for example:

- If your page definition produces multiple-up output, and if the data values you are using for your indexing attributes appear on more than one of the multiple-up subpages, ACIF may produce two indexing tags for the same physical page of output. In this situation, only the first index attribute name will appear as a group name, when you are using OnDemand. 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 a skip-to-channel character to start a new page (typically X'89' or X'8B') as a TRIGGER, the indexing tag created will point to the page on which the carriage control character was found, not to the new page started 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.
- If your input file contains application-generated separator pages (for example, banner pages), and you want to use data values for your indexing attributes, you can write an Input Data exit program to remove the separator pages. Otherwise, the presence of those pages in the file will make the input data too unpredictable for ACIF to reliably locate the data values. As alternatives to writing an exit program, you can also change your application program to remove the separator pages from its output, or you can use the INDEXSTARTBY parameter to instruct ACIF to start indexing on the first page after the header pages.
- If you want to use data values for your indexing attributes, but none of the values appear on the first page of each logical document, you can cause ACIF to place an indexing tag on the first page by defining a FIELD parameter with a large enough negative relative record number from the anchor record to “page” backward to the first page. Without referencing this FIELD parameter in an INDEX parameter, the tag generated by any INDEX parameter will be placed on the first page.

Concatenating the Resource Group to the Document

You can create a print file containing all the required print resources by concatenating the output document file to the end of the resource file. Do, however, remember two things when doing this:

- First, although OnDemand 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 OnDemand, because it removes any inline objects before using the offset information.

Specifying the IMAGEOUT Parameter

ACIF converts IM1 format images in the input file, in overlays, and in page segments to uncompressed IOCA format, if **IMAGEOUT=IOCA** (the default) is specified. An uncompressed IOCA image may use a significantly higher number of bytes than an IM1 image and may take more processing time to convert, especially for shaded or patterned areas. Although IOCA is the MO:DCA-P standard for image data, and some data stream receivers may require it, all products may not accept IOCA data. All software products from the IBM Printing Systems Company do, however, accept IOCA data as well as IM1 image data.

IBM recommends that you specify **IMAGEOUT=ASIS**, unless you have a specific requirement for IOCA images.

Chapter 8. ACIF Data Stream Information

General-use Programming Interface and Associated Guidance Information is contained in this chapter.

This chapter describes the Tag Logical Element (TLE) structured field and the formats of the resource data sets.

Tag Logical Element (TLE) Structured Field

TLE structured fields are allowed only in AFP data stream (MO:DCA-P) documents. AFP Application Programming Interface (AFP API) supports the TLE structured field and can be used from host COBOL and PL/I applications to create indexed AFP data stream (MO:DCA-P) documents. Document Composition Facility (DCF), with APAR PN36437, can also be used to insert TLE structured fields in an output document.

The format of the TLE structured field that ACIF supports and generates is as follows:

Carriage Control Character (X'5A')
Specifies the carriage control character, which is required in the first position of the input record to denote a structured field.

Structured Field Introducer (8 bytes)
Specifies the standard structured-field header containing the structured field identifier and the length of the entire structured field, including all of the data.

Tag Identifier Triplet (4–254 bytes)
Specifies the application-defined identifier or attribute name associated with the tag value. An example is 'Customer Name'. This is a Fully Qualified Name triplet (X'02') with a type value of X'0B' (Attribute Name). For more information, refer to *Mixed Object Content Architecture Reference.*

Tag Value Triplet (4–254 bytes)
Specifies the actual value of the index attribute. If the attribute is 'Customer Name', the actual tag value might be 'Bob Smith'. This triplet contains a length in byte 1, a type value of X'36' (Attribute Value) in byte 2, and 2 reserved bytes (X'0000').

The following is an example of a 39-byte TLE structured field containing two index values. For the purposes of illustration, each field within the structured field is listed on a separate line. **X'** ' denotes hexadecimal data, and ″ ″ denotes EBCDIC or character data.

```
X'5A0026D3A090000000'
X'11020B00'
"Customer Name"
X'0D360000'
"Bob Smith"
```

TLE structured fields can be associated with a group of pages or with individual pages. Consider a bank statement application. Each bank statement is a group of pages, and you may want to associate specific indexing information at the statement level (for example, account number, date, customer name, and so on). You may also want to index (tag) a specific page within the statement, such as the summary page. The following is an example of a print file that contains TLEs at the group level as well as at the page level:

```
BDT
  BNG
    TLE Account #, 101030
    TLE Customer Name, Mike Smith
      BPG
        Page 1 data
      EPG
      BPG
        Page 2 data
      EPG
      ...
      ...
      BPG
        TLE Summary Page, n
        Page n data
      EPG
  ENG
  ...
EDT
```

ACIF can accept input files that contain both group-level and page-level indexing tags. You can also use the input record exit of ACIF to insert TLE structured fields into an AFP data stream (MO:DCA-P) file, where applicable. The indexing information in the TLE structured field applies to the page or group containing them. In the case of groups, the TLE structured field can appear anywhere between a Begin Named Group (BNG) structured field and the first page (BPG structured field) in the group. In the case of composed-text pages, the TLE structured field can appear anywhere following the Active Environment Group, between the End Active Environment (EAG) and End Page (EPG) structured fields. Although ACIF does not limit the number of

TLE structured fields that can be placed in a group or page, you should consider the performance and storage ramifications of the number included.

ACIF does not require the print file to be indexed in a uniform manner; that is, every page containing TLE structured fields does not have to have the same number of tags as other pages or the same type of index attributes or tag values. This allows a great deal of flexibility for the application. When ACIF completes processing a print file that contains TLE structured fields, the resultant indexing information file may contain records of variable length.

Format of the Resources File

ACIF retrieves referenced AFP resources from specified libraries and creates a single file that contains these resources. Using ACIF, you can control the number of resources as well as the type of resources in the file by using a combination of **RESTYPE** values and processing in the resource exit.

ACIF can retrieve all the resources used by the print file and can place them in a separate resource file. The resource file contains a resource group structure whose syntax is as follows:

```
BRG
  BR
    AFP Resource 1
  ER
  BR
    AFP Resource 2
  ER
  ..
  BR
    AFP Resource n
  ER
ERG
```

ACIF does not limit the number of resources that can be included in this object, but available storage is certainly a limiting factor.

Begin Resource Group (BRG) Structured Field

ACIF assigns a null token name (X'FFFF') to this structured field and also creates three additional triplets: an FQN type X'01' triplet, an Object Date and Time Stamp triplet, and an FQN type X'83' triplet. The FQN type X'01' triplet contains the data set name identified in the DDname statement for **RESOBJDD**. The Object Date and Time Stamp triplet contains date and time information from the operating system on which ACIF runs. The date and time values reflect when ACIF was invoked to process the print file. The FQN

type X'83' triplet contains the AFP output print file name identified by the DDname specified in the **OUTPUTDD** parameter.

Begin Resource (BR) Structured Field

ACIF uses this structured field to delimit the resources in the file. ACIF also identifies the type of resource (for example, overlay) that follows this structured field. The type is represented as a 1-byte hexadecimal value where:

X'03' Specifies a GOCA.

X'05' Specifies a BCOCA.

X'06' Specifies a IOCA.

X'40' Specifies a font character set.

X'41' Specifies a code page.

X'92' Specifies an object container.

X'FB' Specifies a page segment.

X'FC' Specifies an overlay.

X'FE' Specifies a form definition.

End Resource (ER) and End Resource Group (ERG) Structured Fields

ACIF always assigns a null token name (X'FFFF') to the Exx structured fields it creates. The null name forces a match with the corresponding BR and BRG structured fields.

Chapter 9. Format of the ACIF Index Object File

General-use Programming Interface and Associated Guidance Information is contained in this chapter.

One of the optional files ACIF can produce contains indexing, offset, and size information. The purpose of this file is to enable applications such as archival and retrieval applications to selectively determine the location of a page group or page within the AFP data stream print file, based on its index (tag) values.

The following example shows the general internal format of this object:

```
BDI
  IEL GroupName=G1
    TLE (INDEX1)
    ...
    TLE (INDEXn)
      IEL PageName=G1P1
        TLE (INDEX1)
        ...
        TLE (INDEXn)
      ...
      IEL PageName=G1Pn
  ...
  IEL GroupName=Gn
    TLE (INDEX1)
    ...
    TLE (INDEXn)
      IEL PageName=GnP1
        TLE (INDEX1)
        ...
        TLE (INDEXn)
      ...
      IEL PageName=GnPn
EDI
```

The example illustrates an index object file containing both page-level and group-level Index Element (IEL) and Tag Logical Element (TLE) structured fields.

Group-Level Index Element (IEL) Structured Field

If **INDEXOBJ=GROUP** is specified, ACIF creates an index object file with the following format:

```
BDI
  IEL Groupname=G1
    TLE
```

```
    ...
    TLE
  ...
  IEL Groupname=Gn
    TLE
    ...
    TLE
EDI
```

This format is useful to reduce the size of the index object file, but it allows manipulation only at the group level; that is, you cannot obtain the offset and size information for individual pages. You also lose any indexing information (TLEs) for pages; the TLE structured fields for the pages still exist in the output print file, however.

Page-Level Index Element (IEL) Structured Field

If **INDEXOBJ=ALL** is specified, ACIF creates an index object file with the following format:

```
BDI
  IEL Groupname=G1
    TLE
    ...
      IEL Pagename=G1P1
        TLE
        ...
      ...
      IEL Pagename=G1Pn....
  ...
  IEL Groupname=Gn
    TLE
    ...
      IEL Pagename=GnP1
      ...
      IEL Pagename=GnPn
        TLE
        ...
EDI
```

This example contains IEL structured fields for both pages and groups. Notice that TLE structured fields are associated with both pages and groups. In this example, where an application created an indexed AFP print file containing both page-level and group-level TLE structured fields, ACIF can create IEL structured fields for the appropriate TLE structured fields.

An index object file containing both page-level and group-level IEL structured fields can provide added flexibility and capability to applications that operate on the files created by ACIF. This type of index object file provides the best performance when you are viewing a file using OnDemand.

Begin Document Index (BDI) Structured Field

ACIF assigns a null token name (X'FFFF') and an FQN type X'01' triplet to this structured field. The FQN type X'01' value is the file name identified by the DDname specified in the **INDEXDD** parameter. ACIF also creates an FQN type X'83' triplet containing the name of the AFP output print file, identified by the DDname specified in the **OUTPUTDD** parameter.

ACIF also creates a Coded Graphic Character Set Global Identifier triplet X'01' using the code page identifier specified in the **CPGID** parameter. For more information on the **CPGID** parameter, see the arsacif command reference, beginning on 61. ACIF assigns a null value (X'FFFF') to the Graphic Character Set Global Identifier.

Index Element (IEL) Structured Field

The IEL structured field associates indexing tags with a specific page or group of pages in the output document file. It also contains the byte and structured-field offset to the page or page group and the size of the page or page group in both bytes and structured-field count. The following is a list of the triplets that compose this structured field:

- FQN Type X'8D'

  This triplet contains the name of the active medium map associated with the page or page group. In the case of page groups, this is the medium map that is active for the first page in the group, because other medium maps can be referenced after subsequent pages in the group. If no medium map is explicitly invoked with an Invoke Medium Map (IMM) structured field, ACIF uses a null name (8 bytes of X'FF') to identify the default medium map; that is, the first medium map in the form definition.
- Object Byte Extent (X'57')

  This triplet contains the size, in bytes, of the page or group this IEL structured field references. The value begins at 1.
- Object Structured Field Extent (X'59')

  This triplet contains the number of structured fields that compose the page or group referenced by this IEL structured field. In the host environment, each record contains only one structured field, so this value also represents the number of records in the page or group. The value begins at 1.
- Direct Byte Offset (X'2D')

  This triplet contains the offset, in bytes, from the start of the output print file to the particular page or group this IEL structured field references. The value begins at 0.
- Object Structured Field Offset (X'58')

This triplet contains the offset, in number of structured fields, from the start of the output print file to the start of the particular page or group this IEL structured field references. The value begins at 0.

- FQN Type X'87'

  This triplet contains the name of the page with which this IEL structured field is associated. The name is the same as the FQN type X'01' on the BPG structured field. This triplet applies **only** to page-level IEL structured fields.

- FQN Type X'0D'

  This triplet contains the name of the page group with which this IEL structured field is associated. The name is the same as the FQN type X'01' on the BNG structured field. This triplet applies **only** to group-level IEL structured fields.

- Medium Map Page Number (X'56')

  This triplet defines the relative page count since the last Invoke Medium Map (IMM) structured field was processed or from the logical invocation of the default medium map. In the case of page groups, this value applies to the first page in the group. The value begins at 1 and is incremented for each page.

Tag Logical Element (TLE) Structured Field

ACIF creates TLE structured fields as part of its indexing process, or it can receive these structured fields from the input print file. When ACIF creates TLE structured fields, the first TLE structured field is **INDEX1**, the next TLE structured field is **INDEX2**, and so on, to a maximum of 32 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.

Chapter 10. Format of the ACIF Output Document File

This chapter contains General-use Programming Interface and Associated Guidance Information.

Although ACIF can create three separate output files, only one of the files is required. ACIF always creates a print file. In doing so, ACIF may create the following structured fields:

- Tag Logical Element (TLE)
- Begin Named Group (BNG)
- End Named Group (ENG)

The TLE was described in “Chapter 9. Format of the ACIF Index Object File” on page 181; the other two structured fields will be described in this chapter. The examples on the next two pages illustrate the two possible AFP data stream document formats ACIF may produce.

```
BDT
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        Page 1 of group 1
      EPG
      BPG
        Page 2 of group 1
      EPG
      ...
      BPG
        Page n of group 1
      EPG
  ENG
  ...
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        Page 1 of group n
      EPG
      BPG
        Page 2 of group n
      EPG
      ...
      BPG
        Page n of group n
      EPG
  ENG
EDT
```

Figure 27. Example of Code Containing Group-Level Indexing

Figure 27 illustrates the one format ACIF can produce when it converts and indexes a print file, generating indexes at the group level.

```
BDT
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        TLE (INDEX1)
        ...
        TLE (INDEXn)
        Page 1 of group 1
      EPG
      BPG
        Page 2 of group 1
      EPG
      ...
      BPG
        TLE (INDEX1)
        ...
        TLE (INDEXn)
        Page n of group 1
      EPG
  ENG
  ...
  BNG Groupname=(index value + sequence number)
    TLE (INDEX1)
    TLE (INDEX2)
    ...
    TLE (INDEXn)
      BPG
        Page 1 of group n
      EPG
      BPG
        TLE (INDEX1)
        ...
        TLE (INDEXn)
        Page 2 of group n
      EPG
      ...
      BPG
        Page n of group n
      EPG
  ENG
EDT
```

Figure 28. Example of Code Containing Group- and Page-Level Indexing

Figure 28 illustrates an input file that has already been indexed (tagged) and converted to MO:DCA-P format, which the AFP API program can do. This example shows that you can index (tag) both groups and pages from an application.

Page Groups

Page groups are architected groups of one or more pages to which some action or meaning is assigned. Consider the example of the bank statement application. Each bank statement in the print file comprises one or more pages. By grouping each statement in a logical manner, you can assign specific indexing or tag information to each group (statement). You can then use this grouping to perform actions such as archival, retrieval, viewing, preprocessing, postprocessing, and so on. The grouping also represents a natural hierarchy. In the case of OnDemand, 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 server all of the bank statements for a specific branch. You can then select a specific bank statement (group-level) to view and select a tagged summary page (page-level).

Begin Document (BDT) Structured Field

When ACIF processes an AFP data stream print file, it checks for an FQN type X'01' triplet in the BDT structured field. If the FQN triplet exists, ACIF uses it; otherwise, ACIF creates one using the file name identified in the DDname statement for **OUTPUTDD**. ACIF uses the FQN value when it creates an FQN type X'83' triplet on the Begin Document Index (BDI) structured field in the index object file and on the Begin Resource Group (BRG) structured field in the resource file. Although the input file may contain multiple BDT structured fields, the ACIF output will contain only one BDT structured field. (The same is true of End Document (EDT) structured fields.)

