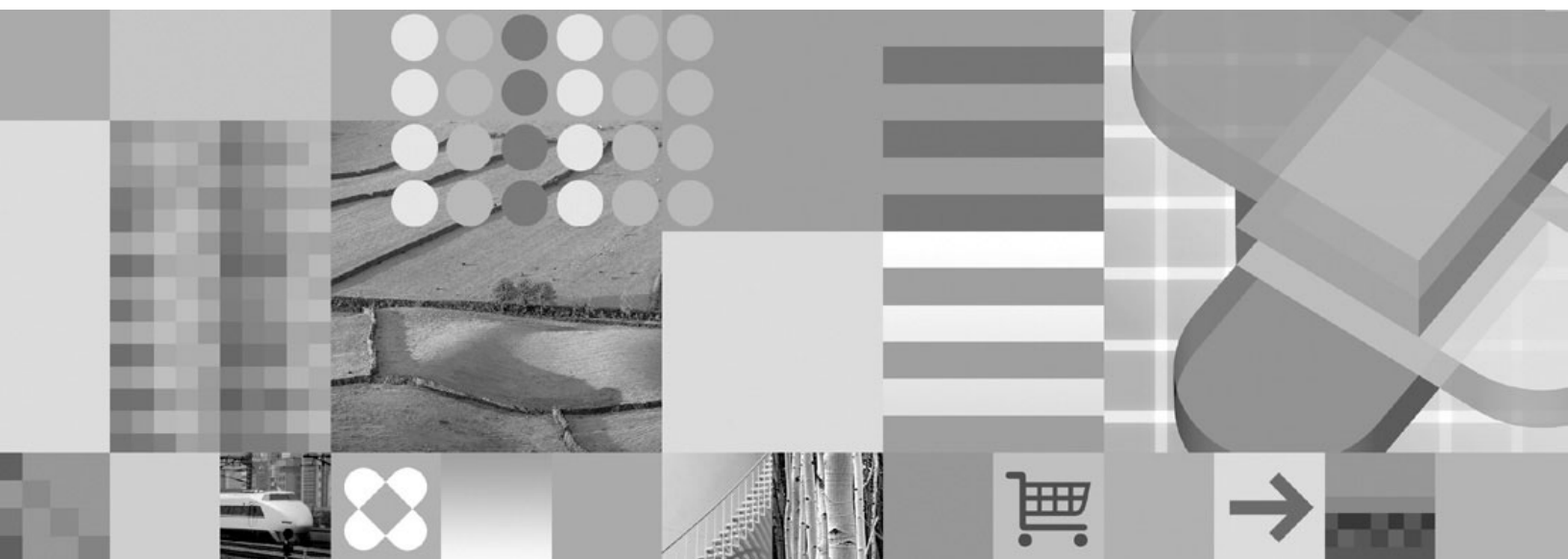


Version 7.1



Indexing Reference

Version 7.1



Indexing Reference

Note

Before using this information and the product it supports, read the information in “Notices” on page 239.

Fifth Edition (September 2004)

This edition replaces SC27-1375-03.

This edition applies to Version 7 Release 1 of IBM DB2 Content Manager OnDemand for z/OS and OS/390 (product number 5655-H39) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2001, 2004. All rights reserved.

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

Contents

About this guide. ix

Who should use this guide ix

How this guide is organized ix

Where to find more information ix

Accessibility information for OnDemand for z/OS
and OS/390 x

Information about other products x

Support available on the Web xi

Education and training. xi

How to send your comments xi

| Summary of changes xiii

Part 1. ACIF reference. 1

Chapter 1. Overview 3

About ACIF 4

Indexing concepts 4

Indexing parameters. 5

Converting line data to AFP 6

AFP data 7

Mixed Object Document Content Architecture

Data 7

Line data 7

Mixed-mode data. 7

AFP resources 8

How OnDemand uses index information 9

ACIF parameters for EBCDIC data 10

Creating indexing parameters 10

Specifying indexing parameters. 10

Chapter 2. Using ACIF 13

Example one 13

About the report. 13

Key concepts 15

Defining the application, part 1. 16

Opening the report 16

Defining triggers 17

Defining fields 18

Defining indexes 19

Displaying triggers, fields, and indexes 19

Indexer Properties 20

Defining the application, part 2. 20

Example two 21

About the report. 21

Key concepts 24

Defining the application, part 1. 25

Opening the report 26

Defining triggers 26

Defining fields 27

Defining indexes 28

Displaying triggers, fields, and indexes 30

Indexer Properties 30

Defining the application, part 2. 31

Example three 31

About the report. 31

Key concepts 33

Defining the application, part 1. 34

Opening the report 35

Defining triggers 35

Defining fields 38

Defining indexes 40

Displaying triggers, fields, and indexes 42

Indexer Properties 42

Defining the application, part 2. 43

Example four. 44

About the report. 44

Key concepts 46

Defining the application, part 1. 46

Creating indexing parameters 47

Defining the data format 47

Defining indexing information 47

Defining resource information 47

Defining the application, part 2. 48

Chapter 3. ACIF parameter reference 49

CC 49

Syntax 49

Options and values. 49

Related parameters. 49

CCTYPE 49

Syntax 50

Options and values. 50

Related parameters. 50

CHARS. 50

Syntax 51

Options and values. 51

Related parameters. 51

CONVERT. 51

Syntax 52

Options and values. 52

Related parameters. 52

CPGID 52

Syntax 53

Options and values. 53

DCFPAGENAMES 53

Syntax 53

Options and values. 53

EXTENSIONS 53

Syntax 54

Options and values. 54

Related parameters. 54

FDEFLIB 54

Related parameters. 55

FIELD 55

Trigger field syntax. 55

Constant field syntax 57

Transaction field syntax 58

Related parameters. 60

FILEFORMAT	60	NEWPAGE	74
Syntax	60	Syntax	74
Options and values.	60	Options and values.	74
Related parameters.	61	Related parameters.	74
FONTLIB	61	OUTEXIT	74
Related parameters.	61	Syntax	75
FORMDEF.	61	Options and values.	75
Syntax	62	OUTPUTDD	75
Options and values.	63	Syntax	75
Related parameters.	63	Options and values.	75
GROUPMAXPAGES	63	OVLYLIB	75
Syntax	63	Syntax	76
Options and values.	63	Options and values.	76
Related parameters.	63	Related parameters.	76
GROUPNAME	64	PAGEDEF	76
Syntax	64	Syntax	77
Options and values.	64	Options and values.	77
Related parameters.	64	Related parameters.	78
IMAGEOUT	64	PARMDD	78
Syntax	64	Syntax	78
Options and values.	65	Options and values.	78
Related parameters.	65	PDEFLIB	78
INDEX	65	Syntax	79
Syntax	65	Options and values.	79
Options and values.	65	Related parameters.	79
Examples	67	PRMODE	79
Related parameters.	68	Syntax	79
INDEXDD.	68	Options and values.	80
Syntax	69	Related parameters.	80
Options and values.	69	PSEGLIB	80
INDEXOBJ	69	Syntax	81
Syntax	69	Options and values.	81
Options and values.	69	Related parameters.	81
Related parameters.	69	RESEXIT	81
INDEXSTARTBY.	70	Syntax	81
Syntax	70	Options and values.	81
Options and values.	70	RESFILE	81
INDXEXIT.	70	Syntax	82
Syntax	70	Options and values.	82
Options and values.	71	Related parameters.	82
INPEXIT	71	RESOBJDD	82
Syntax	71	Syntax	83
Options and values.	71	Options and values.	83
INPUTDD.	71	RESTYPE	83
Syntax	71	Syntax	83
Options and values.	71	Options and values.	83
INSERTIMM	72	Related parameters.	85
Syntax	72	TRACE	85
Options and values.	72	Syntax	85
Related parameters.	72	Options and values.	85
LINECNT	72	TRC	85
Syntax	73	Syntax	86
Options and values.	73	Options and values.	86
Related parameters.	73	Related parameters.	86
MCF2REF	73	TRIGGER	86
Syntax	73	Syntax	87
Options and values.	73	Options and values.	87
Related parameters.	73	Notes	88
MSGDD	73	Examples	88
Syntax	74	Related parameters.	89
Options and values.	74	UNIQUEBNGS	89

Syntax	89
Options and values.	89
Related parameters.	89
USERLIB	89
Syntax	90
Options and values.	90
USERMASK	90
Syntax	90
Options and values.	90
Examples	91
Related parameters.	91

Chapter 4. Messages. 93

Chapter 5. User exits and attributes of the input file 95

User programming exits	95
Input record exit.	95
Index record exit	97
Output record exit	98
Resource exit	100
User exit search order	102
Non-zero return codes	102
Attributes of the input file	102

Chapter 6. Hints and tips 105

Working with control statements that contain numbered lines.	105
Understanding how ACIF processes unbounded box fonts (3800)	105
Placing TLEs in named groups	106
About ANSI and machine carriage controls	106
Common methods of transferring files	107
Physical media	107
PC file transfer program.	108
FTP	108
Download for z/OS feature	108
Other considerations for transferring files	109
Invoke Medium Map (IMM) structured field	109
Indexing considerations	109
Concatenating the resource group to the document	110
Concatenating resources to an AFP file	111
Running ACIF with inline resources	111
Writing inline resources to the output file	111
Specifying the IMAGEOUT parameter	112

Chapter 7. ACIF data stream information 113

Tag Logical Element (TLE) structured field	113
Understanding how ACIF processes fully composed AFP files	114
Format of the resources file.	115
Begin Resource Group (BRG) structured field.	116
Begin Resource (BR) structured field.	116
End Resource (ER) and End Resource Group (ERG) structured fields	116

Chapter 8. Format of the ACIF index object file 117

Group-level Index Element (IEL) structured field	117
Page-level Index Element (IEL) structured field	118
Begin Document Index (BDI) structured field.	118
Index Element (IEL) structured field.	119
Tag Logical Element (TLE) structured field	119
End Document Index (EDI) structured field	120

Chapter 9. Format of the ACIF output document file 121

Page groups.	122
Begin Document (BDT) structured field.	123
Begin Named Group (BNG) structured field	123
Tag Logical Element (TLE) structured field	124
Begin Page (BPG) structured field	124
End Named Group (ENG), End Document (EDT), and End Page (EPG) structured fields	124
Output MO:DCA-P data stream	124
Composed Text Control (CTC) structured field	124
Map Coded Font (MCF) Format 1 structured field	124
Map Coded Font (MCF) Format 2 structured field	125
Presentation Text Data Descriptor (PTD) Format 1 structured field	125
Inline resources.	125
Page definitions	125

Chapter 10. Using ACIF in z/OS 127

Sample JCL	127
About the JCL statements	127
ACIF parameters	129
Syntax Rules	129
JCL and ACIF parameters	129
z/OS libraries	131
ACIF output.	131
Concatenating files	131

Part 2. Generic indexer reference 133

Chapter 11. Overview 135

Processing AFP data	135
-------------------------------	-----

Chapter 12. Specifying parameters 137

CODEPAGE:	137
Syntax.	137
Options and values	137
Example	138
COMMENT:.	138
Syntax.	138
Options and values	138
Example	138
GROUP_FIELD_NAME:.	138
Syntax.	138
Options and values	138
Example	139
GROUP_FIELD_VALUE:	139
Syntax.	139
Options and values	139
Example	139

GROUP_FILENAME:	139
Syntax.	140
Options and values	140
Example	140
GROUP_LENGTH:	141
Syntax.	141
Options and values	141
Example	141
GROUP_OFFSET:	141
Syntax.	141
Options and values	141
Example	142
Chapter 13. Parameter examples	143
Part 3. PDF indexer reference	145
Chapter 14. Overview	147
What is the PDF indexer?	147
How OnDemand uses index information	149
Indexing input data	150
Indexing concepts	150
Coordinate system.	150
Indexing parameters	150
How to create indexing parameters	153
Processing PDF input files with the graphical indexer	153
Troubleshooting	157
Chapter 15. System considerations	159
System requirements	159
System limitations.	159
Input data requirements.	159
NLS considerations	160
Chapter 16. Parameter reference	161
COORDINATES	161
Syntax.	161
Options and values	161
FIELD.	161
Trigger field syntax	161
Constant field syntax	163
Related parameters	164
INDEX	164
Syntax.	164
Options and values	164
Examples.	165
Related parameters	165
INDEXDD	165
Syntax.	165
Options and values	165
INDEXSTARTBY	166
Syntax.	166
Options and values	166
INPUTDD	167
Syntax.	167
Options and values	167
MSGDD	167
Syntax.	167

Options and values	167
OUTPUTDD.	167
Syntax.	168
Options and values	168
PARMDD	168
Syntax.	168
Options and values	168
TRACEDD parameter	168
TRIGGER	168
Syntax.	169
Options and values	169
Examples.	169
Related parameters	170

Chapter 17. Message reference. . . . 171

Chapter 18. Using ARSPDOCI in z/OS 173

Sample JCL	173
About the JCL statements	173
Sample parameter file	175
About the parameters	175

Chapter 19. Using ARSPDUMP in z/OS 177

Sample JCL	177
About the JCL statements	177

Chapter 20. Trace facility 179

Part 4. Xenos transform. 181

Chapter 21. Understanding Xenos . . 183

Chapter 22. Xenos transform. 185

Convert data streams.	185
AFP documents to HTML	185
AFP data to PDF	186
AFP documents to XML.	186
Metacode to AFP	186
Metacode documents to HTML	187
Metacode to Metacode	187
Metacode to PDF	187
Metacode documents to XML	188
PCL to PDF	188
Index documents	188
Indexing with data values	189
Collect resources	189
Summary.	190

Chapter 23. Loading data 191

Chapter 24. Scenarios for using Xenos 193

Viewing and printing with the Windows client	194
Viewing and printing with the OnDemand Web Enablement Kit.	195

Chapter 25. How to specify parameters to the Xenos transform . . 197

Processing sample data	197
Specifying parameters to OnDemand	198

Chapter 26. Using JSMAIN in z/OS 201

Generating the locations of text strings	201
Sample JCL	202
About the JCL statements	203
Sample parameter file	204
About the parameters	205
Sample script file	205
Indexing and converting input data	207
Sample JCL	208
About the JCL statements	208
Sample parameter file	209
About the parameters	210
Sample script file	211
Generic index file	214

Part 5. OS/390 indexer reference 217

Chapter 27. Overview 219

Chapter 28. Indexing concepts 221

Chapter 29. OS/390 indexer

parameters 223

ACIF parameters used with the OS/390 indexer	223
TRIGGER	223
INDEX	223
FIELD	223
GROUPMAXPAGES	223
FILEFORMAT	224
INDEXOBJ	224
Other Parameters used with the OS/390 indexer	224
INPEXIT	224
INDEXEXIT	226
ANYEXIT	227
INDEXSTYLE	229

Chapter 30. Using the OS/390 indexer 237

The OnDemand application	237
The ARSLOAD program.	237

Notices 239

Trademarks	241
----------------------	-----

Index 243

About this guide

This reference guide contains information about indexing methods, preparing index data, and using tools to index reports that you plan to store in and retrieve from IBM® DB2™ Content Manager OnDemand for z/OS™ and OS/390®, Version 7.1 (OnDemand).