In the case of line-mode files, ACIF creates the BDT structured field. ACIF assigns a null token name (X'FFFF') and creates an FQN type X'01' triplet using the file name identified in the DDname statement for **OUTPUTDD**.

ACIF also creates a Coded Graphic Character Set Global Identifier triplet X'01' using the code page identifier specified in the **CPGID** parameter. For more information on the **CPGID** parameter, see the arsacif command reference, beginning on page 61. ACIF assigns a null value (X'FFFF') to the Graphic Character Set Global Identifier.

ACIF also creates two additional FQN triplets for the resource name (type X'0A') and the index object name (type X'98'). These two values are the same as those contained in their respective type X'01' triplets on the BDI and BRG structured fields.

Begin Named Group (BNG) Structured Field

When ACIF processes an AFP data stream print file containing page groups, it checks for an FQN type X'01' triplet on each BNG structured field. If the FQN triplet exists, ACIF uses the value when it creates an FQN type X'0D' triplet on the corresponding Index Element (IEL) structured field in the index object file. ACIF appends an 8-byte rolling sequence number to ensure uniqueness in the name. If no FQN triplet exists, ACIF creates one. Here too, ACIF appends a rolling, 8-byte EBCDIC sequence number to ensure uniquely named groups, up to a maximum of 99 999 999 groups within a print file.

When ACIF indexes a print file, it creates the BNG structured fields. It assigns a rolling 8-byte EBCDIC sequence number to the token name (for example, 00000001 where 1=X'F1'). The sequence number begins with 00000001 and is incremented by 1 each time a group is created. ACIF also creates an FQN type X'01' triplet by concatenating the specified index value (**GROUPNAME**) with the same sequence number used in the token name. If the value of the index specified in **GROUPNAME** is too long, the trailing bytes are replaced by the sequence number. This occurs only if the specified index value exceeds 242 bytes in length. A maximum of 99 999 999 groups can be supported before the counter wraps. This means that ACIF can guarantee a maximum of 99 999 999 unique group names.

Tag Logical Element (TLE) Structured Field

As was mentioned in “Chapter 9. Format of the ACIF Index Object File” on page 181, 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 32 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 99 999 999 pages within a print file. ACIF creates IEL structured fields for pages only if **INDEXOBJ=ALL** is specified.

When ACIF processes a line-mode print file, it creates the BPG structured fields. It assigns a rolling 8-byte EBCDIC sequence number to the token name (for example, 00000001, where 1=X'F1'). The sequence number begins with 00000001 and is incremented by 1 each time a group is created. ACIF also creates an FQN type X'01' triplet using the same sequence number value, and uses this value in the appropriate IEL structured field if **INDEXOBJ=ALL** is specified. A maximum of 99 999 999 groups can be supported before the counter wraps. This means that ACIF can guarantee a maximum of 99 999 999 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

When ACIF produces an output file in the MO:DCA-P format, each structured field in the file is a single record preceded by a X'5A' carriage control character. The following sections describe the required changes ACIF must make to support MO:DCA-P output format.

Composed Text Control (CTC) Structured Field

Because this structured field has been declared obsolete, ACIF ignores it and does not pass it to the output file.

Map Coded Font (MCF) Format 1 Structured Field

ACIF converts this structured field to an MCF Format 2 structured field. 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 may affect document integrity in the case of archival, because no explicit resource names are referenced for ACIF to retrieve.

Presentation Text Data Descriptor (PTD) Format 1 Structured Field

ACIF converts this structured field to a PTD Format 2 structured field.

Inline Resources

MO:DCA-P does not support inline resources at the beginning of a print file (before the BDT structured field); therefore, inline resources must be removed. The resources will be saved and used as requested.

Page Definitions

Because page definitions are used only to compose line-mode data into pages, this resource is not included in the resource file. The page definition is not included because it is no longer needed to view or print the document file.

Chapter 11. Using ACIF in MVS

This chapter provides information about using ACIF in the MVS environment.

Note: ACIF is also provided with PSF for OS/390 for the OS/390 environment. Statements in this chapter about ACIF in MVS and PSF/MVS apply equally to ACIF in the OS/390 environment and PSF for OS/390. PSF/MVS and PSF for OS/390 are commonly referred to as PSF throughout this book.

Sample JCL

Figure 29 shows sample JCL to invoke ACIF to process print output from an application.

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

Figure 29. Sample MVS JCL to Invoke ACIF

Explaining the MVS JCL Statements

The JCL statements in Figure 29 are explained as follows. For more information about programming JCL, refer to the PSF *Application Programming Guide*.

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

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

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

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

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

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

INPUT
This DD statement defines the print data set to be processed by ACIF. In the sample in Figure 29 on page 193, this is the same data set as defined in the **PRINTOUT** DD statement.

OUTPUT
This DD statement defines the name of the print data set that ACIF creates as a result of processing the application's print data set. See Figure 29 on page 193 for the DCB requirements.

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

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

This parameter is not required unless indexing is requested or unless the input print data set contains indexing structured fields. If you are not sure

whether the print data set contains indexing structured fields, and you do not want an index object file created, specify DD DUMMY; no index object file will be created.

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

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

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

ACIF Parameters

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

For MVS, you may need to consult your system programmer for information on resource library names and other printing defaults contained in the PSF startup procedures used in your installation.

Syntax Rules for MVS Parameters

The following are general syntax rules for parameter files:

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

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

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

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

Is equivalent to specifying:

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

- When ACIF processes any unrecognized or unsupported parameter, it issues a message, ignores the parameter, and continues processing any remaining parameters until the end of the file, at which time it terminates processing.
- If the same parameter is specified more than one time, ACIF uses the last value specified. For example, if the following is specified:

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

  ACIF uses code page 395.
- Comments must be specified using "/*" as the beginning delimiter. For example:

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

  Comments can appear anywhere, but ACIF ignores all information in the record following the "/*" character string.
- Although ACIF supports parameter values spanning multiple records, it does not support multiple parameters in a single record. The following is an example of this:

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

JCL and ACIF Processing Parameters

Figure 30 on page 197 shows an example of MVS JCL and ACIF processing parameters used to invoke the ACIF program to index an input file.

```
//job...   JOB ...
//APKSMAIN 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 *
                    /* 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                    */
FIELD5 = 4,60,12                    /* Date Due                */
INDEX1 = 'Account Number',field1    /* 1st INDEX attribute     */
INDEX2 = 'Name',field2              /* 2nd 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
OVLYLIB = SYS1.OVERLIB
PDEFLIB = SYS1.PDEFLIB
PSEGLIB = SYS1.PSEGLIB
RESFILE = SEQ                       /* resource file type      */
RESTYPE = FDEF,PSEG,OVLY            /* 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=(12288,(150,15),RLSE),UNIT=SYSDA,
//         DCB=(LRECL=12284,BLKSIZE=12288,RECFM=VBM,DSORG=PS)
//SYSPRINT DD DSN=APKACIF.SYSPRINT,DISP=(NEW,CATLG),
//         SPACE=(9044,(5,5),RLSE),UNIT=SYSDA,
//         DCB=(BLKSIZE=9044,RECFM=VBA,DSORG=PS)
```

Figure 30. Example of an MVS ACIF Application

MVS Libraries

The example ACIF parameters defined the following libraries:

Library Name	MVS Name
FDEFLIB Form definition library	SYS1.FDEFLIB
FONTLIB Font libraries	SYS1.FONTLIBB SYS1.FONTLIBB.EXTRA
OVLYLIB Overlay library	SYS1.OVERLIB
PDEFLIB Page definition library	SYS1.PDEFLIB
PSEGLIB Page segment library	SYS1.PSEGLIB

ACIF Output

The example ACIF job created the following output files:

Type of File	MVS Name
Document file, including indexing structured fields	APKACIF.OUTPUT
Index object file	APKACIF.INDEX
Resource file	APKACIF.RESLIB
Message file listing: • ACIF parameters used • Resources used • Return code	APKACIF.SYSPRINT

Concatenating Files

Before you can process the MVS files created by ACIF with the OnDemand data loading programs, you must concatenate the index object file and the resource file to the document file, creating a single input file for OnDemand. You can then transfer the concatenated file to an OnDemand server and process it with the OnDemand data loading programs.

Figure 31 on page 199 shows MVS JCL you can use to concatenate the files created by ACIF:

```
//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)
```

Figure 31. Example of an MVS JCL used to Concatenate ACIF Files

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**.

Part 2. OnDemand Generic Indexer Reference

This part of the book provides information about the OnDemand generic indexer.

Chapter 12. Introducing the generic indexer

Overview

The OnDemand generic indexer allows you to index input files that you cannot or do not want to index with ACIF. The generic indexer obtains index data and input file information from a generic index file. Using the information in the generic index file, OnDemand creates one index row for each input file (or portion of a file) identified in the generic index file. The index row contains the values that users can use to construct queries and retrieve documents. For example, suppose that you want to archive a library of publications about product offerings. The publications are in a format that can be stored in and viewed with OnDemand. However, you cannot index the publications with ACIF, because the source data is not AFP or line data. To allow users to search for, locate, and view the publications using OnDemand, you must process them with the generic indexer.

You typically use the arsload command to process the generic index file and load the input files referenced in the generic index file.

Format of the generic indexer file

The generic index file contains indexing, offset, and file records grouped into two types of stanzas:

- A field names stanza. The FIELD NAMES BEGIN: and FIELD NAMES END: records delimit the field names stanza. Each record in the field names stanza identifies one database field name. Create one field name record for each application group database field.
- One or more index row stanzas. Create one index row stanza for each input file (or portion of an input file) you want to index. Each index record in an index row stanza identifies one index value. You must supply an index value for each database field name identified in the field names stanza. An index row stanza also contains records that identify the name of the input file, the offset value, and the length value.

Figure 32 on page 204 shows the general format of records in the generic index file.

```
FIELD NAMES BEGIN:
index_1_name
⋮

index_32_name
FIELD NAMES END:
index_1_value
⋮

index_32_value
input_filename
offset
length
index_1_value
⋮

index_32_value
input_filename
offset
length
⋮
```

Figure 32. Format of the Generic Index File

Note: In the figure, the vertical ellipsis character means that we left out one or more records.

The parameters, values, and options are:

FIELD NAMES BEGIN:
The beginning of the field names stanza. A required string that identifies where OnDemand can locate the names of the index fields. This string must be entered in uppercase letters.

index_n_name
The name of an index. Specify one record for each application group field. OnDemand supports up to 32 indexes per application group. The name of an index should be the same as its corresponding application group database field name. Otherwise, you must map the index names to the database field names on the application Load Information tab. You must provide an index for each application group database field.

FIELD NAMES END:
The end of the field names stanza. A required string that signals the end of the index name records. This string must be entered in uppercase letters.

index_n_value
The index value for an associated index record. Add one index value

record for each index name record in the field names stanza. You must provide an index value for each application group database field.

input_filename
The file name or full path name of the input file. On UNIX servers, file and path names are case sensitive. If you do not specify a path, OnDemand searches the current directory. If the **input_filename** record is blank, the previous **input_filename** record is used to process the input data. If the first **input_filename** record in the generic index file is blank, you must specify the name of the input file when you run the arsload command.

offset The number of bytes into the source file where OnDemand locates the beginning of the data to index. Specify 0 (zero) for the beginning of the file.

length The number of bytes of data that OnDemand processes. Specify 0 (zero) to process the entire file.

AFP data and the generic indexer

You can create generic index data for files that contain AFP resources and documents and process them with the generic indexer. However, when you create the generic index file:

- The offset of the first AFP document should always be 0 (zero), even though the actual offset of the first AFP document in the file is not zero when AFP resources are contained in the file.
- Subsequent offsets should be calculated using the length and offset of the previous AFP document in the file. The OnDemand loading program determines where the AFP resources end in the file and processes the AFP documents using the offsets and lengths provided, relative to where the resources end.

Chapter 13. Using the generic indexer

Processing TIF files

Figure 33 shows a portion of a generic index file used to create index information for image files that will be stored in OnDemand. Each file represents a letter that was scanned and saved as a TIF file.

```
FIELD NAMES BEGIN:
scandate
letterdate
name
company
subject
reference
FIELD NAMES END:
08/01/95
01/12/95
G. S. Dalderman
B and P Electric
Repairs and Maintenance
Invoice Number 900-10V17
IMAGE001.tiff
0
0
:

08/01/95
06/27/95
Norma Pinestuff
Rogers Accounting
1995 Tax Audit
Account No. 12345-G
IMAGE999.tiff
0
0
```

Figure 33. Generic Index TIF Files

In the example, OnDemand creates one row in the database for each index record. Each row in the database contains six user-defined index fields: `scandate`, `letterdate`, `name`, `company`, `subject`, and `reference`. Each index record identifies one input file. Users can open a folder that references the application group where OnDemand stored the index data and files and search for letters using the letter date and any combination of scan date, name, company, subject, and reference.

Figure 34 shows how to define a generic index file to create index data for a single file that contains several TIFF images. We must identify the starting point and length of each image in the file, so that the OnDemand loading program can store them as individual, retrievable entities.

```
FIELD NAMES BEGIN:
tdate
tnumber
tname
FIELD NAMES END:
07/27/95
00000001
tif1
arsd0mst.tif
0
50000
07/27/95
00000002
tif2

50000
50000
07/27/95
00000003
tif3

100000
50000
07/27/95
00000004
tif4

150000
50000
```

Figure 34. Generic Index TIF Documents

In the example, OnDemand creates four database rows: one for each set of index values (each TIFF image contained in the input file). OnDemand uses the offset and length values to locate and process the images. Each row contains three user-defined index fields: `tdate`, `tnumber`, and `tname`. We identify the input file in the first index row stanza. The remaining index row stanzas contain a blank input filename record, which causes OnDemand to process the previously named input file.

Processing AFP documents

Figure 35 shows how to define a generic index file to create index data for a single AFP file that contains multiple documents. Even though the input file contains resources concatenated to the beginning of the file, the offset of the first AFP document in the file is zero. The OnDemand loading program processes the documents using the offsets provided, relative to where the resources end in the input file.

```
FIELD NAMES BEGIN:
ddate
dnumber
dname
FIELD NAMES END:
07/27/95
0000-0001-0001
statement1
arsr0mst.afp
0
24000
07/27/95
0000-0001-0002
statement2

24000
24000
07/27/95
0000-0001-0003
statement3

48000
24000
:

07/27/95
0000-0001-0100
statement100

2376000
24000
```

Figure 35. Generic Index AFP Documents

In the example, OnDemand creates one database row for each set of index values (each AFP document contained in the file). OnDemand uses the offset and length values to locate and process the documents. Each row contains three user-defined index fields: `ddate`, `dnumber`, and `dname`. We identify the input file in the first index row stanza. The remaining index row stanzas contain a blank input filename record, which causes OnDemand to process the previously named input file.

Part 3. OnDemand PDF Indexer Reference

This part of the book provides information about the OnDemand PDF indexer.

Chapter 14. Overview

What is the PDF indexer?

The OnDemand PDF Indexer is a utility you can use to extract index data from or generate index data about PDF input files. The index data can enhance your ability to store, retrieve, and view documents with OnDemand. The PDF Indexer supports PDF Version 1.2 input and output data streams. For more information about the PDF data stream, refer to the *Portable Document Format Reference Manual*, published by Adobe Systems Incorporated. Adobe also provides online information with the Acrobat Exchange and Acrobat Distiller products, including online guides for Capture, PDFWriter, Distiller, and Exchange.

You process and store PDF documents on the server using standard OnDemand functions. To process a document, you must define an OnDemand application and application group. As part of the application, you must define the indexing parameters used by the PDF Indexer to generate the index data. You can automate the indexing and loading of data by configuring and running the OnDemand data loading program as a daemon (UNIX servers) or service (Windows NT servers).