Who should use this guide

This reference guide is of primary interest to administrators and other people in an organization who are responsible for preparing data to be stored in OnDemand.

How this guide is organized

This guide is organized into the following parts. Each part contains information about one of the indexing tools that you can use to index data to be stored in the system.

- Part 1, “ACIF reference,” on page 1 provides information about indexing Advanced Function Printing™ (AFP™) data and line data with AFP Conversion and Indexing Facility (ACIF)
- Part 2, “Generic indexer reference,” on page 133 describes how to use the OnDemand Generic indexer to specify index data for other types of input data
- Part 3, “PDF indexer reference,” on page 145 describes how to use the OnDemand PDF indexer to generate index data for Adobe PDF files
- Part 4, “Xenos transform,” on page 181 describes how to use the Xenos transforms to process AFP and Metacode data
- Part 5, “OS/390 indexer reference,” on page 217 provides information about the OnDemand OS/390 indexer, which you can use to index line data and AFP reports

Where to find more information

Your product package includes a complete set of information to help you plan for, install, administer, and use your system. Product documentation is provided in both BookManager® and portable document format (PDF). You can view the PDF files online using the Adobe Acrobat Reader for your operating system. If you do not have the Acrobat Reader installed, you can download it from the Adobe Web site at www.adobe.com.

Product documentation is also available from the OnDemand Web site (www.ibm.com/software/data/ondemand/390) and the IBM Publication Ordering System (www.ibm.com/shop/publications/order).

Table 1 lists the OnDemand for z/OS and OS/390 Version 7 publications.

Table 1. Version 7 publications

Title	Order number
<i>Configuration Guide</i>	GC27-1373
<i>Administration Guide</i>	SC27-1374
<i>Indexing Reference</i>	SC27-1375

Table 1. Version 7 publications (continued)

Title	Order number
<i>Web Enablement Kit Implementation Guide</i>	SC27-1376
<i>OnDemand Distribution Facility Installation and Reference Guide</i>	SC27-1377
<i>Messages and Codes</i>	SC27-1379
<i>Introduction and Planning Guide</i>	GC27-1438
<i>User's Guide</i>	SC27-0836
<i>Windows® Client Customization Guide and Reference</i>	SC27-0837
<i>Migration Guide</i>	LY37-3746
Note: The IBM DB2 Content Manager OnDemand for z/OS: <i>Migration Guide</i> , is a licensed publication. It is available only for customers who are migrating from OnDemand for OS/390 Version 2.1 to OnDemand for z/OS and OS/390, Version 7.1. See your local IBM representative to obtain this publication.	

The IBM DB2 Content Manager OnDemand for z/OS: *Introduction and Planning Guide* contains a glossary of terms for the OnDemand library. The IBM DB2 Content Manager OnDemand: *User's Guide* has a smaller glossary selected for OnDemand users rather than administrators. The IBM DB2 Content Manager OnDemand for z/OS: *OnDemand Distribution Facility Installation and Reference Guide* includes a glossary of terms that are specific to the OnDemand Distribution Facility.

Accessibility information for OnDemand for z/OS and OS/390

For complete information about accessibility features that are supported by this product, see the IBM DB2 Content Manager OnDemand for z/OS: *Administration Guide*.

Information about other products

You can find current information about z/OS and OS/390 from these Web sites:

OS/390

www.ibm.com/servers/s390/os390

z/OS

www.ibm.com/servers/eserver/zseries/zos

From the above Web sites, follow the links to access the z/OS and OS/390 manuals in BookManager READ and Adobe PDF formats. The manuals are also available in Collections on CD-ROM that can be ordered from the IBM Publications Center on the Web. Use the following links to find current information about these products:

CICS® www.ibm.com/software/ts/cics/

DB2 Universal Database™

www.ibm.com/software/data/db2/

TCP/IP

To find current information about TCP/IP support, see the z/OS Web site at www.ibm.com/servers/eserver/zseries/zos or the OS/390 Web site at www.ibm.com/servers/s390/os390.

USS (UNIX® System Services)

www.ibm.com/servers/eserver/zseries/zos/unix/

OAM (Object Access Method) and VSAM (Virtual Storage Access Method) are documented under DFSMS/MVS[®] system management software. DFSMS/MVS product information is available from the z/OS and OS/390 Web sites listed above. For OAM, see *z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC35-0426.

Support available on the Web

IBM provides updated product information online. Follow the Support link at www.ibm.com/software/data/ondemand/390/ for frequently asked questions, hints and tips, and technical notes.

Education and training

IBM offers several classes for OnDemand administrators. Follow the Training and certification link at www.ibm.com/software/data/ondemand/ for course descriptions and prices.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this publication or other OnDemand documentation. You can use either of the following methods to provide comments:

- Send your comments from the Web. Visit the IBM Data Management Online Reader's Comment Form (RCF) page at: www.ibm.com/software/data/rcf
- Send your comments by e-mail to: ondemand@us.ibm.com

Be sure to include the name of the product, the version number of the product, and the name of the book. If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

Summary of changes

Summary of changes for IBM DB2 Content Manager OnDemand for z/OS and OS/390, Version 7.1: Indexing Reference, SC27-1375-04

This publication contains additions and changes to information previously presented in *IBM DB2 Content Manager OnDemand for z/OS and OS/390, Version 7.1: Indexing Reference*, SC27-1375-03. The technical additions and changes are marked with a revision bar (|) in the left margin.

The following information is new or updated:

- Enhanced the procedure in the section “Processing PDF input files with the graphical indexer” on page 153, and added a subsection called “Troubleshooting” on page 157.
- Added a note at the end of Chapter 21, “Understanding Xenos,” on page 183.
- Added ARSEXINP-DDNAME and ANY-DDNAME parameters for use with the OS/390 indexer. See “Other Parameters used with the OS/390 indexer” on page 224.

Part 1. ACIF reference

This part of the book provides information about ACIF. You can use ACIF to extract index data from and generate index data about data sets that contain line data. You can also use ACIF to convert line data to AFP and collect the resources required to view and reprint the AFP data.

Note for Infoprint® and PSF customers: The OnDemand product includes an enhanced version of the ACIF software. The enhanced version of the ACIF software provides additional function that is not available with the version of ACIF that is included with Infoprint and PSF. IBM recommends that you process your reports with the enhanced version of ACIF that is provided with OnDemand.

Chapter 1. Overview

Note: Statements in this chapter about ACIF in OS/390 apply equally to ACIF in the z/OS environment.

ACIF is a powerful tool for indexing the print data streams of OS/390 application programs. ACIF indexes reports based on the organization of the data in the report. You can optionally convert line data print streams into AFP data. ACIF processes three input sources:

- Indexing parameters that specify how the data should be indexed. You can create the indexing parameters when you define an OnDemand application.
- AFP resources required to view and print the data, if the data was created by an AFP application.
- The print data stream.

The output of ACIF is either a fully composed AFP data stream or the original line data input. ACIF can convert line data input to AFP data, can produce an index file that OnDemand uses to create index data for the database, and optionally, can collect resources into a resource group file.

ACIF produces a resource group file for AFP data. To create a resource group file, ACIF must have access to the resources required by the input data stream. OnDemand usually stores the resources in cache storage and retrieves the resources associated with a specific document when a user selects the document for viewing.

ACIF indexes input data based on the organization of the data:

- Document organization. For reports made up of logical items, such as statements, policies, and invoices. ACIF can generate index data for each logical item in the report.
- Report organization. For reports that contain line data with sorted values on each page, such as a transaction log or general ledger. ACIF can divide the report into groups of pages and generate index data for each group of pages.

Before you can index a report with ACIF, you need to create a set of *indexing parameters*. The indexing parameters describe the physical characteristics of the input data, identify where in the data stream that ACIF can locate index data, and provide other directives to ACIF. Collecting the information needed to develop the indexing parameters requires several steps. For example:

1. Examine the input data to determine how users use the report, including what information they need to retrieve a report from the system (indexing requirements).
2. For line data, decide whether or not to convert the input data to AFP.
 - 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, then you must convert the source data to AFP.
 - If you need to generate *page-level* index information for a line data input file, then you must convert the input data to AFP. Page-level indexes can be used to move to specific pages in a document. The page-level index information is not stored in the database and, therefore, cannot be used to search for and retrieve documents.

3. Examine the input data to determine the 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 to AFP.
6. Create parameters for collecting resources for viewing and printing AFP data.

You can run ACIF as part of the OnDemand application or as a stand-alone job prior to running the OnDemand load process. The information provided in this book assumes that you will run ACIF as part of the OnDemand application. The OnDemand application retrieves the indexing parameters from the OnDemand database and uses the parameters to process the input data.

Note for Infoprint and PSF customers: The OnDemand product includes an enhanced version of the ACIF software. The enhanced version of the ACIF software provides additional function that is not available with the version of ACIF that is included with Infoprint and PSF. IBM recommends that you process your reports with the enhanced version of ACIF that is provided with OnDemand.

Important: When loading data using ACIF, the locale must be set appropriately for the CPGID parameter. For example, if CPGID=273 is specified, set the LC_ALL environment variable to De_DE.IBM-273 or some other locale that correctly identifies upper and lower case characters in code page 273.

About ACIF

ACIF is a batch utility that provides three major functions:

- Sophisticated indexing functions.
ACIF can logically divide reports 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. 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 users to view the report 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

Indexing parameters can contain indexing, conversion, and resource collection parameters, options, and values. For most reports, ACIF requires three indexing parameters to extract or 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 illustrates a portion of a page from a sample report.

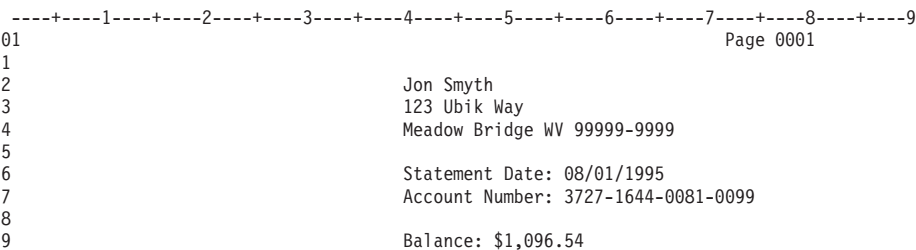


Figure 1. Indexing a report

The following indexing parameters could be used to generate index data for the report shown 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 line 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

AFP data

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

- Graphics Object Content Architecture (GOCA)
- Presentation Text Object Content Architecture (PTOCA)
- Image Object Content Architecture (IOCA)
- Bar Code Object Content Architecture (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 *Data Stream and Object Architectures 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 a hexadecimal 5A (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 *Data Stream and Object Architectures 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*.

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. ACIF collects the resources and places them in a resource file. OnDemand loads the resource file at the same time it loads the indexed reports.

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 2 lists typical values for the RESTYPE parameter.

Table 2. Collecting Resources

RESTYPE=	Function	Purpose
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.
- If a resource is inline and ACIF is collecting that type of resource, the resource will be saved in the resource file regardless of whether it is used in the document, unless EXTENSIONS=RESORDER is specified in the ACIF parameters. Another method to remove unwanted resources from the resource file is to use a resource exit.

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 load a report into OnDemand, the data loading program invokes ACIF to process the indexing parameters and extract index data from the report. The data loading program then updates the database with the index data, storing the group-level attribute values that ACIF extracted from the report into database fields. Figure 2 shows an overview of the index creation and loading process.

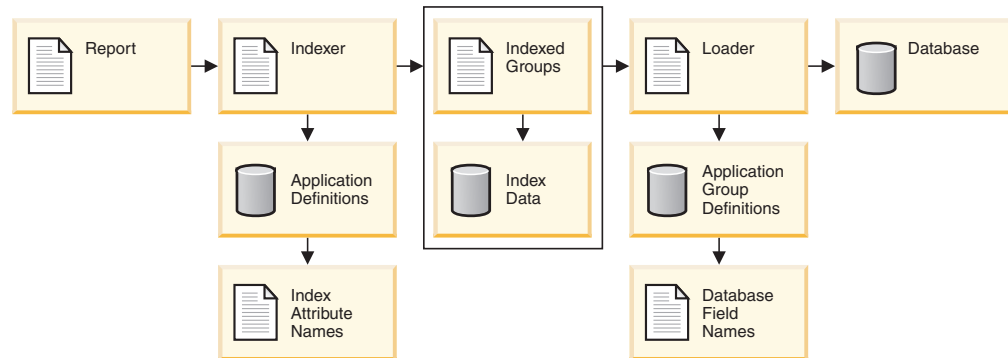


Figure 2. Indexing reports

You typically create an OnDemand application for each report that you plan to store in OnDemand. The application contains 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 data, 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 one of the application group database fields.

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 use ACIF to index a report, ACIF creates index data that contains the index field names and the 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 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 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 storage. If the item is an AFP document or a line data document stored in large objects and page-level indexes were generated when the report was processed by ACIF, the user can move to pages of the item using the page-level index information.

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 move to a specific page in the document.

ACIF parameters for EBCDIC data

Most reports are created on an OS/390 system in EBCDIC format. This topic describes the index parameters, options, and data values that can be used to process a report that contains EBCDIC data.

Creating indexing parameters

You can use the following methods to create indexing parameters:

- Process a sample report with the graphical indexer
- Import a parameter file into the application
- Type indexing parameters directly into the application

If you plan to index reports outside of the OnDemand load process, then you should define the indexing parameters in an indexing data set that is accessible to the ACIF program.

Specifying indexing parameters

Assume that a 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 character '1' in the first byte, ACIF uses the second trigger to look for the character string 'Page 0001' starting in column 72 of the same input record. When this condition is found, a new statement exists, and the record containing a character '1' in the first byte is considered the anchor record. ACIF uses the anchor record to locate index values. The trigger specifications are expressed as follows:

```
trigger1=*,1,'1'           /* Skip to Channel 1 */
trigger2=0,72,'Page 0001'
```

Figure 3. Indexing EBCDIC data (part 1 of 4)

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 character string '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='custnam',field1 /* index/db field = custnam */
```

Figure 4. Indexing EBCDIC data (part 2 of 4)

To create the indexing tag for the statement date attribute, define the character string '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='sdate',field2    /* index/db field = sdate */
```

Figure 5. Indexing EBCDIC data (part 3 of 4)

To create the indexing tag for the account number attribute, define the character string '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='acctnum',field3 /* index/db field = acctnum */
```

Figure 6. Indexing EBCDIC data (part 4 of 4)

After indexing the report, OnDemand stores the index values in the database for each of the three indexing attributes for each statement in the input 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 7 on page 14 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 system. 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 8 on page 15 shows the indexer parameters required for ACIF to process the loan report.

Accessing report data

This example provides instructions for 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 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 OS/390 system. We transferred the report to the PC as a binary file. We can then load the sample data into the graphical indexer.

It is important that the sample data that you use to create the indexing information matches the actual data to be indexed and loaded into the system. When you load a report into the system, OnDemand uses the indexing parameters, options, and data values from the OnDemand application 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, then OnDemand may not index the data properly. For example, OnDemand may not be able to locate triggers, indexes, and fields or extract the correct index values.

REPORT D94100100
BANK 001
FROM 10/01/94
TO 10/01/94

PENNANT NATIONAL BANK

LOAN DELINQUENCY REPORT

DATE 10/01/94
TIME 16:03:46
MODE 9
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 7. 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,'1',(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='load_date',FIELD1,(TYPE=GROUP,BREAK=YES) /* report date index        */
INDEX2='loan_number',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 8. 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 fields for the report date, beginning load number, and ending loan number.

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=N0. 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 report

Open the Add an Application window to the Indexer Information page. In the Parameter Source area, select Sample Data. Click Modify to open the Open dialog box. Select the name of the file that contains the sample data. Click Open.

OnDemand opens the Indexer Properties dialog box. After you click OK or Cancel, OnDemand loads the input 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 from the OnDemand application to index the data. If the data being loaded does not match the data that you used to generate indexing parameters with the graphical indexer, then OnDemand may not properly index the data. For example, OnDemand may 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 ('1'). This is the only trigger that 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 to open the Add a Trigger dialog box.

The Identifier (Trigger1) determines the name of the trigger parameter. Trigger1 must always be defined and establishes the starting point at which the 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.

Note: The graphical indexer specifies literals as hexadecimal strings. This is required when running ACIF on platforms other than OS/390 or z/OS. Since we will be running ACIF on the OS/390 or z/OS platform, expressing the literals as character strings is acceptable. After you have finished using the graphical indexer, you can manually change the hexadecimal strings to character strings, if you wish.

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

ACIF parameters

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You 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 Indexer Parameters dialog box.

Defining fields

ACIF uses fields 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 to open 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 to open 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

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You can see the result of defining the fields. Note the field parameters with the options and data values that we specified in the Add a Field/Transaction Field dialog boxes.

Close the Display Indexer Parameters dialog box.

Defining indexes

Our next task in defining indexing parameters for the loan report is to define indexes. The indexes 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 to open 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 then click Add>>. OnDemand moves Field1 from the Fields list to the Order list. Click Add to add the index.

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 Add to add the index.

Click Done to close the Add an Index dialog box.

ACIF parameters

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You can see the result of defining the indexes. Note the index parameters with the options and data values that we specified in the Add an Index dialog box.

Close the Display Indexer 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 to open the Select dialog box. 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 to open the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties to open the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

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 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 that you make on the View Information page, and the triggers, fields, and indexes that you define. 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 49 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 to 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 Indexer Properties dialog box.

Load Information

The Load Information page is where we map the index attribute name that we defined to hold loan number attribute values to the 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 9 on page 22 shows an example of a statement viewed with one of the OnDemand client programs. Figure 10 on page 23 shows what the input data looks like viewed with an ISPF browser on the OS/390 system.


For ease of retrieval, we'll segment the report into groups of pages, with 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 11 on page 24 shows the ACIF indexer parameters required to process the data.

Accessing the report

This example provides instructions for 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 application. To process a sample report, you typically create or extract a subset of a complete report. The report for this example was generated on an OS/390 system. We transferred the report to the PC as a binary file. We can then load the sample data into the graphical indexer.

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

Make check payable to




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

TOTAL \$56.97

Return this portion with your payment.

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

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



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

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

THANK YOU FOR YOUR PAYMENT

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

CURRENT CHARGES \$56.97

DATE DUE JAN 29, 1993

AMOUNT DUE \$56.97

PAGE 1

Figure 9. 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 10. 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,'1',(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=PSF.FDEFLIB                                                 /* formdef directories       */
FONTLIB=PSF.FONTLIB                                                 /* font directories          */
OVLYLIB=PSF.OVLYLIB                                                 /* overlay directories       */
PDEFLIB=PSF.PDEFLIB                                                 /* pagedef directories       */
PSEGLIB=PSF.PSEGLIB                                                 /* pseg directories         */
USERLIB=ARS.PSF.USERLIB                                             /* user resources            */
RESTYPE=fdef,pseg,ovly                                              /* collect these resources   */

```

Figure 11. 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 fields 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=N0. 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 report

First, in the Parameter Source area, select Sample Data. Then click Modify to open the Open dialog box. Select the name of the file that contains the sample data. Click Open. OnDemand opens the Indexer Properties dialog box. After you click OK or Cancel, OnDemand loads the input 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 into the system, OnDemand uses the indexing parameters, options, and data values from the OnDemand application to index the data. If the data being loaded does not match the data that you use to generate the indexing parameters with the graphical indexer, then OnDemand may not index the data properly. For example, OnDemand may 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 '1'. 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.

Note: The graphical indexer specifies literals as hexadecimal strings. This is required when running ACIF on platforms other than OS/390 or z/OS. Since we will be running ACIF on the OS/390 or z/OS platform, expressing the literals as character strings is acceptable. After you have finished using the graphical indexer, you can manually change the hexadecimal strings to character strings, if you wish.

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

Locating the account number

To define the trigger that is 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 to open 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

Click the Display Indexer Parameters icon on the toolbar to open 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 Indexer Parameters dialog box.

Defining fields

ACIF uses fields 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 to open 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 to open 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 to open 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

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You 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 Indexer Parameters dialog box.

Defining indexes

Our next task in defining indexing parameters for the phone bill report is to define indexes. The indexes 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 to open 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 then click Add>>. OnDemand moves Field1 from the Fields list to the Order list. Click Add to add the index.

Defining the customer name index

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 then click Add>>. OnDemand moves Field2 from the Fields list to the Order list. Click Add to add the index.

Defining the bill date index

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 then click Add>>. OnDemand moves Field3 from the Fields list to the Order list. Click Add to add the index.

Click Done to close the Add an Index dialog box.

ACIF parameters

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You 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 Indexer 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 to open the Select dialog box. 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 to open the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties to open the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

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 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 that you make on the View Information page, and the triggers, fields, and indexes that you define. 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 49 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 libraries where the resources reside. For example, in the User Directories field, we entered ARS.PSF.USERLIB 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 to 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 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 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 indexing 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 12 on page 32 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, with 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 13 on page 33 shows the ACIF indexer parameters required to process the data.

Accessing the sample report

This example provides instructions for 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 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 OS/390 system. We transferred the report to the PC as a binary file. You can then load the sample data into the graphical indexer.

It is important that the sample data that you used to create the indexing information matches the actual data to be indexed and loaded into the system. When you load a report into the system, OnDemand uses the indexing parameters, options, and data values from the OnDemand application to index the data. If the data being loaded does not match the data that you used to generate the indexing parameters with the graphical indexer, then OnDemand may not index the data properly. For example, OnDemand may not be able to locate triggers, indexes, and fields or extract the correct index values.

Income Statement# 123-45-6789		Page 1 of 5 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 12. 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,'1',(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=PSF.FDEFLIB                                                 /* formdef directories        */
PDEFLIB=PSF.PDEFLIB                                                 /* pagedef directories        */
RESTYPE=fdef                                                         /* collect these resources    */

```

Figure 13. 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 fields for the statement number, statement date, and total income. (ACIF will generate page-level indexes for the types of income and subtotals fields. Page-level indexes are not stored in the database; only group-level indexes are stored in the database.)

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=N0. 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 report

First, in the Parameter Source area, select Sample Data. Then click Modify to open the Open dialog box. Select the name of the file that contains the sample data. Click Open. OnDemand opens the Indexer Properties dialog box. After you click OK or Cancel, OnDemand loads the input 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 into the system, OnDemand uses the indexing parameters, options, and data values from the OnDemand application to index the data. If the data being loaded does not match the data that you used to generate the indexing parameters with the graphical indexer, then OnDemand may not index the data properly. For example, OnDemand may 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 '1'. 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 to open 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 '1' in the Value area.

Note: The graphical indexer specifies literals as hexadecimal strings. This is required when running ACIF on platforms other than OS/390 or z/OS. Since we will be running ACIF on the OS/390 or z/OS platform, expressing the literals as character strings is acceptable. After you have finished using the graphical indexer, you can manually change the hexadecimal strings to character strings, if you wish.

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 to open 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 to open 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 to open 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 to open 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

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You 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 Indexer Parameters dialog box.

Defining fields

ACIF uses fields 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 to open 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 to open 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 to open 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 to open 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 to open 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

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You 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 Indexer Parameters dialog box.

Defining indexes

Our next task in defining indexing parameters for the income statement report is to define indexes. The indexes 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 to open 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 then click Add>>. OnDemand moves Field1 from the Fields list to the Order list. Click Add to add the index.

Defining the statement number index

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 then click Add>>. OnDemand moves Field2 from the Fields list to the Order list. Click Add to add the index.

Defining the total income index

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 then click Add>>. OnDemand moves Field3 from the Fields list to the Order list. Click Add to add the index.

Defining the type of income index

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 then click Add>>. OnDemand moves Field4 from the Fields list to the Order list. Click Add to add the index.

Defining the subtotal index

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 Subtotal1. 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 then click Add>>. OnDemand moves Field5 from the Fields list to the Order list. Click Add to add the index.

Click Done to close the Add an Index dialog box.

ACIF parameters

Click the Display Indexer Parameters icon on the toolbar to open a window that contains the indexing parameters. You 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 Indexer 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 to open the Select dialog box. 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 to open the Update an Index dialog box. Click Cancel. Click Trigger1 and then click Properties to open the Update a Trigger dialog box. Click Cancel. Close the Select dialog box.

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 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 that you make on the View Information page, and the triggers, fields, and indexes that you define. 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 49 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 PSF.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 to 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 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. Figure 14 on page 45 shows a page of a sample 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 15 on page 46 shows the indexer parameters required for ACIF to process the AFP file.

Accessing the report

Because we do not 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

Notices	xv
Trademarks and Service Marks	xv
 About this publication	xvii
Who should use this publication	xvii
How this publication is organized	xvii
Related IBM products	xvii
PSF/MVS: MVS Download feature	xvii
Product support	xviii
Our use of typefaces	xx
Platform-specific conventions	xx
Related documentation	xxi
ADSTAR Distributed Storage Manager for AIX Version 2.1	xxi
AIX Version 4.1	xxi
DATABASE 2 for AIX Version 2	xxi
MVS TCP/IP	xxii
OnDemand Version 2.1	xxii
Print Services Facility for AIX	xxii
Print Services Facility/MVS	xxii
RS/6000	xxiii
Transmission Control Protocol/Internet Protocol	xxiii
 Part 1. OnDemand ACIF Reference	1
 Chapter 1. Introducing ACIF	3
Overview	3
About ACIF	3
Indexing concepts	4
Indexing parameters	5
Converting data to AFP	6
AFP data	7
Mixed Object Document Content Architecture Data	7
System/370 line data	7
Mixed-mode data	8
Unformatted ASCII data	8
AFP resources	8
How OnDemand uses index information	9
Specifying indexing parameters for EBCDIC data	11
Accessing System/370 data	11
Creating indexing parameters	11

Figure 14. 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 */
UNIQUEBNGS=YES /* unique group names */
IMAGEOUT=ASIS /* leave images alone */
INDEXOBJ=ALL /* required for large object */

/* RESOURCE INFORMATION */
FORMDEF=F1A10110 /* default formdef */
USERLIB=ARS.PSF.USERLIB /* resource library */
RESTYPE=NONE /* do not collect resources */