After you use the PDF Indexer and the data loading program to index and store input files in OnDemand, you can do the following:

- Use one of the OnDemand client programs to view the PDF document or documents created during the indexing and loading process. You can also print pages of the PDF document you are viewing from the OnDemand client program.
- Use the OnDemand server print facility to print the PDF document. This capability requires PSF for AIX.

Figure 36 on page 214 shows an overview of the data indexing and loading process.

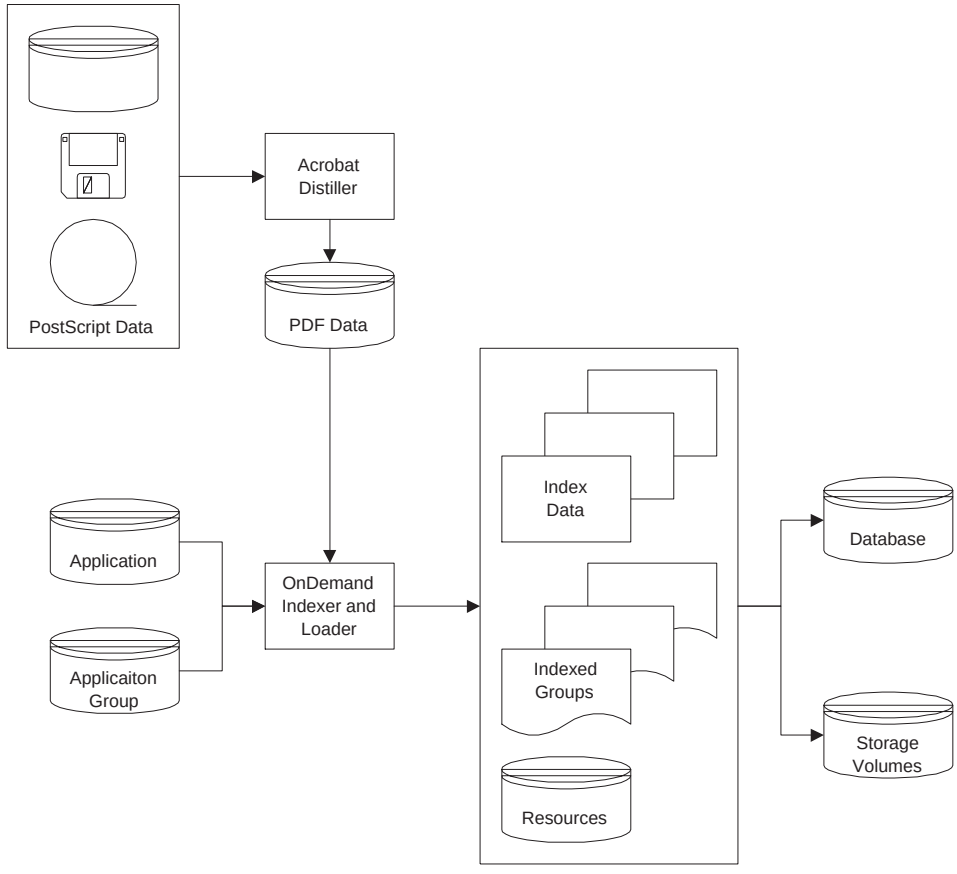


Figure 36. Indexing and Loading PDF Data in OnDemand

The PDF Indexer processes PDF input files. A PDF file is a distilled version of a PostScript file, adding structure and efficiency. A PDF file can be created by Acrobat Distiller or a special printer driver program called a PDFWriter. You can automate the distilling process by configuring and running the Distiller daemon (UNIX servers) or Acrobat Distiller (Windows NT servers). Refer to the online documentation provided with Acrobat Distiller for more information about preparing input data for the Distiller.

The OnDemand data loading and indexing programs obtain processing information from application and application group definitions stored in the database. The application definition identifies the type of input data, the indexing program used to generate index data, the indexing parameters, and other information about the input data. The application group identifies the database and storage management characteristics of the data. You can use the administrative client to create the application and the indexing parameters.

When processing PDF data and the Indexing Information page identifies PDF as the indexer, OnDemand automatically calls the PDF Indexer. The PDF Indexer processes the PDF input file using indexing parameters that determine the location and attributes of the index data. The PDF Indexer extracts index data from the PDF file and generates an index file and an output file. The output file contains groups of indexed pages. A group of indexed pages can represent the entire input file or, more typically, segments of the input file. If the input file contains logical groups of pages, such as statements or policies, the PDF Indexer can create an indexed group for each statement or policy in the input file so that users can retrieve a specific statement or set of statements, rather than the entire file. After indexing the data, OnDemand stores the index data in the database and the indexed groups on storage volumes. You can automate the data indexing and loading process by configuring and running the OnDemand data loading program to run as a daemon (UNIX servers) or service (Windows NT servers).

How OnDemand uses index information

Every item stored in OnDemand is indexed with one or more group-level indexes. Groups are determined when the value of an index changes (for example, account number). When you store a PDF file in OnDemand, the data loading program invokes the PDF indexing program to process the indexing information and create index data. The data loading program then loads the index data in the database, storing the group-level attribute values that the PDF indexing program extracted from the data into their corresponding database fields. Figure 37 shows an overview of the index creation and loading process.

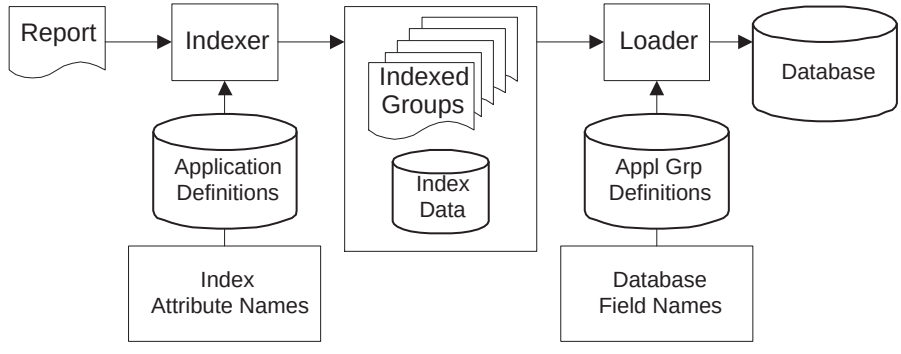


Figure 37. Index Creation and Loading

You create an application for each report that you want to archive in OnDemand. When you create an application, you define the indexing parameters that the indexing program uses to process the report and create

the index data that is loaded into the database. For example, an INDEX parameter includes an attribute name and identifies the FIELD parameter that the indexing program uses to locate the attribute value in the input data. When you create an application, you must assign the application to an application group. The attribute name you specify on an INDEX parameter should be the same as the name of the application group database field where you want OnDemand to store the index values.

You define database fields when you create an application group. OnDemand creates a column in the application group table for each database field that you define. When you index a report, you create index data that contains index field names and index values extracted from the report. OnDemand stores the index data into the database fields.

To search for reports stored in OnDemand, the user opens a folder. The search fields that appear when the user opens the folder are mapped to database fields in an application group (which, in turn, represent index attribute names). The end-user constructs a query by entering values in one or more search fields. OnDemand searches the database for items that contain the values (index attribute values) that match the search values entered by the end-user. Each item contains group-level index information. OnDemand lists the items that match the query. When the user selects an item for viewing, the OnDemand client program retrieves the selected item from cache storage or archive media.

Indexing input data

Indexing concepts

Indexing parameters include information that allow the PDF Indexer to identify key items in the print data stream, *tag* these items, and create *index elements* pointing to the tagged items. OnDemand uses the tag and index data for efficient, structured search and retrieval. You specify the index information that allows the PDF Indexer to segment the data stream into individual items, called *groups*. A group is a collection of one or more pages, such as a bank statement, insurance policy, phone bill, or other logical segment of a report. The PDF Indexer creates indexes for each group when the value of an index changes (for example, account number).

A tag is made up of an *attribute name*, for example, Customer Name, and an *attribute value*, for example, Earl Hawkins. Tags also include information that tell the PDF Indexer where to locate the attribute value on a page. For example, a tag used to collect customer name index values provides the PDF Indexer with the starting and ending position on the page where the customer name index values appear. The PDF Indexer generates index data and stores it

in a generic index file. See “Part 2. OnDemand Generic Indexer Reference” on page 201 for more information about the generic index file.

Coordinate system

The location of the text strings the PDF Indexer uses to determine the beginning of a group and index values are described as *x* and *y* pairs in a coordinate system imposed on the page. For each text string, you identify its upper left and lower right position on the page. The upper left corner and lower right corner form a string box. The string box is the smallest rectangle that completely encloses the text string. The origin is in the upper left hand corner of the page. The *x* coordinate increases to the right and *y* increases down the page. You also identify the page on which the text string appears. For example, the text string Customer Name, that starts 4 inches to the right and 1 inch down and ends 5.5 inches to the right and 1.5 inches down on the first page in the input file can be located as follows:

```
ul(4,1),lr(5.5,1.5),1,'Customer Name'
```

OnDemand provides the arspdump command to help you identify the locations of text strings on the page.

Indexing parameters

Processing parameters can contain index and conversion parameters, options, and values. For most reports, the PDF Indexer requires at least three indexing parameters to generate index data:

- TRIGGER

  The PDF Indexer uses triggers to determine where to locate data. A trigger instructs the PDF Indexer to look for certain information in a specific location on a page. When the PDF Indexer finds the text string in the input file that contains the information specified in the trigger, it can begin to look for index information.

  – The PDF Indexer compares words in the input file with the text string specified in a trigger.

  – The location of the trigger string value must be identified using the *x,y* coordinate system and page offsets.

  – A maximum of 16 triggers can be specified.

  – All triggers must match before the PDF Indexer can begin to locate index information.

- FIELD

  The field parameter specifies the location of the data that the PDF Indexer uses to create index values.

  – Field definitions are based on TRIGGER1 by default, but can be based on any of 16 TRIGGER parameters.

- The location of the field must be identified using the *x*,*y* coordinate system and page offsets.
- A maximum of 32 fields can be defined.
- A field parameter can also specify all or part of the actual index value stored in the database.

• INDEX

The index parameter is where you specify the attribute name and identify the field or fields on which the index is based. We strongly encourage you to name the attribute the same as the application group database field name.

- The PDF Indexer creates indexes for a group of one or more pages.
- You can concatenate field parameters to form an index.
- A maximum of 32 index parameters can be specified.

The PDF Indexer creates a new group and extracts new index values when one or more of the index values change.

Figure 38 on page 219 depicts a portion of a page from a sample input file. We've enclosed the text strings that determine the beginning of a group and the index values in rectangles.

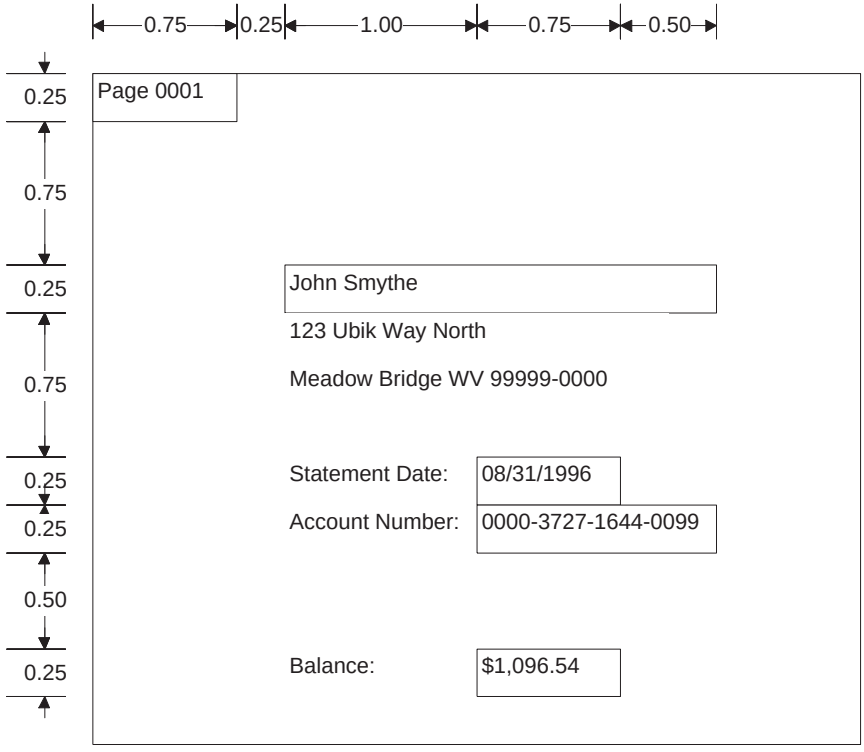


Figure 38. Indexing a Report

TRIGGER parameters tell the PDF Indexer how to identify the beginning of a group in the input. The PDF Indexer requires one TRIGGER parameter to identify the beginning of a group (statement) in the sample file. FIELD parameters determine the location of index values in a statement. Fields are based on the location of trigger records. INDEX parameters identify the attribute names of the index fields. Indexes are based on one or more field parameters. The following parameters could be used to index the report depicted in Figure 38. See "Chapter 16. Parameter reference" on page 225 for details about the parameter syntax.

- Define a trigger to search each page in the input data for the text string that identifies the start of a group (statement):

  ```
  TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 0001'
  ```

- Define fields to identify the location of index data. For the sample report, we might define four fields:
  - FIELD1 identifies the location of customer name index values.

    ```
    FIELD1=ul(1,1),lr(3.25,1.25),0
    ```

  - FIELD2 identifies the location of statement date index values.

```
FIELD2=ul(2,2),lr(2.75,2.25),0
```

- FIELD3 identifies the location of account number index values.

```
FIELD3=ul(2,2.25),lr(3.25,2.5),0
```

- FIELD4 identifies the location of the balance index values.

```
FIELD4=ul(2,3),lr(2.75,3.25),0
```

- Define indexes to identify the attribute name for an index value and the field parameter used to locate the index value.
  - INDEX1 identifies the customer name, for values extracted using FIELD1.

```
INDEX1='cust_name',FIELD1
```

  - INDEX2 identifies the statement date, for values extracted using FIELD2.

```
INDEX2='sdate',FIELD2
```

  - INDEX3 identifies the account number, for values extracted using FIELD3.

```
INDEX3='acct_num',FIELD3
```

  - INDEX4 identifies the balance, for values extracted using FIELD4.

```
INDEX4='balance',FIELD4
```

How do I create indexing parameters?

There are two parts to creating indexing parameters. First, process sample input data to determine the *x,y* coordinates of the text strings the PDF Indexer uses to identify groups and locate index data. Then, create the indexing parameters using the administrative client.

OnDemand provides the arspdump command to help you determine the location of trigger and field string values in the input data. The arspdump command processes one or more pages of sample report data and generates an output file. The output file contains one record for each text string on a page. Each record contains the *x,y* coordinates for a box imposed over the text string (upper left, lower right).

The process works as follows:

- Obtain a printed copy of the sample report.
- Identify the string values that you want to use to locate triggers and fields
- Identify the number of the page where each string value appears. The number is the *sheet number*, not the page identifier. The sheet number is the order of the page as it appears in the file, beginning with the number 1 (one), for the first page in the file. A page identifier is user-defined information that identifies each page (for example, iv, 5, and 17-3).
- Process one or more pages of the report with the arspdump command.

- In the output file, locate the records that contain the string values and make a note of the x,y coordinates.
- Create TRIGGER and FIELD parameters using the *x*,*y* coordinates, page number, and string value.

Indexing parameters are part of the OnDemand application. The administrative client provides an edit window you can use to maintain indexing parameters for the application.

Chapter 15. System considerations

System requirements

The OnDemand PDF Indexer requires OnDemand Version 2.2 and must be installed on an OnDemand server. The *Installation and Configuration Guide* for your operating system lists the OnDemand software and hardware requirements and describes how to install and configure OnDemand servers.

OnDemand provides the Adobe PostScript distilling and PDF viewing software. See the *Installation and Configuration Guide* for your operating system for information about installing and configuring the Adobe software on the OnDemand server.

System limitations

If you are using the PDF Indexer to generate index data for applications that create PostScript and PDF data, you need to take the following limitations into consideration:

- PDF files can contain a maximum of 32,756 pages
- DBCS languages are not currently supported
- Input data delimited with PostScript Passthrough markers cannot be indexed
- The Adobe Toolkit does not validate link destinations to other pages in a document or to other documents. Links may or may not resolve correctly, depending on how you segment your documents.
- To print PDF documents from the OnDemand server, you must use PSF for AIX Version 2.1 or later.
- If a font is referenced but not embedded in a PDF file, the Acrobat viewing software attempts to find the font using information contained in the PDF font descriptor. If the Acrobat viewing software finds the font, it uses the font to display the text. If the Acrobat viewing software does not find the font, it displays the text using a substitute Type 1 font.

Input data requirements

The PDF Indexer processes PDF input data. PostScript data generated by applications must be processed by Acrobat Distiller before you run the PDF Indexer. The online documentation provided with Acrobat Distiller describes methods you can use to generate PDF data.

You can use a variety of methods to provide the PDF Indexer with access to input data, including FTP and NFS. If you use a file transfer method to copy PDF data to the OnDemand server, transfer the files in binary format.

If you plan to automate the data indexing and loading process on the OnDemand server, the input file name must identify the application group and application to load. Use the following convention to name input files:

```
MVS.JOBNAME.DATASET.FORMS.YYDDD.HHMMSST.PDF
```

By default, the arsload command uses the FORMS part of the filename to identify the application group to load. You can use the -G parameter to specify a different part of the filename that identifies the application group. For example, `arsload -G JOBNAME`. If the application group contains more than one application, you must identify the application to load. Otherwise the load will fail. For example, to use the DATASET part of the filename to identify the application, run the arsload with the `-A DATASET` parameter. Choose one of the MVS, JOBNAME, DATASET, and FORMS parts of the filename to identify the application group and application.

Note: The case of the identifier PDF is ignored. Application group and application names are case sensitive and may include special characters such as the blank character.

NLS considerations

DBCS languages are not currently supported.

Data values specified on TRIGGER and FIELD parameters must be encoded in the same code page as the document. For example, if the characters in the document are encoded in code page 500, any data values you specify on TRIGGER and FIELD parameters must be encoded in code page 500. Examples of data values you might specify include TRIGGER string values and FIELD default and constant values.

For more information about NLS in OnDemand, see the *Installation and Configuration Guide* for your server operating system.

Chapter 16. Parameter reference

This parameter reference assumes that you will use the OnDemand data indexing and loading program to process input files. When you use the OnDemand data indexing and loading program to process input files, the PDF indexer ignores any values you supply for the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, and PARMDD parameters. If you run the arspdoci command from the prompt or call it from a user-defined program, you must provide values for the INPUTDD, OUTPUTDD, and PARMDD parameters and verify the default values for the INDEXDD and MSGDD parameters.

COORDINATES

Identifies the metrics used for *x,y* coordinates in the FIELD and TRIGGER parameters.

Required?
No

Default Value
In

Parameter syntax

COORDINATES=*metric*

Options and values

The *metric* can be:

In

The coordinate metrics are specified in inches (the default).

Cm

The coordinate metrics are specified in centimeters.

Mm

The coordinate metrics are specified in millimeters.

FIELD

Identifies the location of index data and can provide default and constant index values. You must define at least one field. You can define up to sixteen fields. There are two types of fields. A *trigger field* is based on the location of a trigger string value. A *constant field* provides the actual index value stored in the database.

Required?
Yes

Default Value
<none>

Trigger field syntax

FIELD*n*=**ul(***x*,*y***),lr(***x*,*y***),***page*[**,(TRIGGER=***n*,**BASE=**{**0** | **TRIGGER**}**, MASK='***field_mask***',DEFAULT='***value***')**]

Options and values

n

The field parameter identifier. When adding a field parameter, use the next available number, beginning with 1 (one).

ul(*x*,*y***)**

The coordinates for the upper left corner of the field string box. The field string box is the smallest rectangle that completely encloses the field string value (one or more words on the page). The PDF Indexer must find the field string value inside the field string box. The supported range of values is 0 to 45, page width and length, in inches.

lr(*x*,*y***)**

The coordinates for the lower right corner of the field string box. The field string box is the smallest rectangle that completely encloses the field string value (one or more words on the page). The PDF Indexer must find the field string value inside the field string box. The supported range of values is 0 to 45, page width and length, in inches.

page

The sheet number where the PDF Indexer begins searching for the field, relative to a trigger or 0 (zero) for the same page as the trigger. If you specify BASE=0, the *page* value can be –16 to 16. If you specify BASE=TRIGGER, the *page* value must be 0 (zero), which is relative to the sheet number where the trigger string value is located.

TRIGGER=*n*

Identifies the trigger parameter used to locate the field. This is an optional keyword, but the default is TRIGGER1. Replace *n* with the number of a defined TRIGGER parameter.

BASE={0|TRIGGER}

Determines whether the PDF Indexer uses the upper left coordinates of the trigger string box to locate the field. Choose from 0 (zero) or TRIGGER. If BASE=0, the PDF Indexer adds zero to the field string box coordinates. If BASE=TRIGGER, the PDF Indexer adds the upper left coordinates of the location of the trigger string box to the coordinates provided for the field string box. This is an optional keyword, but the default is BASE=0.

You should use BASE=0 if the field data always starts in a specific area on the page. You should use BASE=TRIGGER if the field is not always located in the same area on every page, but is always located a specific distance from a trigger. This capability is useful when the number of lines on a page varies, causing the location of field values to change. For example, given the following parameters:

```
TRIGGER2=ul(4,4),lr(5,8),1,'Total'
FIELD2=ul(1,0),lr(2,1),0,(TRIGGER=2,BASE=TRIGGER)
```

The trigger string value can be found in a one by four inch rectangle. The PDF Indexer always locates the field in a one inch box, one inch to the right of the location of the trigger string value. If the PDF Indexer finds the trigger string value in location `ul(4,4),lr(5,5)`, it attempts to find the field in location `ul(5,4),lr(6,5)`. If the PDF Indexer finds the trigger string value in location `ul(4,6),lr(5,7)`, it attempts to find the field in location `ul(5,6),lr(6,7)`.

MASK='*field_mask***'**

The pattern of symbols that the PDF Indexer matches to data located in the field. When you define a field that includes a mask, an INDEX parameter based on the field cannot reference any other fields. Valid mask symbols can include:

@ Matches alphabetic characters. For example:

```
MASK='@@@@@@@@@@@@@@@'
```

Causes the PDF Indexer to match a 15-character alphabetic field, such as a name.

Matches numeric characters. For example:

```
MASK='##########'
```

Causes the PDF Indexer to match a 10-character numeric field, such as an account number.

¬ Matches any non-blank character.

^ Matches any non-blank character.

% Matches the blank character and numeric characters.

= Matches any character.

DEFAULT='*value***'**

Defines the default index value, when there are no words within the coordinates provided for the field string box.

For example, assume that an application program generates statements that contain an audit field. The contents of the field can be PASSED or FAILED. However, if a statement has not been audited, the application program does not generate a value. In that case, there are no words within the field string box. To store a default value in the database for unaudited records, define the field as follows:

```
FIELD3=ul(8,1),lr(8.5,1.25),1,(DEFAULT='NOT AUDITED')
```

The PDF Indexer assigns the index associated with FIELD3 the value `NOT AUDITED`, if the field string box is blank.

Examples

The following field parameter causes the PDF Indexer to locate the field at the coordinates provided for the field string box. The field is based on TRIGGER1 and located on the same page as TRIGGER1. We specify BASE=0 because the field string box always appears in a specific location on the page.

```
TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 0001'
FIELD1=ul(1,1),lr(3.25,1.25),0,(TRIGGER=1,BASE=0)
```

Constant field syntax

FIELD*n***='***constant***'**

Options and values

n

The field parameter identifier. When adding a field parameter, use the next available number, beginning with 1 (one).

'*constant*'

The literal (constant) string value of the field. This is the index value stored in the database. The constant value can be 1 to 250 bytes in length. The PDF Indexer does not validate the type or content of the constant.

Examples

The following field parameter causes the PDF Indexer to store the same text string in each INDEX1 value it creates.

```
FIELD1='000000000'
INDEX1='acct',FIELD1
```

The following field parameters cause the PDF Indexer to concatenate a constant value with the index value extracted from the data. The PDF Indexer

concatenates the constant value specified in the FIELD1 parameter to each index value located using the FIELD2 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.

```
FIELD1='0000-'
FIELD2=ul(2,2),lr(2.5,2.25),0,(TRIGGER=1,BASE=0)
INDEX1='acct_num',FIELD1,FIELD2
```

Related parameters

INDEX parameter on page 230.

TRIGGER parameter on page 235.

FONTLIB

Identifies the directory or directories in which fonts are stored. Specify any valid path. The PDF Indexer searches for fonts in the order that the paths are listed. If a font is referenced in an input file but not embedded in the file, the PDF Indexer attempts to locate the font in the directory or directories listed on the FONTLIB parameter. If the font is located, the PDF Indexer adds it to the output file. If the font cannot be located, the Adobe viewing software displays the text using a substitute Type 1 font when the document is retrieved by a client program.

Required?

No

Default Value

/usr/lpp/Acrobat3/Fonts (AIX)

/opt/Acrobat3/Fonts (HP-UX, Solaris)

\PSFONTS (Windows NT)

Parameter syntax

FONTLIB=*pathlist*

Options and values

The *pathlist* is a colon-separated string of one or more valid path names. For example:

```
FONTLIB=/usr/lpp/Acrobat3/Fonts:/usr/lpp/ars/fontlib
```

The PDF Indexer searches the paths in the order in which they are specified. Delimit UNIX path names with the colon (:) character. Delimit Windows NT path names with the semicolon (;) character.

INDEX

Identifies the index name and the field or fields on which the index is based. You must specify at least one index parameter. You can specify up to 32 index parameters. When you create index parameters, we strongly encourage you to name the index the same as the application group database field name.

Required?
Yes

Default Value
<none>

Parameter syntax

INDEXn=’*name*’**,**FIELDnn[,...FIELDnn]

Options and values

n

The index parameter identifier. When adding an index parameter, use the next available number, beginning with 1 (one).

’*name*’

Determines the index name associated with the actual index value. For example, assume INDEX1 is to contain account numbers. The string *acct_num* would be a meaningful index name. The index value of INDEX1 would be an actual account number, for example, 000123456789.

The index name is a string from 1 to 250 bytes in length. We strongly encourage you to name the index the same as the application group database field name.

FIELD*nn*

The name of the field parameter or parameters ACIF uses to locate the index. A maximum of 32 field parameters can be specified. Separate field parameter names with a comma. The total length of all the specified field parameters cannot exceed 250 bytes.

Examples

The following index parameter causes the PDF Indexer to create group-level indexes for date index values (the PDF Indexer supports group-level indexes only). When the index value changes, the PDF Indexer closes the current group and begins a new group.

```
INDEX1='report_date',FIELD1
```

The following index parameters cause the PDF Indexer to create group-level indexes for customer name and account number index values. The PDF Indexer closes the current group and begins a new group when either the customer name or the account number index value changes.

```
INDEX1='name',FIELD1
INDEX2='acct_num',FIELD2
```

Related parameters

FIELD parameter on page 226.

INDEXDD

Determines the name or the full path name of the index object file, where the PDF Indexer writes indexing information. If you specify the file name without a path, the PDF Indexer puts the index object file in the current directory. If you do not specify the INDEXDD parameter, The PDF Indexer writes indexing information to the file INDEX.

Required?
No

Note: When you use the OnDemand data indexing and loading program to process files, the PDF indexer ignores any value you supply for the INDEXDD parameter. If you run the arspdoci command, verify the value of the INDEXDD parameter.

Default Value
INDEX

Parameter syntax

INDEXDD=*filename*

Options and values

The *filename* is a valid filename or full path name.

INDEXSTARTBY

Determines the page number by which the PDF Indexer must locate TRIGGER1 and match at least one of the INDEX parameters. The PDF Indexer stops processing if it does not locate TRIGGER1 and match at least one index by the specified page number. This parameter is optional, but the default is that the PDF Indexer must locate TRIGGER1 and match an index on the first

page of the input file.This parameter is helpful if the input file contains header pages. For example, if the input file contains two header pages, you can specify a page number one greater than the number of header pages (INDEXSTARTBY=3) so that the PDF Indexer will stop processing only if it does not locate TRIGGER1 and match an index by the third page in the input data.

Note: When you use INDEXSTARTBY to skip header pages, the PDF Indexer does not copy non-indexed pages to the output file or store them in OnDemand. For example, if you specify INDEXSTARTBY=3 and the first index is found on page three, pages one and two are not copied to the output file or stored in OnDemand. If you specify INDEXSTARTBY=3 and the first index is found on page two, page one is not copied to the output file (or stored in OnDemand).

Required?
No

Default Value
1

Parameter syntax

INDEXSTARTBY=*value*

Options and values

The *value* is the page number of the report by which the PDF Indexer must locate TRIGGER1 and match one or more of the INDEX parameters.

INPUTDD

Identifies the name or the full path name of the PDF input file that the PDF Indexer will process.

Required?
No

Note: When you use the OnDemand data indexing and loading program to process input files, the PDF indexer ignores any value you supply for the INPUTDD parameter. If you run the arspdoci command, you must specify a value for the INPUTDD parameter.

Default Value
<none>

Parameter syntax

INPUTDD=*name*

Options and values

The *name* is the file name or full path name of the input file. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the PDF Indexer searches the current directory for the specified file.

MSGDD

Determines the name or the full path name of the file where the PDF Indexer writes error messages. If you do not specify the MSGDD parameter, the PDF Indexer writes messages to standard error (UNIX) or the console (Windows NT).

Required?
No

Note: When you use the OnDemand data indexing and loading program to process input files, the PDF indexer ignores any value you supply for the MSGDD parameter. If you run the arspdoci command, verify the value of the MSGDD parameter.

Default Value
stderr

Parameter syntax

MSGDD=*name*

Options and values

The *name* is the file name or full path name where the PDF Indexer writes error messages. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the PDF Indexer places the error file in the current directory.

OUTPUTDD

Identifies the name or the full path name of the output file.

Required?
No

Note: When you use the OnDemand data indexing and loading program to process input files, the PDF indexer ignores any value you supply for the OUTPUTDD parameter. If you run the arspdoci command, you must specify a value for the OUTPUTDD parameter.

Default Value
<none>

Parameter syntax

OUTPUTDD=*name*

Options and values

The *name* is the file name or full path name of the output file. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the PDF Indexer puts the output file in the current directory.

PARMDD

Identifies the name or the full path name of the file that contains the indexing parameters used to process the input data.

Required?
No

Note: When you use the OnDemand data indexing and loading program to process input files, the PDF indexer ignores any value you supply for the PARMDD parameter. If you run the arspdoci command, you must specify a value for the PARMDD parameter.

Default Value
<none>

Parameter syntax

PARMDD=*name*

Options and values

The *name* is the file name or full path name of the file that contains the indexing parameters. On UNIX servers, file and path names are case sensitive. If you specify the file name without a path, the PDF Indexer searches for the file in the current directory.

TEMPDIR

Determines the name of the directory where the PDF Indexer writes temporary files.

Required?
No

Default Value
/arstmp (UNIX)

\arstmp (Windows NT)

Parameter syntax

TEMPDIR=*directory*

Options and values

The *filesystem* is a valid directory name.

TRIGGER

Identifies locations and string values required to uniquely identify the beginning of a group and the locations and string values of fields used to define indexes. You must define at least one trigger and can define up to 16 triggers.