```

Figure 15. 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 will uncompress the appropriate groups of pages. To enable large object support, you must 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 fields 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 indexing parameters

Open the Add an Application window to the Indexer Information page. In the Parameter Source area, select Keyboard. Click Modify to open the Edit Indexer 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 Edit Indexer 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=ARS.PSF.USERLIB
- RESTYPE=NONE

We're finished defining indexing information for the AFP document. Close the Edit Indexer Parameters window. OnDemand asks us if we want to save our changes. Click Yes to 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 sample AFP document contains TLEs that contain the index attribute names and values. ACIF extracts the index attribute names and values from the AFP document and writes them to an index object file. The index attribute names in the AFP document must match the application group database field names. If they do not, then we can map the attributes names to the database field names on the Load Information page. To do so, select a database field in the Application Group DB Name list. Verify that the value in the Load ID Name field is the index attribute name.

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 use the OnDemand application to process your input data. When you use the OnDemand application to process your input data:

- ACIF is automatically invoked if the input data needs to be indexed.
- Any values that you specify for the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, PARMDD, and RESOBJDD parameters are ignored.

If you run ACIF outside of the OnDemand application, then you should verify the values of the INDEXDD (page 68), INPUTDD (page 71), MSGDD (page 73), OUTPUTDD (page 75), PARMDD (page 78), and RESOBJDD (page 82) parameters.

CC

Required

No

Default Value

YES

Data Type

AFP, Line

Determines whether the input contains carriage-control characters.

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 49.

CCTYPE

Required

No

Default Value

A

Data Type

AFP, Line

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, then ACIF assumes that the input contains ANSI carriage-control characters encoded in ASCII. If you are running ACIF on a z/OS or OS/390 system, then ACIF assumes that the carriage-controls are encoded in EBCDIC.

Note: It is very important to correctly identify the type of carriage control characters in the input file. ACIF may process an input file even though the CCTYPE parameter incorrectly identifies the type of carriage control characters in the input file. However, the output file may be unusable. If you have questions about the type of carriage control characters that are in the input file, then you should contact someone who can help you inspect the input data and determine the correct type of carriage control characters in the input file.

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 49.

CHARS

Required

No

Default Value

(None)

Data Type

AFP

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, then the CHARS parameter also provides the names of the fonts that ACIF saves in the resource group. The CHARS parameter 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.

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

When ACIF is used to convert S/390® line data or mixed-mode data, you must specify a page definition with the PAGEDEF parameter. You can then specify the fonts either in the page definition or with the CHARS parameter, but not both. You cannot mix fonts specified in a page definition with fonts specified with CHARS for a single file. If you use the CHARS parameter to specify fonts, but you also use the PAGEDEF parameter to specify a page definition that names fonts, the CHARS parameter is ignored. Therefore, if your page definition names fonts, you should not use the CHARS parameter.

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.

The fonts that you specify must reside in a library that is specified with FONTLIB parameter or reside in a user library that is specified with the USERLIB parameter.

Related parameters

- FONTLIB parameter on page “FONTLIB” on page 61.
- PAGEDEF parameter on page 76.
- USERLIB parameter on page “USERLIB” on page 89.

CONVERT

Required

No

Default Value

YES

Data Type

AFP, Line

Determines whether ACIF converts the input data to AFP.

- To collect any type of resources, you must specify CONVERT=YES (the default value).
- ACIF can now generate page-level indexes without converting the line data input file to AFP. If you do not otherwise need to convert the line data input file to AFP, but you do want ACIF to generate page-level indexes, then you should specify CONVERT=NO.

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.

Related parameters

The RESTYPE parameter on page 83.

CPGID

Required

No

Default Value

500

Data Type

AFP, Line

Identifies the code page of the index data. For most customers, the CPGID will be the same as the code page of the input data.

Important: When loading data using ACIF, the locale must be set appropriately for the CPGID parameter. For example, if CPGID=273 is specified, set the LC_ALL environment variable to De_DE.IBM-273 or some other locale that correctly identifies upper and lower case characters in code page 273.

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

The code page identifier is used by the OnDemand client programs to display indexing information. The client programs use this identifier with code page translation tables to represent the index attribute and value data. For code-page numbers less than 100, add leading zeros (for example, 037). If a non-decimal value is specified, ACIF reports an error condition and ends processing.

Syntax

CPGID=*value*

Options and values

The *value* can be:

500

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

No

Default Value

NO

Data Type

AFP, Line

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

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.

EXTENSIONS

Required

No

Default Value

NONE

Data Type

AFP

Determines the extended options that ACIF uses. Extensions are MO:DCA data stream advanced features that might not be supported for all presentation devices. You should use care when choosing these options and make sure that they are available on your print server, viewer, or printer.

Syntax

`EXTENSIONS=`*value*

Options and values

The *value* can be:

NONE

ACIF does not use any extended options.

ALL

ACIF uses all of the extended options.

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

RESORDER

When the RESORDER value is specified, inline resources do not have to appear in any particular order in the input file, although they must all appear before the beginning of the document. ACIF will read the inline resources into memory and use them when they are requested. Note: If there are many inline resources and little internal memory available, the system may run out of memory when using this option.

When the RESORDER value is not specified, inline resources must appear in the input file in the order in which they are used.

BOX

ACIF uses the GOCA box drawing order when using a Record Formatting Page Definition.

CELLED

ACIF uses the IOCA Replicate and Trim function when converting IM1 celled images. This image might reduce the number of bytes needed for a raster image. It requires that IMAGEOUT=IOCA be specified (the default).

FRACLINE

ACIF uses the GOCA fractional line width drawing order when using a Record Formatting Page Definition.

PRCOLOR

ACIF uses GOCA process color drawing orders when using a Record Formatting Page Definition.

SPCMPRS

ACIF uses the repeat string PTOCA order to remove trailing blanks from line data and compress embedded blanks.

extension,...extension

A comma-separated list of two or more specific types of extended options. For example, to specify that ACIF should use the PRCOLOR and BOX extended options, use the following format of the parameter:

`EXTENSIONS=prcolor,box`

Related parameters

IMAGEOUT parameter on page 64.

FDEFLIB

Required

No

Default Value
(None)

Data Type
AFP

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

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

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

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

Notes:

1. Data sets must be specified as fully-qualified names without quotation marks.
2. Separate data set names with a comma.
3. For systems before MVS/DFP™ Version 2.3, data sets must be concatenated with the largest block size first.
4. The FDEFLIB parameter is required if the USERLIB parameter is not specified. If the FDEFLIB parameter is not specified, ACIF reports an error condition and ends processing.

Related parameters

USERLIB parameter on page 89.

FIELD

Required
Yes

Default Value
(None)

Data Type
AFP, Line

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. ACIF supports the following types of fields:

- Trigger field, which is based on the location of a trigger string value
- Constant field, which allows you to provide the actual index value that is stored in the database
- Transaction field, which you can use to index input data that contains 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

```
FIELDn=record,column,length,(TRIGGER=n,BASE={0 |  
TRIGGER},MASK='@#=-^%')[,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 (zero) 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 (one) 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, unless a DEFAULT value is specified.

length

The number of contiguous bytes (characters) that compose the field. The supported range of values are 1 (one) 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 that ACIF uses to locate the field. This is an optional parameter, unless you specify the MASK keyword; then you must specify a trigger that was defined with TYPE=FLOAT. Replace *n* with the number of a defined TRIGGER parameter. The default value is TRIGGER1.

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.

MASK='@#=-~^%'

Specifies a pattern of symbols that ACIF uses to match data located in the field columns. If the data matches the MASK, then ACIF selects the field. **Important:** If you specify a MASK, then the field must be based on a floating trigger. An INDEX parameter that is based on the field cannot include any other fields and must not create grouprange or pagerange indexes.

You can specify the following symbols in the MASK:

@	Matches alphabetic characters.
#	Matches numeric characters.
=	Matches any character.
¬	Matches any non-blank character.
^	Matches any non-blank character.
%	Matches the blank character and numeric characters.

For example, given the following definitions:

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

ACIF selects the field only if the data in the field columns contains numeric characters.

DEFAULT=*'value'*

Determines the default value for the index when a record is not long enough to contain the field data.

Note: If a record is not long enough to contain the field data and you do not specify a default value, ACIF will fail.

The default value can be specified either as a character string or a hexadecimal string. If the data to be indexed is anything other than ASCII, then the default value must be specified as a hexadecimal string. For example, X'value'. 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

Note: A constant field cannot be concatenated with a field that is based on a floating trigger. A constant field is a field for which you specify the actual index value that will be stored in the database. It is possible to generate an index value by concatenating or combining the value that you specify for a constant field with the value that ACIF extracts from a document by using a trigger field. However, the trigger field cannot be based on a floating trigger.

FIELD*n*=*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 that gets 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 perform any validity checking on 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 field must specify an asterisk, meaning ACIF searches every record in the group.

*

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

Note: If you enter a value other than an asterisk, ACIF ignores the value. When you specify 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 90) 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 52.

INDEX parameter on page 65.

TRIGGER parameter on page 86.

FILEFORMAT

Required

No

Default Value

(None)

Data Type

Line

Specifies the format of the output data set. If the FILEFORMAT parameter is not specified, ACIF will write the output data set according to the DCB characteristics that are specified for the data set that is identified by the OUTPUTDD parameter.

Syntax

```
FILEFORMAT=value
```

Options and values

The *value* can be **HFSOUT**. For example:

```
FILEFORMAT=HFSOUT
```

ACIF will write records to the output data set (specified by the DCB) with a two-byte length prefix.

The FILEFORMAT parameter applies to the output data set, not the index or resource data sets.

The FILEFORMAT parameter is used only with CONVERT=NO, so that the data can be loaded into OnDemand.

OnDemand requires that line data either be fixed length or contain two-byte length prefixes.

Related parameters

CONVERT parameter on page 51.
OUTPUTDD parameter on page 75.

FONTLIB

Required

No

Default Value

(None)

Data Type

AFP

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

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

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

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

Notes:

1. Data sets must be specified as fully-qualified names without quotation marks.
2. Separator data set names with a comma.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
4. This is a required parameter if font retrieval is requested and the USERLIB parameter is not specified, or if you specify MCF2REF=CPCS and any coded fonts are referenced in the input file or in an overlay. The RESTYPE parameter determines whether fonts are to be retrieved for inclusion in the resource data set. If this parameter is not specified, and font retrieval is requested or a coded font is referenced, ACIF reports an error condition and ends processing.

Related parameters

MCF2REF parameter on page 73
RESTYPE parameter on page 83
USERLIB parameter on page 89.

FORMDEF

Required

Yes

Default Value

(None)

Data Type

AFP

Specifies the member 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 to convert a line data file to AFP.

The form definition can be located:

- Inline in the file (that is, within the file itself)
- In a z/OS or OS/390 user library referenced in the USERLIB parameter
- In a z/OS or OS/390 library referenced in the FDEFLIB parameter

To use an inline form definition, do the following:

- Include an inline form definition in the member
- Specify CC=YES to indicate that the member contains carriage control characters. If the length of the records in the form definition is less than or equal to the logical record length defined for the member, you can specify fixed length records for the record format:
 - In UNIX and Windows, specify FILEFORMAT=RECORD,n (where n is the logical record length defined for the file).
 - In z/OS and OS/390, specify record format FBA (fixed block with ANSI carriage control characters) or FBM (fixed block with machine carriage control characters) for the input data set.

If the length of the records in the form definition is greater than the logical record length defined for the member, you must specify variable length records:

- In UNIX and Windows, specify FILEFORMAT=RECORD. The first two bytes of each record determine the record length.
- In z/OS and OS/390, specify record format VBA (variable blocked with ANSI carriage control characters) or VBM (variable blocked with machine carriage control characters) for the input data set.
- Specify the FORMDEF parameter with one of these values:
 - *fdefname*, where *fdefname* is the name of the inline form definition.
If the name specified in the FORMDEF parameter does not match the name of an inline form definition, ACIF looks for the form definition in the FORMDEF search path.
 - DUMMY
If you specify FORMDEF=DUMMY but the file does not include an inline form definition, ACIF looks for the form definition named DUMMY. If ACIF cannot find a form definition named DUMMY, it reports an error and ends processing.

If the form definition file is in a library, use the USERLIB or FDEFLIB parameter to specify the data sets. For example:

```
FORMDEF=MEMO
USERLIB=USER.RESOURCES
FORMDEF=MEMO
FDEFLIB=USER.RESOURCES
```

Syntax

FORMDEF=*fdefname*

Options and values

The *fdefname* is the member name of the form definition, one to eight alphanumeric or national characters, including the two-character prefix, if there is one.

Notes:

1. If the file name of the form definition includes a file extension, do not use the file extension when specifying the form definition. For example, to use a form definition named MEMO.FDEF38PP, specify FORMDEF=MEMO.
2. ACIF requires a form definition to process the input file (even though the form definition actually gets used at print time). If you do not specify the FORMDEF parameter or you specify FORMDEF without a form definition file name, ACIF reports an error condition and ends processing.

Related parameters

- FDEFLIB parameter on page 54.
- USERLIB parameter on page 89.

GROUPMAXPAGES

Required

No

Default Value

(None)

Data Type

AFP, Line

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 (one) to 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.

Syntax

GROUPMAXPAGES=*value*

Options and values

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

Related parameters

INDEX parameter on page 65.

GROUPNAME

Required

No

Default Value

INDEX1

Data Type

AFP

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, do not use an index that contains a field based on a floating trigger.

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 65.

IMAGEOUT

Required

No

Default Value

IOCA

Data Type

AFP, Line

Determines the format of the image data produced by ACIF.

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.

Related parameters

EXTENSIONS parameter on page 53.

INDEX

Required

Yes

Default Value

(None)

Data Type

AFP, Line

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.

Syntax

INDEX n =*name*,FIELD n n[,...FIELD n n][,(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=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.

ACIF can now generate page-level indexes without converting the line data input file to AFP. If you do not otherwise need to convert the line data input file to AFP, but you do want ACIF to generate page-level indexes, then you should specify CONVERT=NO.

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.

ACIF can now generate page-level indexes without converting the line data input file to AFP. If you do not otherwise need to convert the line data input file to AFP, but you do want ACIF to generate page-level indexes, then you should specify CONVERT=NO.

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 55.

INDEXOBJ parameter on page 69.

INDEXDD

Note: This parameter is ignored when you process reports with the OnDemand application.

Required

No

Default Value

INDEX

Data Type

AFP, Line

Specifies the DD name for the index object file. When ACIF is indexing the input file, it writes indexing information to the specified DD name. The suggested DCB characteristics for the file are:

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

If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the INDEX DD statement. If that happens, ACIF ends processing abnormally. If the INDEXDD parameter is not specified, ACIF uses INDEX as the default DD name.

- Variable blocked format
- Physical sequential format

Syntax

INDEXDD=*DD name*

Options and values

The *DD name* is a one- to eight-byte character string.

INDEXOBJ

Required

No

Default Value

GROUP

Data Type

AFP, Line

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

Syntax

INDEXOBJ=*value*

Options and values

The *value* can be:

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.

BDTLY

ACIF normally removes any Begin/End Document structured fields from the input file and generates a single BDT/EDT for the entire output because MO:DCA indexes are relative to the Begin Document structured field. However, the stapling function uses BDT/EDT to indicate document boundaries for stapling. A new indexing option has been added to allow ACIF to pass through any BDT/EDT and not create its own. This file is suitable for printing, but should not be used with indexing because the resultant index will not be MO:DCA compliant and may not be processed correctly by programs which use the index, such as OnDemand.

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.

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 65.

INDEXSTARTBY

Required

No

Default Value

1

Data Type

AFP, Line

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. The maximum value for INDEXSTARTBY is 99.

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 with the document. 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 with the document.

Syntax

INDEXSTARTBY=*value*

Options and values

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

A 0 (zero) indicates that there is no limit to the page where ACIF must find an indexing field.

INDEXEXIT

Required

No

Default Value

(None)

Data Type

AFP, Line

Specifies the name of the module ACIF loads during initialization and subsequently calls for every record (structured field) it writes to the index object file (specified with the INDEXDD parameter). If this parameter is not specified, no index record exit is used.

Syntax

INDEXEXIT=*modulename*

Options and values

The *modulename* is a one- to eight-byte character name of the index record exit program.

INPEXIT

Required

No

Default Value

(None)

Data Type

AFP, Line

Specifies the name of the module ACIF loads during initialization and subsequently calls for every input record it reads from the input file (specified with the INPUTDD parameter). If this parameter is not specified, no input record exit is used.

Syntax

INPEXIT=*modulename*

Options and values

The *modulename* is the one- to eight-byte character name of the input record exit program.

INPUTDD

Note: This parameter is ignored when you process reports with the OnDemand application.

Required

No

Default Value

INPUT

Data Type

AFP, Line

Specifies the DD name for the input file ACIF processes. When ACIF processes an input file, it reads from this DD name. If INPUTDD is not specified, ACIF uses INPUT as the default DD name.

Syntax

INPUTDD=*DD name*

Options and values

The *DD name* is the one- to eight-byte character DD name for the input file that ACIF will process.

INSERTIMM

Required

No

Default Value

NO

Data Type

AFP

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

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 61.

LINECNT

Required

No

Default Value

0

Data Type

Line

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.

The default value is 0 (zero) and means that ACIF will not create any page breaks. The document will be stored as a single page if there are no carriage control characters present in the input data.

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 49.

The CONVERT parameter on page 51.

MCF2REF

Required

No

Default Value

CPCS

Data Type

AFP

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).

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.

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 83.

MSGDD

Note: This parameter is ignored when you process reports with the OnDemand application.

Required

No

Default Value

SYSPRINT

Data Type

AFP, Line

Specifies the DD name for the data set to which ACIF writes messages. When ACIF processes an input data set, it writes message to the specified DD name. If MSGDD is not specified, ACIF uses SYSPRINT as the default DD name.

Syntax

MSGDD=*DD name*

Options and values

The *DD name* is the one- to eight-byte character DD name for the ACIF message file.

NEWPAGE

Required

No

Default Value

1

Data Type

AFP, Line

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).

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 49.

The CONVERT parameter on page 51.

OUTEXIT

Required

No

Default Value

(None)

Data Type

AFP, Line

Specifies the name of the output record exit program. This is the module ACIF loads during initialization and subsequently calls for every output record it writes to the output document file (**OUTPUTDD**). If this parameter is not specified, no output record exit is used.

SyntaxOUTEXIT=*modulename***Options and values**

The *modulename* is the one- to eight-byte character name of the output record exit program.

OUTPUTDD

Note: This parameter is ignored when you process reports with the OnDemand application.

Required

No

Default Value

OUTPUT

Data Type

AFP, Line

Specifies the DD name for the output document file ACIF produces when it processes a file. When ACIF processes a print file, it writes the resultant converted print data to this DD name. Suggested DCB characteristics of the file are:

- Variable blocked format
- A maximum record length of 32756
If a record length other than 32756 is specified, ACIF might produce a record of length greater than that which is allowed by the OUTPUT DD statement. If this happens, ACIF ends processing abnormally.
- A block size of 32760
- Physical sequential format

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

SyntaxOUTPUTDD=*DD name***Options and values**

The *DD name* is a one- to eight-byte character DD name for the output file.

OVLYLIB**Required**

No

Default Value
(None)

Data Type
AFP

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

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

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

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

Notes:

1. Data sets must be specified as fully-qualified names without quotation marks.
2. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
3. This is a required parameter if overlay retrieval is requested and USERLIB is not specified. The RESTYPE parameter determines whether overlays are to be retrieved for inclusion in the resource data set. If this parameter is not specified, and overlay retrieval is requested, ACIF reports an error condition and ends processing.

Syntax

```
OVLYLIB=dsname1[dsname2][dsname3...]
```

Options and values

The names of one to eight data sets that compose the overlay library. Delimit data set names with the comma (,) character.

Related parameters

USERLIB parameter on page 89.

PAGEDEF

Required
No

Default Value
(None)

Data Type
AFP

Specifies the member 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 S/390 line data, mixed-mode data, or unformatted ASCII data into AFP.

The page definition can be located:

- Inline in the file (that is, within the file itself)
- In a library referenced in the USERLIB or PDEFLIB parameter

To use an inline page definition you must do the following:

- Include an inline page definition in the file.
- Specify PAGEDDEF with one of these values:
 - *pdefname*, where *pdefname* is the name of the inline page definition. If the name specified on the PAGEDDEF parameter does not match the name of an inline page definition, ACIF looks for the page definition in the PDEFLIB search path or uses the page definition from the resource library.
 - DUMMY
If you specify PAGEDDEF=DUMMY and there is no inline page definition, ACIF searches the PDEFLIB search path for a page definition named DUMMY. If ACIF cannot find a page definition named DUMMY, it reports an error condition and stops processing.
- Specify CC=YES to indicate that the file contains carriage control characters. If the length of the records in the page definition is less than or equal to the logical record length defined for the file, you can specify fixed length records for the record format:
 - On UNIX and Windows servers, specify FILEFORMAT=RECORD,n (where n is the logical record length defined for the file).
 - In z/OS and OS/390, specify record format FBA (fixed block with ANSI carriage control characters) or FBM (fixed block with machine carriage control characters) for the input data set.

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 servers, specify FILEFORMAT=RECORD. The first two bytes of each record determine the record length.
- In z/OS and OS/390, specify record format VBA (variable blocked with ANSI carriage control characters) or VBM (variable blocked with machine carriage control characters) for the input data set.

If the page definition file is in a library, use the USERLIB or PDEFLIB parameter to specify the data sets. For example:

```
PAGEDDEF=MEMO
USERLIB=USER.RESOURCES

PAGEDDEF=MEMO
PDEFLIB=USER.RESOURCES
```

Syntax

PAGEDDEF=*pdefname*

Options and values

The *pdefname* can be one to eight alphanumeric or national characters, including the two-character prefix, if there is one.

Notes:

1. If the file name of the page definition includes a file extension, do not use the file extension when specifying the page definition. For example, to use a page definition named MEMO.PDEF38PP, specify PAGEDDEF=MEMO.

2. ACIF does not require a page definition when indexing an AFP data stream file. However, ACIF does require a page definition to transform an input file that contains S/390 line data, mixed-mode data, or unformatted ASCII data into MO:DCA-P. If you are transforming such an input file and you do not specify the PAGEDEF parameter or you specify PAGEDEF without a page definition file name, ACIF reports an error condition and ends processing.
3. If you use the PAGEDEF parameter to specify a page definition that names fonts, but you also use the CHARS parameter to specify fonts, the CHARS parameter is ignored. Therefore, if your page definition names fonts, you should not use the CHARS parameter.
4. ACIF does not support a parameter equivalent to the LINECT parameter on the /*JOBPARM, /*OUTPUT, and OUTPUT JCL statements. The maximum number of lines processed on a page is defined in the page definition.

Related parameters

- CHARS parameter on page 50.
- PDEFLIB parameter on page 78.
- USERLIB parameter on page 89.

PARMDD

Note: This parameter is ignored when you process reports with the OnDemand application.

Required

No

Default Value

SYSIN

Data Type

AFP, Line

Specifies the DD name for the file that contains the ACIF parameters, options, and data values. Specify the PARMDD parameter only when running ACIF from outside of the OnDemand load process. When you process files using the OnDemand load process, OnDemand automatically retrieves the ACIF parameters from the database. If the PARMDD parameter is not specified, then ACIF uses SYSIN as the default DD name.

Syntax

PARMDD=*DD name*

Options and values

The *DD name* is a one- to eight-byte character DD name for the parameter file.

PDEFLIB

Required

No

Default Value

(None)

Data Type

AFP

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

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

Notes:

1. For systems before MVS/DFP Version 2.3, files must be concatenated with the largest block size first.
2. This is a required parameter if the input file contains any line-mode data and USERLIB is not specified. If this parameter is not specified and the input file contains line-mode data, ACIF reports an error condition and ends processing.

Syntax

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

Options and values

You can specify a maximum of eight data sets. For example:

PDEFLIB=SYS1.PDEFLIB,USER.PDEFLIB

Data sets must be specified as fully-qualified names without quotation marks. Delimit data set names with the comma (,) character.

Related parameters

PAGEDEF parameter on page 76.

USERLIB parameter on page 89.

PRMODE

Required

No

Default Value

(None)

Data Type

AFP

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.

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.

SOSI4

Intended for use on workstation platforms where the user has DBCS text being converted from ASCII to EBCDIC, and is also using a PAGEDEF to convert the data to AFP. SOSI4 processing is similar to SOSI2, with the following difference. Specifying SOSI4 will cause ACIF to scan the input (EBCDIC) for SOSI characters and if any are found, they will be skipped but not counted as part of the input columns. This means that the PAGEDEF FIELD offsets should be correct after conversion from ASCII to EBCDIC and the user does not need to account for SOSI characters when computing the PAGEDEF FIELD offsets.

Note: The SOSI characters do have to be counted in determining the ACIF trigger and field offsets.

aaaaaaaa

Specifies an eight-byte alphanumeric string. The value is supplied to all of the ACIF user exits.

Related parameters

The CHARS parameter on page 50.

PSEGLIB

Required

No

Default Value

(None)

Data Type

AFP

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

If the USERLIB parameter is also specified, ACIF searches for the resource in the files specified in USERLIB before searching the files identified in PSEGLIB.

Notes:

1. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.
2. This is a required parameter if page segment retrieval is requested and USERLIB is not specified. The RESTYPE value determines whether page segments are to be retrieved for inclusion in the resource data set. If this parameter is not specified, and page segment retrieval is requested, ACIF reports an error condition and ends processing.

Syntax

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

Options and values

You can specify a maximum of eight data sets. For example:

PSEGLIB=SYS1.PSEGLIB,USER.PSEGLIB

Data sets must be specified as fully-qualified names without quotation marks. Delimit data set names with the comma (,) character.

Related parameters

USERLIB parameter on page 89.

RESEXIT**Required**

No

Default Value

(None)

Data Type

AFP

Specifies the name of the resource exit program. This is name of the module ACIF loads during initialization and subsequently calls each time it attempts to retrieve a requested resource from a library. If this parameter is not specified, no resource exit is used.

Syntax

RESEXIT=*modulename*

Options and values

The *modulename* is the one- to eight-byte character name of the resource exit program.

RESFILE**Required**

No

Default Value

SEQ

Data Type

AFP

Specifies the format of the resource file that ACIF creates. ACIF can create either a sequential data set (SEQ) or a partitioned data set (PDS) from resources it retrieves from the PSF resource libraries. If this parameter is not specified, ACIF writes a sequential data set to the DDname specified in the RESOBJDD parameter, assuming a sequential format.

It is important that the parameters you use to allocate the RESOBJDD data set be compatible with the value of the RESFILE parameter. For example, if you specify RESFILE=PDS, then DSORG=P0 must be specified in the DD statement of the data set named by the RESOBJDD parameter. In addition, the SPACE parameter must include a value for directory blocks, such as SPACE=(12288,(150,15,15)), in the DD statement of the data set named by the RESOBJDD parameter.

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

Syntax

RESFILE=*type*

Options and values

The *type* can be:

SEQ

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

PDS

Creates a member that can be placed in a user library or in a system library. The file created by selecting PDS cannot be concatenated to the document file or used as inline resources.

Related parameters

RESOBJDD parameter on page 82.

RESOBJDD

Note: This parameter is ignored when you process reports with the OnDemand application.

Required

No

Default Value

RESOBJ

Data Type

AFP

Specifies the DD name for the resource file produced by ACIF. The resource file contains all of the resources required to view or reprint pages of the report.

Syntax

RESOBJDD=*DD name*

Options and values

The DD name is the one- to eight-byte character DD name for the resource group file. Suggested DCB characteristics for the file are:

- Variable blocked format
- A maximum record length of 32756

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

- A block size of 32760
- Physical, sequential format

If you do not specify the RESOBJDD parameter, ACIF uses RESOBJ as the default DD name.

RESTYPE

Required

No

Default Value

NONE

Data Type

AFP

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

Note: To collect any resources, you must specify CONVERT=YES (the default value). Resources are not collected when you specify CONVERT=NO.

Syntax

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

Options and values

The values are:

NONE

No resource file is created. This is the default value.

ALL

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

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.

OBJCON

Specifies that all object container files requested by the input data stream be included in the resource file.

INLINE

Specifies that inline resources are written to the output file in addition to being written to the resource file. The resources precede the document in the output file. For example, RESTYPE=FONT,PSEG,INLINE causes any inline fonts and page segments to be written to the output file. Also, both inline and library fonts and page segments are written to the resource file.

Important: Do not use the INLINE option for documents loaded into OnDemand. OnDemand requires a separate resource file.

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

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.

If a resource is inline and ACIF is collecting that type of resource, the resource will be saved in the resource file regardless of whether it is used in the document, unless EXTENSIONS=RESORDER is specified in the ACIF parameters. Another method to remove unwanted resources from the resource file is to use a resource exit.

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

- B1 for BCOCA objects
- E1 for encapsulated PostScript objects
- G1 for GOCA objects

- H1 for microfilm setup objects
- I1 for IOCA objects
- IT for IOCA tile objects
- PP for PDF single-page objects
- PR for PDF resource objects
- S1 for page segments

Related parameters

The CONVERT parameter on page 51.
 The MCF2REF parameter on page 73.
 The RESOBJDD parameter on page 82.

TRACE

Required

No

Default Value

NO

Data Type

AFP, Line

Specifies that ACIF should provide diagnostic trace information while processing the file.

Syntax

TRACE=*value*

Options and values

The *value* can be:

NO

ACIF does not produce diagnostic trace records.

YES

ACIF uses the facilities of the OS/390 and MVS™ Generalized Trace Facility (GTF) to produce diagnostic trace records. ACIF writes GTF trace records with a user event ID of X'314'. To capture ACIF GTF records, GTF needs to be started with the option TRACE=USRP, and subsequently modified with USR=(314).

Tracing increases processor overhead and should be turned off unless you need to do problem determination. If YES is specified and GTF is active, ACIF ends with a Return Code 4 (RC=4).

TRC

Required

No

Default Value

NO

Data Type

AFP, Line

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.

Notes:

1. The order in which the fonts are specified in the CHARS parameter establishes which number is assigned to each associated TRC. For example, the first font specified is assigned 0, the second font 1, and so on.
2. If you specify TRC=YES 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.
3. 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.

Syntax

TRC=*value*

Options and values

The *value* can be:

NO

The input does not contain TRCs.

YES

The input contains TRCs.

Related parameters

The CHARS parameter on page 50.

TRIGGER

Required

Yes

Default Value

(None)

Data Type

AFP, Line

Specifies 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.

Note: In ACIF, a *group* is a named collection of sequential pages that form a logical subset of an input file. A group must contain at least one page; a group can contain all of the pages in an input file. However, most customers define their group triggers so that ACIF can logically divide an input file into smaller parts, such as by statement, policy, bill, or, for transaction data, number of pages. A group is determined when the value of an index changes (for example, account number) or when the maximum number of

pages for a group is reached. ACIF generates indexes for each group in the input file. Because a group cannot be smaller than one page, a group trigger should not appear more than once on a page. Please see the BREAK option of the INDEX parameter for more information about breaking groups.

In OnDemand, each indexed group of pages is known as a *document*. When you index an input file and load the data into the system, OnDemand stores the group indexes that are generated by ACIF into the database and stores the documents on storage volumes. OnDemand uses the group indexes to determine the documents that match the search criteria that is entered by the user.

Syntax

TRIGGER*n=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 (zero, 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 1 (one) to 32756. Specifying an * (asterisk) or 0 (zero) results in ACIF scanning the record from left to right looking for the trigger string value. A value of 1 (one) refers to the first byte of the record. For files containing carriage control characters, column 1 (one) refers to the carriage control character.

value

The actual string value ACIF uses to match the input data. The string value is case sensitive. When running ACIF on an OS/390 or z/OS system, the value can be entered either as a character string or as a hexadecimal string. The value can be from 1 (one) 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 (zero) 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.

Notes

1. ACIF requires that at least one TRIGGERn or FIELDn value appear within the page range that is specified by the INDEXSTARTBY parameter, unless the INDEXSTARTBY parameter is set to zero. If no TRIGGERn or FIELDn parameter is satisfied within the INDEXSTARTBY page range, then ACIF stops processing and issues an error message.
2. At least one TRIGGERn or FIELDn value must exist on the first page of every unique page group. ACIF cannot detect an error condition if TRIGGERn or FIELDn is missing, but the output might be incorrectly indexed.
3. TRIGGER1 must be specified when ACIF is requested to index the file.
4. An error condition occurs if you specify any TRIGGERn parameters when the input file contains indexing tags.

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'E3A8978540968640C99583969485',(TYPE=FLOAT)
```

Related parameters

The FIELD parameter on page 55.

UNIQUEBNGS

Required

No

Default Value

YES

Data Type

AFP, Line

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.

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 53.

USERLIB

Required

No

Default Value
(None)

Data Type
AFP

Specifies data sets containing AFP resources for processing the input data set. You can specify a maximum of eight data sets. ACIF dynamically allocates these data sets and searches for resources in them in the order specified in the USERLIB parameter. If a resource is not found, ACIF searches the appropriate resource libraries defined for that resource type (for example, PDEFLIB for page definitions). The libraries you specify can contain any AFP resources (fonts, page segments, overlays, page definitions, or form definitions). If Resource Access Control Facility (RACF[®]) is installed on your system, RACF checks the authority of the user ID requesting access to a user library (data set). If ACIF is not authorized to allocate the data set, it reports an error condition and ends processing.

Notes:

1. Because AFP resources (except page segments) have reserved prefixes, naming conflicts should not occur.
2. An inline resource overrides a resource of the same name contained in the USERLIB parameter.
3. For systems before MVS/DFP Version 2.3, data sets must be concatenated with the largest block size first.

Syntax

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

Options and values

You can specify a maximum of eight data sets. For example:

USERLIB=USER.IMAGES,USER.AFP.RESOURCE

Data sets must be specified as fully-qualified names without quotation marks. Delimit data set names with the comma (,) character.

USERMASK

Required
No

Default Value
(None)

Data Type
AFP

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.

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. When running ACIF on an OS/390 or z/OS system, the string can be entered either as a character string or as a hexadecimal string. For example:

```
USERMASK=1,'*',X'C181C282C383'
```

Or:

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

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 fifteen 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 CPGID parameter on page 52.

The FIELD parameter on page 55.

Chapter 4. Messages

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

ACIF messages contain instructions for the OnDemand, PSF or Infoprint Manager system programmer. Please show your system programmer these messages, because they might not be contained in the OnDemand, PSF or Infoprint Manager messages publications.

Note: Please see *IBM DB2 Content Manager OnDemand: Messages and Codes*, SC27-1379 for a list of the messages that can be generated by ACIF, along with explanations of the messages and actions that you can take to respond to the messages. You can get a copy of the *Messages and Codes* publication from www.ibm.com/software/data/ondemand/390/library.html.

Chapter 5. User exits and attributes of the input file

A user exit is a point during ACIF processing that lets 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 chapter contains programming interface information and describes the four user programming exits provided with ACIF. It also describes the information that ACIF provides to the exits about the input print file attributes.

User programming exits

ACIF provides these sample programming exits so that you can custom the program:

- Input record
- Index record
- Output record
- Resource

These exits are described in the following sections. Sample OS/390, VM, or VSE DSECTs are shown for each programming exit.

Using 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 49.

Sample exits for OS/390, VM, and VSE are available from the IBM Printing Systems Division Web page at www.ibm.com/printers. Search the Web site for “ACIF sample code”.

Input record exit

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

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

In a MO:DCA-P document, indexing information can be passed in the form of a Tag Logical Element (TLE) structured field (see Chapter 7, “ACIF data stream information,” on page 113 for more information). The exit program can create these structured fields while ACIF is processing the print file. You can insert No Operation (NOP) structured fields into the input file in place of TLEs and use ACIF’s indexing parameters (FIELD, INDEX, and TRIGGER) to index the NOPs. This is an alternative to modifying the application in cases where the indexing information is not consistently present in the application output.

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

Figure 16 contains a sample DSECT that describes the control block for OS/390, VM, or VSE exit programs.

PARMLIST	DSECT		Parameters for the input record exit
WORK@	DS	A	Address of 16-byte static work area
PFATTR@	DS	A	Address of print-file-attribute information
RECORD@	DS	A	Address of the input record
	DS	A	Reserved for future use
RECORDLN	DS	H	Length of the input record
	DS	H	Reserved for future use
REQUEST	DS	X	Add, delete, or process the record
EOF	DS	C	EOF indicator

Figure 16. OS/390, VM, or VSE Sample Input Record Exit DSECT

The address of the control block containing the following parameters is passed to the input record exit. For OS/390, VM, and VSE, the address is passed to a standard parameter list pointed to by Register 1.

WORK@ (Bytes 1–4)

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

PFATTR@ (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 102 for more information about the format of this data structure and the information that 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. The record resides in a 32 KB (where KB equals 1024 bytes) buffer.

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 let it insert the next record. The exit program can set this value to save the current record, insert a record, and then

supply the saved record at the next call. After the exit inserts the last record, the exit program must reset the **REQUEST** byte to X'00'.

A value of X'00' on entry to the exit program specifies that the record is to 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 one-byte character code that specifies whether an EOF condition has been encountered. When **EOF** is signaled (**EOF=Y**), the last record has already been presented to the input exit, and the input file has been closed. The pointer **RECORD@** is no longer valid. Records cannot be inserted when **EOF** is signaled. The following are the only valid values for this parameter:

- Y** Specifies that **EOF** has been encountered.
- N** Specifies that **EOF** has not been encountered.

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

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 **INDEXEXIT** parameter.

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

Figure 17 contains a sample DSECT that describes the control block that is passed to the OS/390, VM, or VSE 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 17. OS/390, VM, or VSE Sample Index Record Exit DSECT

The address of the control block containing the following parameters is passed to the index record exit. For OS/390, VM, and VSE, the address is passed in a standard parameter list that is pointed to by Register 1.