Required?
Yes

Default Value
<none>

Parameter syntax

TRIGGERn**=ul(***x,y***),lr(***x,y***),***page***,'***value***'**

Options and values

n

The trigger parameter identifier. When adding a trigger parameter, use the next available number, beginning with 1 (one).

ul(*x,y***)**

The coordinates for the upper left corner of the trigger string box. The trigger string box is the smallest rectangle that completely encloses the trigger string value (one or more words on the page). The PDF Indexer must find the trigger string value inside the trigger string box. The supported range of values is 0 to 45, page width and length, in inches.

lr(*x*,*y*)

The coordinates for the lower right corner of the trigger string box. The trigger string box is the smallest rectangle that completely encloses the trigger string value (one or more words on the page). The PDF Indexer must find the trigger string value inside the trigger string box. The supported range of values are 0 to 45, page width and length, in inches.

page

The sheet number where the PDF Indexer begins to search for the trigger. The sheet number is the order of the page as it appears in the file, beginning with the number 1 (one), for the first page in the file. The *page* value can be an asterisk (*) or 0 to 16, relative to TRIGGER1. Specify an asterisk to search every page in the input file. You must specify an asterisk for TRIGGER1. The *page* value (0 to 16) for TRIGGER2 through TRIGGER16 is relative to TRIGGER1. For example, the page value of zero (0) means the trigger is located on the same page as TRIGGER1.

'*value*'

The actual string value the PDF Indexer uses to match the input data. The string value is case sensitive. The value is one or more words that can be found on a page.

Examples

TRIGGER1

The following TRIGGER1 parameter causes the PDF Indexer to search the specified location on every page of the input data for the specified string. You must define TRIGGER1 and the page value for TRIGGER1 must be an asterisk.

```
TRIGGER1=ul(0,0),lr(.75,.25),*,'Page 0001'
```

Group triggers

The following trigger parameter causes the PDF Indexer to attempt to match the string value `Account Number` within the coordinates provided for the trigger string box. The trigger can be found on the same page as TRIGGER1.

```
TRIGGER2=ul(1,2.25),lr(2,2.5),0,'Account Number'
```

The following trigger parameter causes the PDF Indexer to attempt to match the string value `Total` within the coordinates provided for the trigger string box. In this example, we've defined a one by four inch trigger string box, because the vertical position of the trigger on the page may vary. For example, assume that the page contains account numbers and balances with a total for all of the accounts listed. There can be one or more accounts listed. The location of the total varies, depending on the number of accounts listed. The field parameter is based on the trigger so that the PDF Indexer can locate the

field regardless of the actual location of the trigger string value. The field is a one inch box that always begins one inch to the right of the trigger. After locating the trigger string value, the PDF Indexer adds the upper left coordinates of the trigger string box to the coordinates provided for the field. The trigger can be found on the page following TRIGGER1.

```
TRIGGER2=ul(4,4),lr(5,8),1,'Total'
FIELD2=ul(1,0),lr(2,1),0,(TRIGGER=2,BASE=TRIGGER)
```

Related parameters

The FIELD parameter on page 226.

Chapter 17. Message reference

Introduction

The PDF Indexer creates a message list at the end of each indexing run. A return code of 0 (zero) means that processing completed without any errors.

The PDF Indexer detects a number of error conditions that can be logically grouped into several categories:

- **Informational**

  Informational messages contain a prefix of ARS6101I to ARS6199I.

  When the PDF Indexer processes a file, it issues informational messages that allow the user to determine if the correct processing parameters have been specified. These messages can assist in providing an audit trail.

- **Warning**

  Warning messages contain a prefix of ARS6301W to ARS6399W.

  The PDF Indexer issues a warning message and a return code of 4 (four) when the fidelity of the document may be in question.

- **Error**

  Error messages contain a prefix of ARS6501E to ARS6599E.

  The PDF Indexer issues an error message and return code of 8 (eight) or 16 (sixteen) and terminates processing the current input file. Most error conditions detected by the PDF Indexer fall into this category. The exact method of termination may vary. For certain severe errors, the PDF Indexer may fail with a segment fault. This is generally the case when some system service fails. In other cases, the PDF Indexer terminates with the appropriate error messages written either to standard error or to a file. When the PDF Indexer is invoked by the OnDemand data loading program, error messages are automatically written to the system log. If you run the arspdoci command, you can specify the name or the full path name of the file to contain processing messages with the **MSGDD** parameter.

- **Adobe Toolkit**

  Messages generated by the Adobe Toolkit contain a prefix of ARS6701 to ARS6799.

- **Internal Error**

  Internal error messages contain a prefix of ARS6901E to ARS6999E.

  The PDF Indexer issues an error message and return code of 16 (sixteen) and terminates processing the current input file.

ARS6101I *parameter* **- copy of input parameter**

Explanation: The parameter, options, and data values listed were used to process the input file.

System Action: None

User Response: No response is necessary.

ARS6102I **ARSPDOCI completed code** *code*

Explanation: The PDF indexer completed processing the input data with the completion code listed.

System Action: None

User Response: No response is necessary.

ARS6103I **Number of Input Pages =** *#pages*

Explanation: The *#pages* value is the number of pages contained in the input file.

System Action: None

User Response: No response is necessary.

ARS6104I **Created Document File** *output_filename*

Explanation: The file name listed is the name of the PDF output document file created.

System Action: None

User Response: No response is necessary.

ARS6501E *keyword* **keyword contains non-numeric identifier**

Explanation: The identifier for the keyword listed must be a number from 1 to 16 (TRIGGER) or 1 to 32 (INDEX, FIELD).

System Action: The PDF Indexer stops.

User Response: Correct the identifier and resubmit the PDF Indexer job.

ARS6502E **Unknown Parameter:** *string*

Explanation: The string listed is not a valid PDF Indexer parameter.

System Action: The PDF Indexer stops.

User Response: Refer to “Chapter 16. Parameter reference” on page 225 for a list of valid parameters. Correct the parameter and resubmit the PDF Indexer job.

ARS6503E **Incorrect** *file_def* **File definition**

Explanation: The file name specified on the INDEXDD, MSGDD, or OUTPUTDD file definition parameter is not a valid file name.

System Action: The PDF Indexer stops.

User Response: Correct the file name specified on the file definition parameter and resubmit the PDF Indexer job.

ARS6504E **Incomplete Indexing information supplied**

Explanation: The current set of indexing parameters does not permit the PDF Indexer to create index data.

System Action: The PDF Indexer stops.

User Response: Correct the parameters and resubmit the PDF Indexer job.

ARS6505E **Trigger1, Index1, or Field1 not defined**

Explanation: You attempted to index the input file but did not define TRIGGER1, INDEX1, or FIELD1. You must define at least one trigger and the identifier must be TRIGGER1. You must define at least one index and the identifier must be INDEX1. You must define at least one field and the identifier must be FIELD1.

System Action: The PDF Indexer stops.

User Response: Correct the parameter and resubmit the PDF Indexer job.

ARS6506E *parameter identifier* **parameter syntax incorrect.**

Explanation: The syntax for the trigger, field, or index parameter listed is not correct.

System Action: The PDF Indexer stops.

User Response: Correct the parameter and resubmit the PDF Indexer job. See the "Chapter 16. Parameter reference" on page 225) for details.

ARS6507E **Trigger(s) not found by page** *page#*

Explanation: The PDF Indexer did not find a trigger by the specified page number. The INDEXSTARTBY parameter determines the page number by which the PDF Indexer must find a trigger and begin indexing. Refer to "INDEXSTARTBY" on page 231 for details.

System Action: The PDF Indexer fails.

User Response: Verify the page number specified on the INDEXSTARTBY parameter. If the page number is correct, verify the TRIGGER parameters. Then resubmit the PDF Indexer job.

ARS6508E **Error allocating** *size* **bytes memory**

Explanation: The PDF Indexer was unable to allocate the requested amount of memory.

System Action: The PDF Indexer stops.

User Response: Decrease the load on the system or increase the amount of memory available to the PDF Indexer and resubmit the PDF Indexer job.

ARS6509E **Error** *operation file_def* **file** *file_name***.**

Explanation: The PDF Indexer encountered a file error. The *operation* identifies the operation attempted (open, write, read, or close). The *file_def* identifies the file definition parameter (INPUTDD, OUTPUTDD, or INDEXDD). The *file_name* identifies the name of the file.

System Action: The PDF Indexer stops.

User Response: Verify the file definition parameter. If the file definition parameter is correct, verify the directory permissions. If the directory permissions are correct, verify there is sufficient free space available in the filesystem to store the data. Then resubmit the PDF Indexer job.

ARS6510E **Document not indexed, no matching triggers found.**

Explanation: The PDF Indexer was not able to match the values specified on the trigger parameters to data in the input file.

System Action: The PDF Indexer stops.

User Response: Correct the coordinates, page numbers, and string values specified on the trigger parameters and resubmit the PDF Indexer job.

ARS6511E **The input file contains an unsupported data type**

Explanation: The input file does not contain PostScript or PDF data.

System Action: The PDF Indexer stops.

User Response: Verify that the correct file is named on the INPUTDD parameter. Verify that the file named on the INPUTDD parameter contains PostScript or PDF data. Then resubmit the PDF Indexer job.

ARS6701I *product* **version** *version.release*

Explanation: A message that displays the product version and release.

System Action: None

User Response: No response is necessary.

ARS6702E **Failed Adobe Tookit initialization rc=***return_code* **Error String:** *error_text*

Explanation: The Adobe toolkit returned an error.

System Action: The PDF Indexer stops.

User Response: Verify the directories specified

on the FONTLIB and TEMPDIR parameters. Verify the directory permissions. Verify that the directories named on the FONTLIB parameter provide access to the fonts required by the PDF Indexer. Verify that the directory named on the TEMPDIR parameter contains sufficient free space to process the input file. If you cannot resolve the problem, contact the IBM support center.

ARS6703E **Adobe** *error_number* **API error. Error string:** *error_text*

Explanation: The Adobe toolkit returned an error.

System Action: The PDF Indexer stops.

User Response: Verify the directories specified on the FONTLIB and TEMPDIR parameters. Verify the directory permissions. Verify that the directories named on the FONTLIB parameter provide access to the fonts required by the PDF Indexer. Verify that the directory named on the TEMPDIR parameter contains sufficient free space to process the input file. If you cannot resolve the problem, contact the IBM support center.

ARS6704E **Create of New Document Segment failed. Error string:** *error_text*

Explanation: The Adobe toolkit returned an error when trying to create a new document segment.

System Action: The PDF Indexer stops.

User Response: Verify the directories specified on the FONTLIB and TEMPDIR parameters. Verify the files and directories named on the INPUTDD and OUTPUTDD parameters. Verify the file and directory permissions. Verify that the directories named on the FONTLIB parameter provide access to the fonts required by the PDF Indexer. Verify that the directory named on the TEMPDIR and OUTPUTDD parameters contain sufficient free space to process the input and output files. If you cannot resolve the problem, contact the IBM support center.

ARS6705E **Page Extraction Failed. Error string:** *error_text*

Explanation: The Adobe toolkit returned an error when trying to extract pages for a new document.

System Action: The PDF Indexer stops.

User Response: Verify the directories specified on the FONTLIB and TEMPDIR parameters. Verify the files and directories named on the INPUTDD and OUTPUTDD parameters. Verify the file and directory permissions. Verify that the directories named on the FONTLIB parameter provide access to the fonts required by the PDF Indexer. Verify that the directory named on the TEMPDIR and OUTPUTDD parameters contain sufficient free space to process the input and output files. If you cannot resolve the problem, contact the IBM support center.

ARS6706E **Word Search and Extraction Error. Error string:** *error_text*

Explanation: The Adobe toolkit returned an error while searching the PDF document.

System Action: The PDF Indexer stops.

User Response: Verify the directories specified on the FONTLIB and TEMPDIR parameters. Verify the directory permissions. Verify that the directories named on the FONTLIB parameter provide access to the fonts required by the PDF Indexer. Verify that the directory named on the TEMPDIR parameter contains sufficient free space to process the input file. If you cannot resolve the problem, contact the IBM support center.

ARS6707E **Error during distill rc=***return_code***. Check the Distiller messages.**

Explanation: The Acrobat Distiller returned an error while trying to distill the input file.

System Action: The PDF Indexer stops.

User Response: Use the Distiller output messages to determine the cause and resolution of the error. After correcting the error, resubmit the PDF Indexer job.

ARS6901E **ARPSDOCI Internal error - module** *module* **function** *function*

Explanation: The PDF Indexer encountered an internal error.

System Action: The PDF Indexer stops.

User Response: Contact the IBM support center.

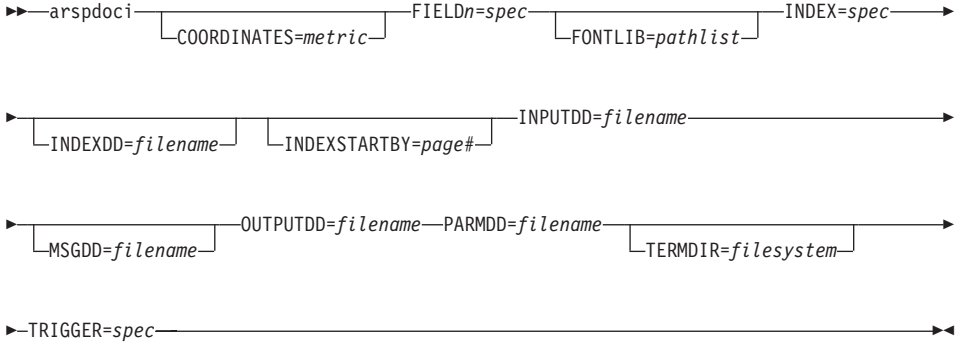
Chapter 18. arspdoci command reference

Purpose

Generate index data for a PDF file.

Syntax

Note: The following syntax should be used only when you run the arspdoci command from the command line or call it from a user-defined program.



Description

The arspdoci command can be used to index a PDF file. The OnDemand data indexing and loading program calls the arspdoci command if the input data type is PDF and the indexer is PDF. If you need to index a PDF and you do not want to use the OnDemand data indexing and loading program, you can run the arspdoci command from the command line or call it from a user-defined program.

Parameters

Refer to “Chapter 16. Parameter reference” on page 225 for details about the parameters that you can specify when you run the arspdoci command from the command line or a user-defined program.

Files

/usr/lpp/ars/bin/arspdoci
The AIX executable program.

/opt/ondemand/bin/arspdoci
The HP-UX and Solaris executable program.

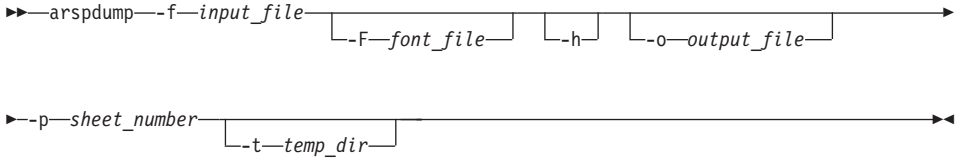
\Program Files\IBM\OnDemand for WinNT\bin\arspdoci
The Windows NT Server executable program.

Chapter 19. arspdump command reference

Purpose

Print the locations of text strings on a page.

Syntax



Description

The arspdump command can be used to identify the locations of text strings on a page in a PDF file. You can use this command to help define triggers and fields. When you define triggers and fields, you must specify the location of the string value used to locate the trigger or field as x and y pairs in a coordinate system imposed on the page. For each string value, you must identify the upper left and lower right position on the page. The output of the arspdump command contains a list of the text strings on the page and the coordinates for each string. If a font is referenced in a PDF file, but not embedded, the arspdump command attempts to find the font using information provided with the -F flag. If the arspdump command does not find the font, it uses a substitute Type 1 font.

Parameters

-f input_file
The file name or full path name of the PDF file to process. On UNIX servers, file and path names are case sensitive.

-F font_dir
Identifies directories in which fonts are stored. Specify any valid path. On UNIX servers, use the color (:) character to separate path names. On Windows NT servers, use the semicolon (;) character to separate path names. The arspdump command searches the paths in the order in which they are specified. If you do not specify this flag and name a

font directory, the arspdump attempts to locate fonts in the /usr/lpp/Acrobat3/Fonts directory (AIX), /opt/Acrobat3/Fonts directory (HP-UX, Solaris), or \PSFONTS directory (Windows NT).

-h List the arspdump command line flags and descriptions.

-o output_file
The file name or full path name of the file where the arspdump command writes output messages. On UNIX servers, file and path names are case sensitive. If you do not specify this flag and name a file, the arspdump command writes output to stdout (UNIX) or the console (Windows NT).

-p sheet_number
The number of the page in the PDF file that you want the arspdump command to process. This is the page that contains the text strings that you want to use to define triggers and fields. The sheet number is the order of the page as it appears in the file, beginning with the number 1 (one), for the first page in the file. Contrast with page identifier, which is user-defined information that identifies each page (for example, iv, 5, and 17-3).

-t temp_dir
Identifies the directory that the arspdump command uses to store work files. Specify any valid directory name. If you do not specify this flag and name a directory, the arspdump stores work files in the /arstmp directory (UNIX servers) or the \arstmp directory (Windows NT servers).

Examples

1. The following example shows how to invoke the arspdump command to print the strings and locations of text found on page number one of sample.pdf to sample.out:

```
arspdump -f sample.pdf -o sample.out -p 1
```

2. The following example shows how to invoke the arspdump command to print the strings and locations of text found on page number three of sample.pdf to sample.out:

```
arspdump -f sample.pdf -o sample.out -p 3
```

Files

/usr/lpp/ars/bin/arspdump
The AIX executable program.

/opt/ondemand/bin/arspdump
The HP-UX and Solaris executable program.

\Program Files\IBM\OnDemand for WinNT\bin\arspdump
The Windows NT Server executable program.

Part 4. Appendixes

Glossary

This glossary includes definitions from the following sources:

- Definitions reprinted from the *American National Dictionary for Information Processing Systems* , copyright 1982 by the Computer Business Equipment Manufacturers Association (CBEMA), are identified by the symbol (A) following the definition. Copies can be purchased from the American Standards Institute, 1430 Broadway, New York, New York 10018.
- Definitions reprinted from a published section of the International Organization for Standardization's (ISO) *Vocabulary—Information Processing* or from a published section of the ISO *Vocabulary—Office Machines* are identified by the symbol (I) following the definition. Because many ISO definitions are also reproduced in the *American National Dictionary for Information Processing Systems* ISO definitions may also be identified by the symbol (A).
- Definitions reprinted from working documents, draft proposals, or draft international standards of ISO Technical Committee 97, Subcommittee 1 (Vocabulary) are identified by the symbol (T) following the definition, indicating that final agreement has not yet been reached among its participating members.
- Definitions are also reprinted from the *CCITT Eighth Plenary Assembly Red Book, Terms and Definitions* and working documents published by the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union, Geneva, 1985.
- Definitions that are specific to IBM products are so labeled, for example, "In OnDemand," or "In MVS."

The following cross references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in its proper place in the dictionary.

Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.

See. This refers the reader to multiple-word terms that have the same last word.

See also. This refers the reader to terms that have a related, but not synonymous, meaning.

A

access. To obtain data from or to put data in storage.

ACIF. See Advanced Function Presentation Conversion and Indexing Facility.

Acrobat. The Adobe viewer for PDF files. Acrobat is similar to the IBM AFP Workbench, that is, a stand-alone viewer. Acrobat also supports a robust set of APIs. It is through these APIs that Acrobat is integrated with the OnDemand client program.

active log file. The subset of files consisting of primary log files and secondary log files that are currently needed by the database manager for rollbacks and recovery.

active policy set. In ADSM, the policy set within the policy domain that contains the most

recently activated policy currently in use by all client nodes that have been assigned to that policy domain. See Policy Set.

adapter. A part that electrically or physically connects a device to a computer or to another device.

addressable point. Any point in a presentation surface that can be identified by a coordinate from the coordinate system of the presentation medium. See also Pel.

administrative client. (1) In ADSM, the program that allows administrators to control and monitor the server through administrator commands. (2) In OnDemand, the program that provides administrators with functions to manage OnDemand groups, users, printers, applications, application groups, storage sets, and folders.

ADSM. See ADSTAR Distributed Storage Manager.

ADSTAR Distributed Storage Manager. A program that provides storage management for archived files.

Advanced Function Presentation (AFP). A set of licensed programs that use the all-points-addressable concept to print data on a wide variety of printers or display data on a variety of display devices. AFP also includes creating, formatting, archiving, viewing, retrieving, and distributing information.

Advanced Function Presentation Application Programming Interface (AFP API). An AFP program shipped with PSF/MVS 2.1.1 and PSF/VM 2.1.1 that creates the AFP data stream from the COBOL and PL/1 high-level programming languages.

Advanced Function Presentation Conversion and Indexing Facility. An AFP program shipped with OnDemand that you can use to convert a print file into a MO:DCA-P document, to retrieve resources used by the document, and to index the file for later retrieval and viewing.

Advanced Function Presentation data stream (AFP data stream). A presentation data stream that is processed in the AFP environment. MO:DCA-P is the strategic AFP interchange data stream. IPDS is the strategic AFP printer data stream.

AFP. See Advanced Function Presentation.

AFP API. See Advanced Function Presentation Application Programming Interface.

AFPDS. A term formerly used to identify the composed page, MO:DCA-P-based data stream interchanged in AFP environments.

AIX. (1) Advanced Interactive Executive. (2) IBM's version of the UNIX operating system.

AIX Acrobat Libraries. A subset of the Acrobat Libraries ported to AIX for use by OnDemand.

all-points-addressable (APA). The capability to address, reference, and position data elements at any addressable position in a presentation space or on a physical medium. An example of all points addressability is the positioning of text, graphics, and images at any addressable point on the physical medium. See also Picture Element.

all-points-addressable mode. Synonym for Page Mode.

alphabetic character. A letter or other symbol, excluding digits, used in a language. Usually the uppercase and lowercase letters A through Z plus other special symbols (such as $ and _) allowed by a particular language. See also Alphanumeric Character.

alphanumeric character. Consisting of letters, numbers, and often other symbols, such as punctuation marks and mathematical symbols. See also Alphabetic Character.

alphanumeric string. A sequence of characters consisting solely of the letters a through z and the numerals 0 through 9.

American National Standards Institute (ANSI). An organization for the purpose of establishing voluntary industry standards.

anchor point. The point in a document that signals to ACIF the beginning of a group of pages, after which it adds indexing structured fields to delineate this group.

ANSI. American National Standards Institute.

ANSI carriage control character. A character that specifies that a write, space, or skip operation should be performed before printing the line containing the carriage control. ANSI carriage control characters are encoded in ASCII or EBCDIC.

APA. All points addressable.

API. Application Program Interface.

application. In OnDemand, a definition of the attributes of a report, such as the data format and the compressing method. An application is defined for each output print data stream to be stored in OnDemand.

application group. One or more OnDemand applications with common indexing and storage management attributes, for example, invoice number and life of the data in OnDemand.

Application Program Interface (API). A formally defined programming language interface that is between a program and the user of a program.

archive copy group. In ADSM, a policy object containing attributes that control the generation, destination, and expiration of archive files. An archive copy group belongs to a management class.

archive log file. The subject of files consisting of primary log files and secondary log files that are no longer needed for normal database processing.

archive media. Devices and volumes used to store long-term copies of reports. For example, an optical storage library is one type of archive media supported by OnDemand.

ASCII (American Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

attachment. A device or feature attached to a processing unit, including required adapters. Contrast with Adapter.

authentication. The process of checking a user's password before allowing the user access to resources or the server.

authorize. (1) To grant to a user the right to communicate with or make use of a computer system or display station. (2) To give a user either complete or restricted access to an object, resource, or function.

B

backend. In the AIX operating system, the program that sends output to a particular device. Synonymous with Backend Program.

backend program. Synonym for Backend.

bitmap. A file that contains a bit-mapped graphic.

BMP. Bitmap.

byte. The amount of storage required to represent 1 character; a byte is 8 bits.

C

cache. Short-term, magnetic storage. OnDemand loads the most recent and frequently used versions of reports and documents in the cache.

carriage control character. The first character of an output record (line) that is to be printed; it determines how many lines should be skipped before the next line is printed.

case-sensitive. Able to distinguish between uppercase and lowercase letters.

CCITT. Consultative Committee on International Telegraphy and Telephone.

CD-ROM. Compact disc read-only memory.

channel. A device connecting the processor to input and output devices.

channel adapter. A communication controller hardware unit used to attach the controller to a System/370 data channel.

channel-attached. (1) Pertaining to devices attached to a controlling unit by cables, rather than by telecommunication lines. (2) Synonymous with Local.

character. A letter, digit, or other symbol representing, organizing, or controlling data.

character rotation. The alignment of a character with respect to its character baseline, measured in degrees in a clockwise direction. Examples are 0°, 90°, 180°, and 270°. Zero-degree character rotation exists when a character is in its customary alignment with the baseline.

character set. A group of characters used for a specific reason; for example, the set of characters a printer can print or a keyboard can support.

click. To press the left mouse button while pointing to an object such as a command button or a toolbar button.

client. (1) In a distributed file system environment, a system that is dependent on a server to provide it with programs or access to programs. (2) A personal computer connected to a network running OnDemand software that can log on and query the library server, retrieve documents from OnDemand, and view and print documents.

client domain. The set of optical drives and storage volumes used by ADSM to store report files and resources belonging to an application group.

client node. An application group that has been registered to the ADSM server.

COBOL. Common business-oriented language. A high-level programming language, based on English, that is used primarily for business applications.

code page. An ordered set of up to 256 predefined display symbols. The first 32 code points of each code page are reserved for control codes and are the same for all code pages, leaving up to 224 distinct display symbols per page.

Code Page Global Identifier (CPGID). A unique code page identifier that can be expressed as either a two-byte binary or a five-digit decimal value.

code point. A character within a code page.

coded font. An AFP font that associates a code page and a font character set.

command. A request to perform an operation or run a program. When parameters values, flags, or other operands are associated with a command, the resulting character string is a single command.

command line. The area of the screen where commands are displayed as they are typed.

communication method. The method used by OnDemand and ADSM for exchanging information.

communication protocol. A set of defined interfaces that allow computers to communicate with each other.

compact disc read-only memory (CD-ROM). High capacity read-only memory in the form of an optically read compact disk.

composed page. In Advanced Function Presentation, a page that can be printed only on an all-points-addressable output medium. It may contain composed text and raster images.

composed-text data file. A file containing text data and text control information that dictates the format, placement, and appearance of the data to be printed.

compression. A technique for removing strings of duplicate characters, gaps, empty fields, and trailing blanks before transmitting data.

concatenate. (1) To link together. (2) To join two character strings.

concatenated field. Two or more fields from a physical file record format that have been combined to make one field in a logical file record format.

conditional processing. A page definition function that allows input data records to partially control their own formatting.

configuration. The process of describing to a system the devices, optional features, and program products that have been installed so that these features can be used. Contrast with Customization.

configuration file. A file that specifies the characteristics of a system or subsystem; for example, the operating system queueing system.

configure. To describe to a system the devices, optional features, and licensed programs installed on a system.

console. The main operating system display station.

constant. A data item with a value that does not change during the running of a program. Contrast with Variable.

Consultative Committee on International Telegraphy and Telephone (CCITT). A United Nations Specialized Standards group whose membership includes common carriers concerned with devising and proposing recommendations for international telecommunications representing alphabets, graphics, control information, and other fundamental information interchange issues.

control character. A character that is not a graphic character such as a letter, number, or punctuation mark. Such characters are called control characters because they frequently act to control a peripheral device.

controller. A device that coordinates and controls the operation of one or more input/output devices, such as workstations, and synchronizes the operation of the system as a whole.

conversion. In programming languages, the transformation between values that represent the same data item but belong to different data types.

copies. See Copy Group.

copy group. In ADSM, a policy object that contains attributes that control the generation, destination, and expiration of backup and archive files. There are two kinds of copy groups: backup and archive. Copy groups belong to management classes.

copy storage pool. A named collection of storage volumes that contains copies of files that reside in primary storage pools. Copy storage pools are used to back up the data stored in primary storage pools.

CPGID. Code Page Global Identifier

customization. The process of describing optional changes to defaults of a software program that is already installed on the system and configured so that it can be used. Contrast with Configuration.

customize. To describe the system, the devices, programs, users, and user defaults for a particular data processing system or network. Contrast with Configure.

D

daemon. In UNIX, a process begun by the root user or by the root shell that can be stopped only by the root user. Daemon processes generally provide services that must be available at all

times, such as sending data to the printer. A daemon runs continuously, looking for work to do, performing that work, and waiting for more work. A daemon does not have a controlling terminal associated with it. The OnDemand data download program (arsjesd) is an example of a daemon.

database. (1) The collection of information about all objects managed by OnDemand, including documents, groups, users, printers, application groups, storage sets, applications, and folders. (2) The collection of information about all objects managed by ADSM, including policy management objects, administrators, and client nodes.

Database Managed Space (DMS). A type of DB2 table space. A DSM table space is managed by the database manager.

data set. Synonym for File.

data stream. A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form using a defined format.

data transfer. The movement, or copying, of data from one location and the storage of the data at another location.

data type. The type, format, or classification of a data object.

DCF. Document Composition Facility.

decimal. Pertaining to a system of numbers to the base 10. The decimal digits range from 0 through 9.

decompression. A function that expands data to the length that preceded data compression. See also Compression.

default. A value, attribute, or option that is assumed when no alternative is specified by the user.

default directory. The directory name supplied by the operating system if none is specified.

default printer. A printer that accepts all the printed output from a display station assigned to it.

default value. A value stored in the system that is used when no other value is specified. See also Default.

desktop printer. In this publication, an IBM LaserPrinter 4019 or 4029, or compatible printer.

device class. A named group of ADSM storage devices. Each device class has a unique name and represents a device type of disk, tape, or optical disk.

device driver. A program that operates a specific device, such as a printer, disk drive, or display.

device type. A type of ADSM storage device. Each device class must be categorized with one of the following devices types: disk, tape, or optical disk.

device-independent. Pertaining to a function that can be accomplished without regard for the characteristics of particular types of devices.

dialog box. An application window on the display that requests information from the user.

directory. (1) A type of file containing the names and controlling information for other files or directories. (2) A listing of related files arranged in a useful hierarchy.

disk operating system (DOS). An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data.

Distiller. A batch utility that converts PostScript files to PDF files. The distiller runs under AIX, HP-UX, Solaris, and Windows NT.

DMS. Database Managed Space.

document. (1) In OnDemand, a logical section of a larger file, such as an individual invoice within a report file of thousands of invoices. A document can also represent a 100 page segment

of a report. (2) A file containing an AFP data stream document. An AFP data stream document is bounded by Begin Document and End Document structured fields and can be created using a text formatter such as Document Composition Facility (DCF).

Document Composition Facility. An IBM licensed program used to prepare printed documents.

domain. See Policy Domain or Client Domain.

DOS. Disk operating system.

double-click. To rapidly press the left mouse button twice while pointing to an object.

download. To transfer data from one computer for use on another one. Typically, users download from a larger computer to a diskette or fixed disk on a smaller computer or from a system unit to an adapter.

drag. To hold down the left mouse button while moving the mouse.

driver. The end of a stream closest to an external interface. The principal functions of the driver are handling any associated device, and transforming data and information between the external device and stream.

E

EBCDIC. Extended Binary-Coded Decimal Interchange Code. This is the default type of data encoding in an MVS environment. Contrast with ASCII.

enqueue. To place items in a queue.

enter. (1) An instruction to type specific information using the keyboard. (2) A keyboard key that, when pressed, confirms or initiates the selected command.

environment variable. A variable that is included in the current software environment and is therefore available to any called program that requests it.

error condition. The state that results from an attempt to run instructions in a computer program that are not valid or that operate on data that is not valid.

error log. A file in a product or system where error information is stored for later access.

error log entry. In AIX, a record in the system error log describing a hardware or software failure and containing failure data captured at the time of the failure.

error message. An indication that an error has been detected. (A)

error recovery. The process of correcting or bypassing the effects of a fault to restore a computer system to a prescribed condition. (T)

error type. Identifies whether an error log entry is for a permanent failure, temporary failure, performance degradation, impending loss of availability, or undetermined failure.