WORK@ (Bytes 1–4)

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

PFATTR@ (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 102 for more information about 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 is to be processed. If you want to ignore the record, change the **REQUEST** byte value to X'01'. Any value greater than X'01' is interpreted as X'00'; 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 one-byte character code that signals when ACIF has finished processing the index object file.

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

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

This end-of-file flag, used as a last call indicator, lets the exit program 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 **OUTEXIT** parameter.

The exit receives control before a record 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 18 on page 99 contains a sample DSECT that describes the control block that is passed to the OS/390, VM, or VSE exit program.

PARMLIST	DSECT		Parameters for the output record exit
WORK@	DS	A	Address of 16-byte static work area
PFATTR@	DS	A	Address of print-file-attribute information
RECORD@	DS	A	Address of the record to be written
RECORDLN	DS	H	Length of the output record
REQUEST	DS	X	Delete or process the record
EOF	DS	C	Last call indicator

Figure 18. OS/390, VM, or VSE Sample Output Record Exit DSECT

The address of the control block containing the following parameters is passed to the output record exit. For OS/390, VM, and VSE, the address is passed in a standard parameter list that is pointed to by Register 1.

WORK@ (Bytes 1–4)

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

PFATTR@ (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 102 for more information about 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 is to be processed. If you want to ignore the record, change the **REQUEST** byte value to X'00'. Any value greater than X'00' 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 one-byte character code that signals when ACIF has finished writing the output file.

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

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

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

Resource exit

ACIF provides an exit that lets you “filter” resources from being included in the resource file. If you want to exclude a specific type of resource (for example, an overlay), you can control this with the **RESTYPE** parameter. This exit is useful in controlling resources at the file name level. For example, assume you were going to send the output of ACIF to PSF and you only wanted to send those fonts that were not shipped with the PSF product. You could code this exit program to contain a table of all fonts shipped with PSF 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 **RESEXIT** parameter.

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

Figure 19 contains a sample DSECT that describes the control block that is passed to the OS/390, VM, or VSE 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 19. OS/390, VM, or VSE Sample Resource Exit DSECT

The address of the control block containing the following parameters is passed to the resource exit. For OS/390, VM, and VSE, the address is passed in a standard parameter list that is pointed to by Register 1.

WORK@ (Bytes 1–4)

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

PFATTR@ (Bytes 5–8)

A pointer to the print file attribute data structure. See “Attributes of the input file” on page 102 for more information about 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 one-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'42' Specifies a coded font.
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 **MCF2REF=CF** is specified, coded fonts are included in the resource file. Otherwise, ACIF does not include any referenced coded fonts in the resource file; therefore, resource filtering is not applicable. ACIF needs to process coded fonts to determine the names of the code pages and font character sets they reference, which is necessary to create MCF-2 structured fields.

- COM setup files

A COM setup file (**setup**) is a required resource for processing microfilm files (*microfilm* can mean either microfiche or 16 mm film). If you do not want a setup file included in the resource file, specify **RESTYPE=NONE** or explicitly exclude it from the **RESTYPE** list.

REQUEST (Byte 18)

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 is to be processed. If you want to ignore the resource, change the **REQUEST** byte value to X'01'. Any value greater than X'01' is interpreted as X'00' and the exit processes the resource.

EOF (Byte 19)

An end-of-file (EOF) indicator. This indicator is a one-byte character code that signals when ACIF has finished writing the output file.

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

Y Specifies that the last record has been written.

N Specifies that the last record has not been written.

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

User exit search order

When ACIF loads a specified user exit program during initialization, the OS/390, VM, or VSE operating system determines the search order and method used to locate these load modules.

OS/390	Exit load modules can reside in a load library that is used as STEPLIB, JOBLIB, or in a system library. ACIF uses the standard OS/390 search order to locate the exit load module; that is, it looks first in the STEPLIB, then in the JOBLIB, and finally in the system libraries.
VM	ACIF uses standard CMS search order to locate the specified user exit load module; that is, <i>name.TEXT</i> or <i>name.TEXTLIB</i> .
VSE	Exit load modules are located in the library that is defined by the <code>//LIBDEF PHASE.SEARCH=(...) JCL</code> statement.

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 file

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

Figure 20 shows a sample DSECT that describes the format of the OS/390, VM, or VSE data structure.

PFATTR	DSECT		Print File Attributes
CC	DS	CL3	Carriage controls? - 'YES' or 'NO '
CCTYPE	DS	CL1	Carriage control type - A (ANSI) or M (Machine)
CHARS	DS	CL20	CHARS values, including commas (eg. GT12,GT15)
FORMDEF	DS	CL8	Form Definition (FORMDEF)
PAGEDEF	DS	CL8	Page Definition (PAGEDEF)
PRMODE	DS	CL8	Processing mode
TRC	DS	CL3	Table Reference Characters - 'YES' or 'NO '

Figure 20. OS/390, VM, or VSE Sample Print File Attributes DSECT

The address of the control block containing the following parameters is passed to the input record exit. For OS/390, VM, and VSE, the address is passed in a standard parameter list that is pointed to by Register 1.

CC (Bytes 1–3)

The value of the **CC** parameter as specified to ACIF. ACIF uses the default value if this parameter is not explicitly specified.

CCTYPE (Byte 4)

The value of the **CCTYPE** parameter as specified to ACIF. 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 ACIF, 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 ACIF. 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 ACIF. 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 ACIF. 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 ACIF. ACIF uses the default value if this parameter is not explicitly specified.

Notes:

1. Each of the previous character values is left-justified, with padding blanks added to the right-end of the string. For example, if 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. Hints and tips

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
- 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
 - Download for z/OS feature
- Invoke Medium Map (IMM) structured field
- Indexing considerations
- Concatenating the resource file and the document.
- Concatenating resources to an AFP file
- Running ACIF with inline resources
- Writing inline resources to the output file
- Specifying the IMAGEOUT parameter

Working with control statements that contain numbered lines

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 messages APK451S and APK417I with a numeric value.) To resolve this problem, remove any line numbers from the control statements and rerun the job, or use a comment indicator ("/*") before each line number.

Understanding how ACIF processes unbounded box fonts (3800)

When ACIF reads a font object (coded font, character set) from a library, it searches for the bounded box version of the font (X0, C0 prefix), regardless of the specifications in the page definition or document MCF record. ACIF does not know what device might be displaying or printing the document. ACIF also does not know whether the correct version of fonts in the library is bounded-box or unbounded-box. The reason ACIF searches the bounded box version is that all printer models except the d/t3800 accept bounded-box fonts. The resulting print file will print correctly, regardless of whether the document contains bounded or unbounded font names, because PSF knows the destination printer type and uses the corresponding fonts.

You should be aware that if only unbounded-box fonts are present in the libraries defined for FONTLIB and USERLIB, ACIF will be unable to locate a bounded-box equivalent, in which case, ACIF issues MSGAPK413 and terminates the job.

Therefore, use a bounded-box font library if it is available. If, however, a bounded-box font library is not available, you can allow ACIF to find the correct font and build the output page correctly, but prevent an attempt to access C0 character sets by doing the following:

- Make copies of the unbounded-box CODED fonts
- Rename the copies to begin with X0
- Omit the collection of fonts from the **RESTYPE** parameter. (Do *not* specify **RESTYPE=ALL** or **RESTYPE=FONT**.)

Note: The archiving of unbounded-box fonts is *not* recommended. No current or future MO:DCA receiver (printer or viewer, for example) will support unbounded-box fonts.

Placing TLEs in named groups

To avoid having your job terminated by ACIF, IBM recommends that you place page-level TLEs inside named groups, using one named group per page.

You should be aware that if you specify INDEXOBJ=ALL and the input data set contains composed (AFP data stream) pages, page-level TLEs (TLE records after the AEG), and no named groups (BNG/ENG), your job may end with error message 410 or 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.

About ANSI and machine carriage controls

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:

Table 3. Standard ANSI Characters

ANSI Character	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:

Table 4. Typical Machine Controls

Machine Character	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.

Common methods of transferring files

Note: Although the information in this section is provided for the ACIF feature, the information can be used in general whenever transferring files from a z/OS or OS/390 system to a workstation. This information is important if you plan to process z/OS or OS/390 files with the graphical indexer. The graphical indexer is part of the OnDemand administrative client that runs under Windows.

You can transfer files from a z/OS or OS/390 system to a workstation by 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 a z/OS or OS/390 system to a workstation and produce output that ACIF can use are:

- Physical media (such as tape)
- PC file transfer program
- FTP
- Download for z/OS feature

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 a z/OS or OS/390 system to a workstation by using an implementation of the most common PC file transfer program (IND\$FILE). The variety of possible parameters that can affect printing are host-dependent. IBM recommends the following:

- For OS/390, the default is binary.
- 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 ACIF 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. Because ACIF examines the beginning of the file to determine whether to look for ASCII or EBCDIC newline characters, you must “trick” ACIF by using a prefix of X'0320202020200A'. ACIF will see that the first record is ASCII and therefore look for ASCII newlines and carriage returns.

FTP

From most systems, FTP works similarly to PC file transfer, and most of the same options are provided. Also, when executing FTP on a workstation, you can omit the extraneous carriage return. However, you must test and check your implementation; some FTPs use IMAGE as a synonym for BINARY.

Download for z/OS feature

You can use the Download for z/OS feature to transmit a print data set from the JES spool to a file system on a workstation. The z/OS 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 the workstation. You can use other JCL parameters, such as FORMS and DATASET to determine the application group and application to load. The Download for z/OS feature transmits data in binary format.

JES can be configured to truncate trailing blank characters (X'40') at the end of records. 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 z/OS 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. When the file is placed on the Spool, JES 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.¹ The best solution is to either NFS-mount the file, or write a small filter program on the host system that appends the two-byte record length to each record and transfer the file binary.

Generally, NFS-mounted files are not translated. However, NFS includes a two-byte binary record length as a prefix for variable-length records. (Check your NFS implementation; you may have to use special parameters.)

Note: Some NFS systems do not supply the binary record length for fixed-length files.

ACIF treats a file that contains only structured fields (MO:DCA or AFP data stream 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.

1. 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.

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. **Note:** Using machine carriage control characters for triggers is not recommended.
- If your input file contains application-generated separator pages (for example, banner pages), and you want to use data values for your indexing attributes, you can write an Input Data exit program to remove the separator pages. Otherwise, the presence of those pages in the file 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.

Concatenating resources to an AFP file

A resource group can be created and stored in a file by using the ACIF program. The resource file and the AFP file can then be concatenated together to form a file that can be processed by the indexing program.

The following example lists the parameters that are used to create a resource file using the ACIF program. The parameters process an AFP file named ARS.CREDIT.AFP. The AFP file contains no indexing information or inline resources. For this example, the output file and the index file that ACIF usually generates are not needed; all resources are assumed to be in the directory named by the USERLIB parameter.

Contents of the ACIF parameter file ARS.PARMS.ACIF(CREDIT):

```
CC=YES
CCTYPE=A
RESTYPE=OVLY,PSEG,FEDF
INPUTDD=ARS.CREDIT.AFP
OUTPUTDD=DUMMY1
INDEXDD=DUMMY2
RESOBJDD=ARS.CREDIT.RES
USERLIB=ARS.PSF.USERLIB
```

You can use the IEBGENER utility to concatenate the ARS.CREDIT.RES and ARS.CREDIT.AFP together to create an ARS.CREDIT.OUT data set. You can then process the ARS.CREDIT.OUT data set with the indexing program, using indexing information that you specify in a ARS.CREDIT.IND file.

Running ACIF with inline resources

To successfully process an input file that contains inline resources, the inline resources must be included in the input file in the order in which they are used, or the ACIF parameter EXTENSIONS = RESORDER must be specified. If a resource references another resource, the referenced resource must be included inline before the resource that references it. For example, if an overlay references a coded font that consists of the character set C0D0GT18 and code page T1D0BASE, the inline resources must be in this order:

```
code page T1D0BASE
character set C0D0GT18
coded font
overlay
```

ACIF does not look ahead in the inline resources, so, if the inline resources are not in the correct order, ACIF tries to read the referenced resource from a resource library. If the resource is not found, ACIF ends processing with an error.

Writing inline resources to the output file

When you are indexing and writing inline resources to the output document file, the offsets in the index object file are the same as if you are doing regular resource collection to a resource file. This is because the offsets are calculated from the Begin Document (BDT) structured field, not from the beginning of the output

document file. The offset from the BDT structured field to the indexed data is the same regardless of whether resources precede it.

Specifying the **IMAGEOUT** parameter

ACIF converts IM1 format images in the input file, in overlays, and in page segments to uncompressed IOCA format, if **IMAGEOUT=IOCA** (the default) is specified. An uncompressed IOCA image 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 Division 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 7. 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, two reserved bytes (X'0000'), and the tag value.

The following is an example of a 39-byte TLE structured field containing two index values. For the purposes of illustration, each field within the structured field is listed on a separate line. X' ' denotes hexadecimal data, and " " denotes EBCDIC or character data.

```
X'5A0026D3A090000000'  
X'11020B00'  
"Customer Name"  
X'0D360000'  
"Bob Smith"
```

TLE structured fields can be associated with a group of pages or with individual pages. Consider a bank statement application. Each bank statement is a group of pages, and you 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.