Ethernet. A 10-megabit baseband local area network using CSMA/CD (carrier sense multiple access with collision detection). The network allows multiple stations to access the medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and transmission.

exit program. A user-written program that is given control during operation of a system function.

exit routine. A routine that receives control when a specified event occurs, such as an error.

expiration. The process of deleting index data and reports based on application configuration information. The OnDemand database and storage managers perform expiration processing to remove reports and documents that are no longer needed from storage volumes and reclaim the space.

Extended Binary-Coded Decimal Interchange Code (EBCDIC). A coded character set consisting of eight-bit coded characters.

external library resource (member). Objects that can be used by other program products while running print jobs; for example, coded fonts, code pages, font character sets, form definitions, page definitions, and page segments. Synonym for Resource Object.

external object. Synonym for Resource Object.

F

FCB. Forms control buffer.

field. A specified area in a record used for a particular type of data; for example, a group of characters that represent a customer's name.

file. (1) A named set of records stored or processed as a unit. (T) (2) The major unit of data storage and retrieval. A file consists of a collection of data in one of several prescribed arrangements and described by control information to which the operating system has access.

file system. The collection of files and file management structures on a physical or logical mass storage device, such as a diskette or a minidisk.

file transfer. In remote communications, the transfer of a file or files from one system to another over a communications link.

File Transfer Protocol (FTP). In TCP/IP, the protocol that makes it possible to transfer data among hosts and to use foreign hosts indirectly.

fixed disk. A flat, circular, nonremovable plate with a magnetizable surface layer on which data can be stored by magnetic recording. A rigid magnetic disk.

fixed-disk drive. The mechanism used to read and write information on a fixed disk.

folder. In OnDemand, the end-user view of data archived in one or more application groups. Folders provide end-users a convenient way to find related information, regardless of the source of the information or how the data was prepared.

font. (1) A family of characters of a given size and style, for example 9-point Helvetica. (2) A set of characters in a particular style. See Raster Font.

font character set. Part of an AFP font that contains the raster patterns, identifiers, and descriptions of characters. Often synonymous with Character Set. See also Coded Font.

form definition (FORMDEF). A form definition is a resource used by OnDemand. A form definition specifies the number of copies to be printed, whether the sheet should be printed on both sides, the position of a page of data on the sheet, text suppression, and overlays to be used (if any). Synonymous with FORMDEF.

FORMDEF. See Form Definition.

FSA. Functional SubSystem Application. A collection of programs residing in the FSS address space that control a device.

FSI. Functional SubSystem Interface. An MVS or OS/390 interface that allows communication between JES and a FSS and FSS applications. Download uses an FSI to communicate with the operating system and JES to process spool datasets created by application programs.

FSS. Functional SubSystem. An MVS or OS/390 subsystem comprised of programs residing in the same address space that provide JES-related functions. For example, a print programs that extend the scope of JES processing could be defined as a FSS.

FTP. File Transfer Protocol.

G

GB. Gigabyte.

GIF. Graphic Interchange Format.

gigabyte. A unit of memory or space measurement equal to approximately one billion bytes. One gigabyte equals 1,000 megabytes.

graphic. A symbol produced by a process such as handwriting, drawing, or printing. (I) (A)

graphic character. A character that can be displayed or printed.

Graphical User Interface. A type of user interface that takes advantage of a high-resolution monitor, including some combination of graphics, the use of pointing devices, menu bars, overlapping windows, and icons.

graphics. A type of data created from such fundamental drawing units such as lines, curves, polygons, and so forth.

Graphic Interchange Format (GIF). A bit-mapped color graphics file format for IBM and IBM-compatible computers. GIF employs an efficient compression technique for high resolution graphics.

group. (1) A named collection of sequential pages that form a logical subset of a document. (2) A named collection of users assigned a specific role or belonging to a specific department.

GUI. Graphical user interface.

H

hardware. The physical equipment of computing and computer-directed activities. The physical components of a computer system. Contrast with Software.

help. One or more files of information that describe how to use application software or how to perform a system function.

hex. See Hexadecimal.

hexadecimal (hex). Pertaining to a system of numbers in the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

host. (1) The primary or controlling computer in the communications network. (2) See Host System.

host-based computer. (1) In a computer network a computer that provides end users with services such as computation and data bases and that usually performs network control functions. (T) (2) The primary or controlling computer in a multiple-computer installation.

host system. (1) The controlling or highest level system in a data communication configuration, for example, an OS/390 system is the host system for the terminals connected to it. (2) In TCP/IP, a computer that is a peer system in a network.

I

icon. A 32 by 32 pixel bitmap used by the windows manager to represent an application or other window.

image. (1) An electronic representation of a picture produced by means of sensing light, sound, electron radiation, or other emanations coming from the picture or reflected by the picture. An image can also be generated directly by software without reference to an existing picture. (2) An electronic representation of an original document recorded by a scanning device.

index. (1) A process of segmenting a print file into uniquely identifiable groups of pages (a named collection of sequential pages) for later retrieval. (2) A process of matching reference points within a file and creating structured field tags within the MO:DCA-P document and the separate index object file.

index object file. An index-information file created by the OnDemand Conversion and Indexing Facility that contains the Index Element (IEL) structured fields, which identify the location of tagged groups in the AFP file. The indexing tags are contained in the Tagged Logical Element (TLE) structured fields.

indexing. (1) A process of segmenting a print file into uniquely identifiable groups of pages (a named collection of sequential pages) for later retrieval. (2) 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.

informational message. (1) A message that provides information to the end-user or system administrator but does not require a response. (2) A message that is not the result of an error condition.

input file. A file opened in order to allow records to be read.

install. (1) To add a program, program option, or software program to the system in a manner such that it may be executed and will interact properly with all affected programs in the system. (2) To connect a piece of hardware to the processor.

intelligent printer data stream (IPDS). An all-points-addressable data stream that allows users to position text, images, and graphics at any defined point on a printed page.

interface. Hardware, software, or both, that links systems, programs, or devices.

Internet. A wide area network connecting thousands of disparate networks in industry, education, government, and research. The Internet network uses TCP/IP as the protocol for transmitting information.

Internet Protocol (IP). In TCP/IP, a protocol that routes data from its source to its destination in an Internet environment.

IP. Internet Protocol.

IPDS. Intelligent printer data stream.

J

job. One or more related procedures or programs grouped into a procedure, identified by appropriate job control statements.

job queue. A list of jobs waiting to be processed by the system.

Joint Photographic Experts Group (JPEG). An image compression standard developed to handle larger images with many colors. JPEG uses a lossy algorithm, which means there is some loss of detail when saving and viewing images in this format. However, JPEG files can offer as much as 35% improvement in file size and compression.

JPEG. See Joint Photographic Experts Group.

K

kernel. The part of an operating system that performs basic functions such as allocating hardware resources.

kernel extension. A program that modifies parts of the kernel that can be customized to provide additional services and calls. See Kernel.

K-byte. See Kilobyte.

keyword. Part of a command operand that consists of a specific character string.

kilobyte (K-byte). 1024 bytes in decimal notation when referring to memory capacity; in all other cases, it is defined as 1000.

L

LAN. Local area network.

LAN server. A data station that provides services to other data stations on a local area network; for example, file server, print server, mail server. (T)

laser printer. A nonimpact printer that creates, by means of a laser beam directed on a photosensitive surface, a latent image which is then made visible by toner and transferred and fixed on paper. (T)

Lempel Ziv Welsh (LZW). A data compression algorithm. OnDemand uses the 16-bit version of LZW to compress data.

library. System storage for generated form definitions and page definitions.

library resource (member). A named collection of records or statements in a library.

library resource name. A name by which an object may be called from a library by Advanced Function Printing as part of a print job. Includes the 2-character prefix for the type of object, such as P1 for page definitions, F1 for form definitions, or O1 for overlays (also known as *resource name*).

library server. In OnDemand, the node in the TCP/IP network that contains the core OnDemand code and the OnDemand database.

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. Data prepared for printing on a line printer, such as an IBM 3800 Model 1 Printing Subsystem. Line data is usually characterized by carriage-control characters and table reference characters.

line-data print file. A file that consists of line data, optionally supplemented by a limited set of structured fields.

line printer. A device that prints a line of characters as a unit. (I) (A) Contrast with Page Printer.

literal. (1) A symbol or a quantity in a source program that is itself data, rather than a reference to data. (2) A character string whose value is given by the characters themselves; for example, the numeric literal 7 has the value 7, and the character literal CHARACTERS has the value CHARACTERS.

loading. The logical process of archiving reports in OnDemand. During the loading process, OnDemand programs process reports, create index data, and store report data and resources on cache storage volumes and archive media.

local. Pertaining to a device accessed directly without use of a telecommunication line.

local area network (LAN). (1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (2) A network in which a set of devices is connected to one another for communication and that can be connected to a larger network. See also Token-Ring Network.

logical volume. The combined space from all volumes defined to either the ADSM database or recovery log. The database resides on one logical volume and the recovery log resides on a different logical volume.

log file. A fixed-length file used to record changes to a database.

LPD. Line Printer Daemon. In TCP/IP, the command responsible for sending data from the spooling directory to a printer.

LPR. Line Printer Requestor. In TCP/IP, a client command that allows the local host to submit a file to be printed on a remote print server.

LZW. See Lempel Ziv Welsh.

M

M byte. Megabyte.

MB. Megabyte.

machine carriage control character. A character that specifies that a write, space, or skip operation should be performed either immediately or after printing the line containing the carriage control.

mainframe. A large computer, particularly one to which other computers can be connected so that they can share facilities the mainframe provides. The term usually refers to hardware only.

management class. In ADSM, a policy object that is a named collection of copy groups. A management class can contain one backup copy group, one archive copy group, a backup and archive copy group, or zero copy groups. Users can bind each file to a management class to specify how the server should manage backup versions or archive copies of files. See Copy Group.

mapping. (1) A list that establishes a correspondence between items in two groups. (2) The process of linking database fields in an application group to folder search and display fields.

megabyte (MB). When used with hard drive, diskette, or removable media storage capacity, 1,000,000 bytes. When referring to system memory capacity, 1,048,576 bytes.

memory. Program-addressable memory from which instructions and other data can be loaded directly into registers for subsequent running or processing. Memory is sometimes referred to as “storage”.

menu bar. The area at the top of a window that contains choices that give a user access to actions available in that window.

message. Information from the system that informs the user of a condition that may affect further processing of a current program.

migration. (1) The process of moving data from one computer system to another without converting the data. (2) The process of moving report files, resources, and index data from cache storage to long-term (optical or tape) storage.

mirroring. In ADSM, a feature that protects against data loss with the database or recovery log by writing the same data to multiple disks at the same time. Mirroring supports up to three exact copies of each database or recovery log.

Mixed Object Document Content Architecture - Presentation (MO:DCA-P). (1) A strategic, architected, device-independent data stream for interchanging documents. (2) A printing data stream that is a subset of the Advanced Function Presentation data stream.

MO:DCA-P. Mixed Object: Document Content Architecture for Presentation.

mount. To make a file system accessible.

mouse. A hand-held locator that a user operates by moving it on a flat surface. It allows the user to select objects and scroll the display screen by pressing buttons.

Multiple Virtual Storage (MVS). Consists of MVS/System Product Version 1 and the MVS/370 Data Facility Product operating on a System/370 processor.

MVS. Multiple virtual storage; an IBM operating system.

N

network. A collection of data processing products that are connected by communication lines for information exchange between locations.

Network File System (NFS). A protocol developed by Sun Microsystems that uses Internet Protocol to allow a set of cooperating computers to access each other’s file system as if they were local.

NFS. Network File System.

node. A workstation that operates as an OnDemand library server or object server and is connected to a TCP/IP network.

notes. Comments, clarifications, and reminders that can be attached to an OnDemand document.

non-IPDS printer. In this publication, a printer that is not channel-attached and which does not accept the Intelligent Printer Data Stream.

numeric. Pertaining to any of the digits 0 through 9.

O

object. (1) A collection of structured fields. The first structured field provides a begin-object function and the last structured field provides an end-object function. The object may contain one or more other structured fields whose content consists of one or more data elements of a particular data type. An object may be assigned a name, which may be used to reference the object. Examples of objects are text, graphics, and image objects. (2) A resource or a sequence of structured fields contained within a larger entity, such as a page segment or a composed page. (3) A collection of data referred to by a single name.

object server. A storage manager node connected to a TCP/IP network that provides cache management and, optionally, support for files archived on optical and tape storage volumes.

offset. The number of measuring units from an arbitrary starting point in a record, area, or control block to some other point.

online. Being controlled directly by or directly communicating with the computer.

operating environment. (1) The physical environment; for example, temperature, humidity, and layout. (2) All of the basic functions and the user programs that can be executed by a store controller to enable the devices in the system to perform specific operations. (3) The collection of store controller data, user programs, lists, tables, control blocks, and files that reside in a subsystem store controller and control its operation.

operating system. Software that controls the running of programs and that also can provide such services as resource allocation, scheduling, input and output control, and data management.

optical library. A disk storage device that houses optical disk drives and optical disks, and contains a mechanism for moving optical disks between a storage area and optical disk drives.

optimize. To improve the speed of a program or to reduce the use of storage during processing.

outline fonts. (1) Fonts whose graphic character shapes are defined as mathematical equations rather than by raster patterns. (2) Fonts created in the format described in *Adobe Type 1 Font Format* , a publication available from Adobe Systems, Inc. Synonymous with Type 1 fonts.

overlay. A collection of predefined, constant data such as lines, shading, text, boxes, or logos, that is electronically composed and stored as an AFP resource file that can be merged with variable data on a page while printing or viewing.

P

page. (1) A collection of data that can be printed on one side of a sheet of paper or a form. (2) The boundary for determining the limits of printing. See also Logical Page and Physical Page. (3) Part of an AFP document bracketed by a pair of Begin Page and End Page structured fields.

page definition. A resource used by OnDemand that defines the rules of transforming line data into composed pages and text controls.

page printer. A device that prints one page as a unit. (I) (A) Contrast with Line Printer.

page segment. In Advanced Function Presentation, a resource that can contain text and images and can be positioned on any addressable point on a page or an electronic overlay.

PAGEDEF. Page definition

parallel device. A device that can perform two or more concurrent activities. Contrast with Serial Device.

parameter. (1) Information that the user supplies to a panel, command, or function. (2) In the AIX operating system, a keyword-value pair.

partitioned data set. A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

path. In a network, any route between any two nodes.

path name. A name that specifies the location of a directory within a file system. Path names are used to locate and reference directories and their contents.

PC. Personal Computer.

PCL. Printer control language.

PCX. Picture Exchange Format.

PDF. Portable Document Format.

permissions. Codes that determine how the file can be used by any users who work on the system.

personal computer. A microcomputer primarily intended for stand-alone use by an individual. (T)

Picture Exchange Format (PCX). A file that contains a graphic in the PCX graphics file format, which was originally developed for the PC Paintbrush program, but is now widely used by other programs.

piobe. The printer input/output back end program used by AIX for printing tasks.

pipe. To direct the data so that the output from one process becomes the input to another process. The standard output of one command can be connected to the standard input of another with the pipe operator (¦). Two commands connected in this way constitute a pipeline.

point. (1) To move the mouse pointer to a specific object. (2) A unit of typesetting measure equal to 0.01384 inch (0.35054 mm), or about 1/72 of an inch. There are 12 points per pica.

point size. The height of a font in points. See also Point.

policy domain. A policy object that contains policy sets, management classes, and copy groups that is used by a group of client nodes. See Policy Set, Management Class, Copy Group, and Client Node.

policy set. A policy object that contains a group of management class definitions that exist for a policy domain. At any one time, there can be many policy sets within a policy domain but only one policy set can be active. See Management Class and Active Policy Set.

port. (1) A part of the system unit or remote controller to which cables for external devices (display stations, terminals, or printers) are attached. The port is an access point for data entry or exit. (2) A specific communications end point within a host. A port is identified by a port number.

Portable Document Format. A distilled version of PostScript data that adds structure and efficiency. PDF data has the same imaging model as PostScript but does not have its programmability. PDF also provides direct access to pages and allows hypertext links, bookmarks, and other navigational aids required for viewing. The text in a PDF file is usually compressed using LZW methods. The images is a PDF file are usually compressed using CCITT or JPEG methods.

PostScript. Adobe's page description language used for printing. PostScript is a very flexible programming language and imaging model but is not as structured as AFP. PostScript cannot be parsed to determine page boundaries, it must be interpreted. Because of this limitation, PostScript

is not practical for archiving and viewing. Adobe created PDF for archiving and viewing.

press. To touch a specific key on the keyboard.

primary log file. A set of one or more log files used to record changes to a database. Storage for these files is allocated in advance.

primary storage pool. A named collection of storage volumes that ADSM uses to store archive copies of files.

print file. A file that a user wants to print.

print job. A series of print files scheduled for printing. At print submission time, the user can request one or more files to be printed; therefore, a print job consists of one or more print files.

print queue. A file containing a list of the names of files waiting to be printed.

Print Services Facility (PSF). A sophisticated IBM print subsystem that drives IPDS page printers. PSF is supported under MVS, VSE, VM, OS/2, AIX, and is a standard part of the operating system under OS/400. PSF manages printer resources such as fonts, images, electronic forms, form definitions, and page definitions, and provides error recovery for print jobs.

When printing line data, PSF supports external formatting using page definitions and form definitions. This external formatting extends page printer functions such as electronic forms and use of typographic fonts without any change to applications that generate the data.

Print Services Facility/2 (PSF/2). PSF/2 is an OS/2-based print server that drives IPDS page printers, as well as IBM PPDS and HP-PCL compatible printers. PSF/2 manages printer resources and provides error recovery for print jobs. PSF/2 supports distributed printing of AFP print jobs from PSF for AIX, PSF/MVS, PSF/VSE, PSF/VM, and OS/400. PSF/2 also supports printing from a wide range of workstation applications, including Microsoft Windows and OS/2 Presentation Manager, as well as the ASCII, PostScript, and AFP data streams.

Print Services Facility for AIX (PSF for AIX). An IBM licensed program that produces printer commands from the data sent to it and it runs on the AIX/6000 operating system.

print spooler. The print spooler directs the printing of data from different applications. It temporarily stores information in separate files until they are printed.

Printer Control Language (PCL). The data stream used by Hewlett-Packard LaserJet II and III and other compatible printers.

process. An activity within the system that is started, such as a command, a shell program, or another process.

profile. (1) A file containing customized settings for a system or user. (2) Data describing the significant features of a user, program, or device.

program level. The version, release, modification, and fix levels of a program.

prompt. A displayed symbol or message that requests information or operator action.

protocol. A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication.

PSF. Print Services Facility.

PSF/2. Print Services Facility/2.

PSF for AIX. Print Services Facility for AIX.

PTF. Program temporary fix.

Q

qdaemon. The daemon process that maintains a list of outstanding jobs and sends them to the specified device at the appropriate time.

qualified name. (1) A data name explicitly accompanied by a specification of the class to which it belongs in a specified classification system. (I) (A) (2) A name that has been made unique by the addition of one or more qualifiers.

queue. (1) A line or list formed by items waiting to be processed. (2) To form or arrange in a queue.

queue device. A logical device defining characteristics of a physical device attached to a queue.

R

radio button. Round option buttons grouped in dialog boxes; one is preselected. Like a radio in an automobile, select only one button (“station”) at a time.

RAM. Random access memory. Specifically, the memory used for system memory. Sometimes this memory is referred to as main storage.

raster. In Advanced Function Presentation, an on/off pattern of electrostatic images produced by the laser print head under control of the character generator.

raster font. A font in which the characters are defined directly by the raster bit map. See Font. Contrast with Outline Font.

raster graphics. Computer graphics in which a display image is composed of an array of pixels arranged in rows and columns.

read access. In computer security, permission to read information.

record. (1) In programming languages, an aggregate that consists of data objects, possibly with different attributes, that usually have identifiers attached to them. (2) A set of data treated as a unit. (3) A collection of fields treated as a unit.

recovery log. In ADSM, a log of updates that are about to be written to the database. The log can be used to recover from system and media failures.

recovery procedure. (1) An action performed by the operator when an error message appears on the display screen. This action usually permits the program to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again.

register. Define a client node to ADSM.

remote. Pertaining to a system or device that is accessed through a communications line. Contrast with Local.

remote print. Issuing print jobs to one machine (client) to print on another machine (server) on a network.

remote system. A system that is connected to your system through a communication line.

report file. A print data stream produced by an application program that can contains hundreds or thousands of pages of related information. Some report files can be segmented into single and multiple page objects called documents.

resolution. (1) In computer graphics, a measure of the sharpness of an image, expressed as the number of lines and columns on the display screen. (2) The number of pels per unit of linear measure.

resource. A collection of printing instructions, and sometimes data to be printed, that consists entirely of structured fields. A resource can be stored as a member of a directory and can be called for by the Print Services Facility when needed. The different resources are: coded font, character set, code page, page segment, overlay, and form definition.

resource directory. A place in which resource files are stored.

resource management. The function that protects serially accessed resources from concurrent access by computing tasks.

retention. The amount of time, in days, that archived files will be retained in ADSM before they are deleted.

retry. To try the operation that caused the device error message again.

return code. (1) A value that is returned to a program to indicate the results of an operation issued by that program. (2) A code used to influence the running of succeeding instructions.

root. The user name for the system user with the most authority.

root file system. The file system that contains all the default installation and program directories in the system.

root user. In UNIX environments, an expert user who can log in and execute restricted commands, shut down the system, and edit or delete protected files.

root volume group. The volume group, identified with a single / (forward slash) that contains all the directories in the root file system.

rotation. (1) The alignment of a character with respect to its character baseline, measured in degrees in a clockwise rotation. Examples are 0°, 90°, 180°, and 270°. Zero-degree character rotation exists when a character is in its customary alignment with the baseline. Synonymous with Character Rotation. (2) The number of degrees a character is turned relative to the page coordinates. (3) The orientation of the characters of a font with respect to the baseline.

routing. The assignment of the path by which a message will reach its destination.

S

secondary log file. A set of one or more log files used to record changes to a database. Storage for these files is allocated as needed when the primary log fills up.

segment. (1) A collection of composed text and images, prepared before formatting and included in a document when it is printed. See Page Segment. (2) The resource that contains the structured-field definition of a page segment. (3) A 100 page portion of a report file. OnDemand divides report files into segments to provide enhanced performance and maintenance.

select. To pick a menu command or other object with a single click of the mouse.

serial device. A device that performs functions sequentially, such as a serial printer that prints one byte at a time. Contrast with Parallel Device.

server. (1) On a network, the computer that contains the data or provides the facilities to be accessed by other computers on the network. (2) A program that handles protocol, queuing, routing, and other tasks necessary for data transfer between devices in a computer system. (3) A workstation connected to a TCP/IP network that runs the OnDemand programs that store, retrieve, and maintain report files. OnDemand supports two types of servers: a library server an object server.

server options file. The ADSM file that specifies processing options for communication methods, tape handling, pool sizes, language, and date, time, and number formats.

shell. In UNIX environments, a software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards and pointing devices and communicate them to the operating system.

skip-to-channel control. A line printer control appearing in line data. Allows space to be left between print lines. Compatible with page printers when the data is formatted by page definitions.

SMIT. System Management Interface Tool.

SMS. System Managed Space.

software. Programs, procedures, rules, and any associated documentation pertaining to the operating of a system. Contrast with Hardware.

spool file. (1) A disk file containing output that has been saved for later printing. (2) Files used in the transmission of data among devices.

spooling (simultaneous peripheral operation online). Performing a peripheral operation such as printing while the computer is busy with other work.

spooling subsystem. A synonym for the queuing system that pertains to its use for queuing print jobs.

stand-alone workstation. A workstation that can perform tasks without being connected to other resources such as servers or host systems.

standard input. The primary source of data going into a command. Standard input comes from the keyboard unless redirection or piping is used, in which case standard input can be from a file or the output from another command.

standard output. The primary destination of data coming from a command. Standard output goes to the display unless redirection or piping is used, in which case standard output can be to a file or another command.

status. (1) The current condition or state of a program or device. For example, the status of a printer. (2) The condition of the hardware or software, usually represented in a status code.

storage. (1) The location of saved information. (2) In contrast to memory, the saving of information on physical devices such as disk or tape.

storage device. A functional unit for storing and retrieving data.

storage hierarchy. A logical ordering of storage devices. Generally, the ordering is based on the speed and capacity of the devices.

storage node. A named object that identifies an ADSM domain on an OnDemand object server. There are two types of storage nodes: a primary storage node and a secondary storage node. Application group data is archived in a primary storage node. A secondary storage node contains a backup copy of data stored in a primary node.

storage object. A portion of a storage volume managed as a single entity. A storage object can contain many report file segments.

storage pool. A named collection of storage volumes that is the destination for archived files.

storage pool volume. A volume that has been assigned to a storage pool to store archived files.

storage set. A named collection of storage nodes with the same storage management characteristics. For example, the life of the data in OnDemand.

storage volume. A disk volume that has been assigned to store reports and documents on the OnDemand server.

string. A series or set of alphabetic or numeric characters. A string can be composed of letters, numbers, and special characters.

structure. A variable that contains an ordered group of data objects. Unlike an array, the data objects within a structure can have varied data types.

structured field. (1) A self-identifying, variable-length, bounded record that can have a content portion that provides control information, data, or both. (2) A mechanism that permits variable length data to be encoded for transmission in the data stream. See Field.

subdirectory. In the file system hierarchy, a directory contained within another directory.

subroutine. (1) A sequenced set of statements or coded instructions that can be used in one or more computer programs and at one or more points in a computer program. (2) A routine that can be part of another routine.

syntax. The grammatical rules for constructing a command, statement, or program.

syntax diagram. A diagram for a command that displays how to enter the command on the command line.

system console. A console, usually equipped with a keyboard and display screen, that is used by an operator to control and communicate with a system. Synonymous with Console.

system customization. Specifying the devices, programs, and users for a particular data processing system. See also Configuration.

system integrity. In computer security, the quality of a system that can perform its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

System Managed Space (SMS). A type of DB2 table space. An SMS table space is managed by the filesystem manager.

system management. The tasks involved in maintaining the system in good working order and modifying the system to meet changing requirements.

System Management Interface Tool (SMIT). In the AIX operating system, a series of panels that allow you to perform system functions without directly issuing any commands.

system memory. Synonymous with Main Storage, but used in hardware to refer to semiconductor memory (modules).

system prompt. Synonym for command line. The system prompt is the symbol that appears at the command line of an operating system. The system prompt indicates that the operating system is ready for the user to enter a command.

T

table. A named collection of data consisting of rows and columns.

table reference character (TRC). (1) Usually, the second byte on a line in the user's data. This byte contains a value (0–126) that is used to select a font to be used to print that line. (2) In the 3800 Printing Subsystem, a numeric character (0, 1, 2, or 3) corresponding to the order in which the character arrangement table names have been specified with the **CHARS** keyword. It is used for selection of a character arrangement table during printing.

table space. An abstraction of a collection of containers into which database objects are stored. A table space provides a level of indirection between a database and the tables stored within the database. A table space:

- Has space on media storage devices assigned to it.
- Has tables created within it.

tag. (1) 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. (2) In text formatting markup language, a name for a type of document element that is entered in the source document to identify it.

Tagged Image File Format (TIFF). A bit-mapped graphics format for scanned images with resolutions of up to 300 dpi. TIFF simulates gray scale shading.

TB. Terabyte.

TCP. Transmission Control Protocol.

TCP/IP. Transmission Control Protocol/Internet Protocol.

terabyte. A unit of memory or space measurement capacity equal to approximately one trillion bytes. One terabyte is equal to 1,000 gigabytes, or one million megabytes.

text. (1) A type of data consisting of a set of linguistic characters (letters, numbers, and symbols) and formatting controls. (2) In word processing, information intended for human viewing that is presented in a two-dimensional form, such as data printed on paper or displayed on a screen.

throughput. A measure of the amount of work performed by a computer system over a period of time, for example, the number of jobs per day. (I)

TIFF. Tagged Image File Format.

token name. An eight-byte name that can be given to all data stream objects.

token-ring network. A ring network that allows unidirectional data transmission between data stations, by a token passing procedure, such that the transmitted data return to the transmitting station. (T)

toolbar. The region directly beneath the menu bar of the main window in OnDemand client programs that support a graphical user interface.

toolbar button. A small bitmap on the toolbar that represents a command in OnDemand client programs that support a graphical user interface. Click a toolbar button to quickly access a command.

trigger. Data values that ACIF searches for in a print data stream, to delineate the beginning of a new group of pages. The first trigger is then the anchor point ACIF uses to locate index values.

transfer. To send data to one place and to receive data at another place.

transform. To change the form of data according to specified rules without significantly changing the meaning of the data. (I) (A)

Transmission Control Protocol (TCP). A communications protocol used in Internet and in any network that follows the U.S. Department of Defense standards for inter-network protocol. TCP provides a host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the Internet protocol is the underlying protocol.

Transmission Control Protocol/Internet Protocol (TCP/IP). A set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

TRC. Table reference character.

type. To enter specific information using the keyboard, typing characters exactly as given.

U

unformatted print data. Data that is not formatted for printing. A page definition can contain controls that map unformatted print data to its output format.

UNIX operating system. An operating system developed by Bell Laboratories that features multiprogramming in a multi-user environment. The UNIX operating system was originally developed for use on minicomputers but has been adapted for mainframes and microcomputers.

upload. To transfer data from one computer to another. Typically, users upload from a small computer to a large one.

user. A person authorized to logon to an OnDemand server.

user exit. (1) A point in an IBM-supplied program at which a user exit routine may be given control. (2) A programming service provided by an IBM software product that may be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

user interface. The hardware, software, or both that implements a user interface, allowing the user to interact with and perform operations on a system, program, or device. Examples are a keyboard, mouse, command language, or windowing subsystem.

V

value. (1) A set of characters or a quantity associated with a parameter or name. (2) A quantity assigned to a constant, variable, parameter, or symbol.

variable. (1) A name used to represent a data item whose value can change while the program is running. (2) In programming languages, a language object that can take different values at

different times. (3) A quantity that can assume any of a given set of values.

version number. The version level of a program, which is an indicator of the hardware and basic operating system upon which the program operates. The version, release, modification, and fix levels together comprise the program level or version of a program.

virtual printer. A view of a printer that refers only to the high-level data stream, such as ASCII or PostScript, that the printer understands. It does not include any information about how the printer hardware is attached to the host computer or the protocol used for transferring data to and from the printer.

volume. The basic unit of storage for a database, log file, or a storage pool. A volume can be an LVM logical volume, a standard file system file, a tape cartridge, or an optical platter. Each volume is identified by a unique volume identifier.

W

wildcard. Search characters that represent other letters, numbers, or special characters. In OnDemand, the %(percentage) and the _(underscore) are wildcard characters.

window. A part of a display screen with visible boundaries in which information is presented.

workstation. A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

write access. In computer security, permission to write to an object.

writer. A JES function that processes print output.

Index

A

B

C

D

E

N

O

P

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

EDMSuite OnDemand
Indexing Reference
Version 2.2

Publication No. S544-5489-05

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. _____

Readers' Comments — We'd Like to Hear from You
S544-5489-05

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 Software Solutions
Department MYGA Building 001L
PO Box 1900
Boulder, CO 80301-9817

Fold and Tape **Please do not staple** Fold and Tape

S544-5489-05

Cut or Fold
Along Line

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

S544-5489-05

Spine information:

EDMSuite OnDemand Indexing Reference Version 2.2