Understanding how ACIF processes fully composed AFP files

Fully composed AFP files contain BNG and TLE Structured Fields in the following form:

```
BDT
  BNG
    TLE (group)
    ...
    ...
    BPG
      TLE (page - optional)
      ...
      ...
    EPG
  ENG
  ...
  ...
EDT
```

When an input file contains BNG - ENG pairs or TLE Structured Fields, ACIF does not index the file. If you specify indexing parameters (such as TRIGGER, FIELD, or INDEX) for a file that contains TLE Structured Fields, then ACIF will fail with error message 462 - A trigger parameter was specified, but the input file is already indexed. If you specify indexing parameters for a file that contains BNG - ENG pairs, but does not contain TLE Structured Fields, ACIF will fail with error message 459 -Index needed for the groupname was not found.

ACIF processes a file containing BNG - ENG pairs and TLE Structured Fields in the following way:

1. For every BNG in the input, ACIF creates a group IEL Structured Field in the Index File.
2. ACIF makes a copy of the TLE Structured Fields from the input and places them into the Index File. The original TLE Structured Fields remain in the input file.

Therefore, the result of ACIF processing under these circumstances is the creation of an Index File. ACIF can complete normally but the load process into OnDemand may still fail if the format of the input file is incorrect:

- If the input file contained BNG - ENG pairs with no group level TLE Structured Fields between them, then the load process will fail with the message: 0 fields submitted, n expected, where n is the number of fields defined to OnDemand.
- If the input file does not contain any BNG - ENG pairs, then the load process may run out of memory looking for the start and end of the groups.

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 8. 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 let 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 ACIF command reference, beginning on Chapter 3, "ACIF parameter reference," on page 49. 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 9. Format of the ACIF output document file

This chapter contains General-use Programming Interface and Associated Guidance Information.

ACIF can create three separate output files, one of which is the print file in AFP data stream format. In doing so, ACIF may create the following structured fields:

- Tag Logical Element (TLE)
- Begin Named Group (BNG)
- End Named Group (ENG)

The TLE was described in Chapter 8, “Format of the ACIF index object file,” on page 117; 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 21. Example of Code Containing Group-Level Indexing

Figure 21 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 22. Example of Code Containing Group- and Page-Level Indexing

Figure 22 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 ACIF command reference, beginning on page Chapter 3, "ACIF parameter reference," on page 49. 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 eight-byte rolling sequence number to ensure uniqueness in the name. If no FQN triplet exists, ACIF creates one. Here too, ACIF appends a rolling, eight-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 eight-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 8, “Format of the ACIF index object file,” on page 117, 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 eight-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 eight-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 that ACIF must make to an AFP input file to support MO:DCA-P output format.

Composed Text Control (CTC) structured field

Because this structured field has been declared obsolete, ACIF ignores it and does not pass it to the output file.

Map Coded Font (MCF) Format 1 structured field

ACIF converts this structured field to an MCF Format 2 structured field. Unless **MCF2REF=CF** is specified, ACIF resolves the coded font into the appropriate font character set and code page pairs.

Map Coded Font (MCF) Format 2 structured field

ACIF does not modify this structured field, and it does **not** map any referenced GRID values to the appropriate font character set and code page pairs. This 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 10. Using ACIF in z/OS

Note: Statements in this chapter about ACIF in z/OS apply equally to ACIF in the OS/390 environment.

This chapter provides information about using ACIF in the z/OS environment, outside of the OnDemand application. You can run ACIF on any z/OS system on which the ACIF programs are installed. To use ACIF on a z/OS system requires:

- OS/390 Version 2 Release 8 or later
- OnDemand Version 7.1 or later

Note for Infoprint and PSF customers: The OnDemand product includes an enhanced version of the ACIF software. The enhanced version of the ACIF software provides additional function that is not available with the version of ACIF that is included with Infoprint and PSF. IBM recommends that you process your reports with the enhanced version of ACIF that is provided with OnDemand.

Sample JCL

Figure 23 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 23. Sample JCL to Invoke ACIF

About the JCL statements

The JCL statements in Figure 23 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 3 MB, 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 23 on page 127, 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 23 on page 127 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, then 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 z/OS, 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

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 parameters

Figure 24 on page 130 shows an example of JCL and ACIF processing parameters used to invoke the ACIF program to index an input file.

```

//job... JOB ...
//APKSMIN 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 24. Example of an ACIF Application

z/OS libraries

The example ACIF parameters defined the following libraries:

Library Name	z/OS Name
FDEFLIB Form definition library	SYS1.FDEFLIB
FONTLIB Font libraries	SYS1.FONTLIBB SYS1.FONTLIBB.EXTRA
OVLVLIB 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	z/OS Name
Document file, including indexing structured fields	APKACIF.OUTPUT
Index object file	APKACIF.INDEX
Resource file	APKACIF.RESLIB
Message file listing: <ul style="list-style-type: none">• ACIF parameters used• Resources used• Return code	APKACIF.SYSPRINT

Concatenating files

Before the OnDemand data loading programs can process the files created by ACIF, you must concatenate the index object file and the resource file to the document file, creating a single input file for OnDemand.

Figure 25 shows JCL that 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 25. Example of 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. Generic indexer reference

This part of the book provides information about the OnDemand Generic indexer. You can use the Generic indexer to specify index data for other types of datasets that you want to load into the system.

Chapter 11. Overview

OnDemand provides the Generic indexer so that you can specify indexing information for input data that you cannot or do not want to index with ACIF or the PDF indexer. For example, suppose that you want to load word processing documents into the system. The documents can be stored in OnDemand in the same format in which they were created. The documents can be retrieved from OnDemand and viewed with the word processor. However, because the documents do not contain AFP data, line data, or PDF data, you cannot index them with ACIF or the PDF indexer. However, you can specify index information about the documents to the Generic indexer and load the documents into the system. Users can then search for and retrieve the documents using one of the OnDemand client programs.

To use the Generic indexer, you must specify all of the index data for each input data set that you want to store in and retrieve from OnDemand. You specify the index data in a parameter data set. The parameter data set contains the index fields, index values, and information about the input data sets or documents that you want to process. The Generic indexer retrieves the indexing information from the parameter data set and generates the index data that is loaded into the database. OnDemand creates one index record for each input data set (or document) that you specify in the parameter data set. The index record contains the index values that uniquely identify a data set or document in OnDemand.

The Generic indexer supports group-level indexes. Group indexes are stored in the database and used to search for documents. You must specify one set of group indexes for each data set or document that you want to process with the Generic indexer.

Most customers use the ARSLOAD program to load data on the system. If the input data needs to be indexed, the ARSLOAD program will call the appropriate indexing program (based on the type of input data or, for the Generic indexer, the presence of a valid parameter data set). For example, the ARSLOAD program can invoke the Generic indexer to process the parameter data set and generate the index data. The ARSLOAD program can then add the index data to the database and load the input data sets or documents that you specified in the parameter data set on to storage volumes.

Processing AFP data

You can specify a parameter data set for an input data set that contains AFP resources and documents and process them with the Generic indexer. However, when you specify the parameter data set:

- The starting location (byte offset) of the first AFP document in the input data set should always be 0 (zero), even though the actual starting location is not zero when AFP resources are contained in the input. AFP resources are always located at the beginning of an input data set. The actual starting location of the first document in the input data set is zero plus the number of bytes that comprise the resources. However, to process AFP documents with the Generic indexer, you do not need to calculate the number of bytes taken by the resources.

- The starting locations of the other documents in the input data set should be calculated using the length of and offset from the previous document in the input data set.

The Generic indexer will determine where the AFP resources end in the data set and process the documents using the offsets and lengths that you provide, relative to where the resources end.

Chapter 12. Specifying parameters

The Generic indexer requires an input data sets that you want to store in OnDemand and a parameter data set that contains the indexing information for the data sets or documents. To use the Generic indexer, you must create a parameter data set that contains the indexing information for the data sets or documents that you want to process. This section describes the parameter data set for the Generic indexer.

There are three types of statements that you can specify in a parameter data set:

- **Comments.** You can place a comment line anywhere in the parameter data set.
- **Code page.** You can specify one and only one code page line. If you specify a code page line, you must do so at the beginning of the parameter data set, before you define any of your groups.
- **Groups.** A group represents a document that you want to index. Each group contains the application group field names and their index values, the location of the document in the input data set, the number of bytes (characters) that make up the document, and the name of the input data set that contains the document.

Important:

1. The parameter names in the Generic index data set are case sensitive and must appear in upper case. For example, `GROUP_FIELD_NAME:account` is valid, while `group_field_name:account` is not.
2. When loading data using the Generic indexer, the locale must be set appropriately for the `CODEPAGE:` parameter. For example, if `CODEPAGE:273` is specified, set the `LC_ALL` environment variable to `De_DE.ISM-273` or some other locale that correctly identifies upper and lower case characters in code page 273.

CODEPAGE:

Specifies the code page of the input data. You must specify one and only one code page. The **CODEPAGE:** line must appear before any of the groups.

Important: When loading data using the Generic indexer, the locale must be set appropriately for the `CODEPAGE:` parameter. For example, if `CODEPAGE:273` is specified, set the `LC_ALL` environment variable to `De_DE.ISM-273` or some other locale that correctly identifies upper and lower case characters in code page 273.

Syntax

`CODEPAGE:cpgid`

Options and values

The character string **CODEPAGE:** identifies the line as specifying the code page of the input data. The string `cpgid` can be any valid code page, a three to five character identifier of an IBM-registered or user-defined code page. The `CODEPAGE:` parameter is required.

Example

The following illustrates how to specify a code page of 500 for the input data:

```
CODEPAGE:500
```

COMMENT:

Specifies a comment line. You can place comment lines anywhere in the parameter data set.

Syntax

COMMENT: text on a single line

Options and values

The character string **COMMENT:** identifies the line as containing a comment. Everything after the colon character to the end of the line is ignored.

Example

The following are examples of comment lines:

```
COMMENT:  
COMMENT: this is a comment
```

GROUP_FIELD_NAME:

Specifies the name of an application group field. Each group that you specify in the parameter data set must contain one **GROUP_FIELD_NAME:** line for each application group field. (The application group is where you store a data set or document in OnDemand. You specify the name of the application group to the ARSLOAD program.) OnDemand supports up to 32 fields per application group. If the field names that you specify are different than the application group field names, then you must map the field names that you specify to the application group field names on the Load Information page in the application.

Specify one pair of **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines for each application group field. For example, if the application group contains two fields, then each group that you specify in the parameter data set must contain two pairs of **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines. The following is an example of a group with two application group fields:

```
GROUP_FIELD_NAME:rdate  
GROUP_FIELD_VALUE:05/31/00  
GROUP_FIELD_NAME:studentID  
GROUP_FIELD_VALUE:0012345678
```

The group lines must appear after the **CODEPAGE:** line.

Syntax

GROUP_FIELD_NAME:applgrpFieldName

Options and values

The character string **GROUP_FIELD_NAME:** identifies the line as containing the name of an application group field. The string applgrpFieldName specifies the name of an application group field. OnDemand ignores the case of application group field names.

Example

The following shows some examples of application group field names:

```
GROUP_FIELD_NAME:rdate
GROUP_FIELD_NAME:studentID
GROUP_FIELD_NAME:account#
```

GROUP_FIELD_VALUE:

Specifies an index value for an application group field. Each group that you specify in the parameter file must contain one **GROUP_FIELD_VALUE:** line for each application group field. (The application group is where you store a data set or document in OnDemand. You specify the name of the application group to the ARSLOAD program.) OnDemand supports up to 32 fields per application group. The **GROUP_FIELD_VALUE:** line must follow the **GROUP_FIELD_NAME:** line for which you are specifying the index value.

Specify a pair of **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines for each application group field. For example, if the application group contains two fields, then each group that you specify in the parameter data set must contain two pairs of **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines. The following is an example of a group with two application group fields:

```
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:05/31/00
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
```

The group lines must appear after the **CODEPAGE:** line.

Syntax

GROUP_FIELD_VALUE:value

Options and values

The character string **GROUP_FIELD_VALUE:** identifies the line as containing an index value for an application group field. The string value specifies the actual index value for the field.

Example

The following shows some examples of index values:

```
GROUP_FIELD_VALUE:05/31/00
GROUP_FIELD_VALUE:0012345678
GROUP_FIELD_VALUE:0000-1111-2222-3333
```

GROUP_FILENAME:

The DD name for the input data set.

Note: The system does not delete the source files that are specified on the **GROUP_FILENAME:** parameters in the generic index file. The system only deletes IND, OUT, and RES files.

Each group that you specify in the parameter data set must contain one **GROUP_FILENAME:** line. The **GROUP_FILENAME:** line must follow the **GROUP FIELD NAME:** and **GROUP FIELD VALUE:** lines that comprise a group. The following is an example of a group:

```
GROUP FIELD NAME:rdate
GROUP FIELD VALUE:05/31/00
GROUP FIELD NAME:studentID
GROUP FIELD VALUE:0012345678
GROUP OFFSET:0
GROUP LENGTH:0
GROUP_FILENAME:ARS.GENERIC.OUTPUT
```

If the **GROUP_FILENAME** line is blank (null), then the Generic indexer uses the value of the **GROUP_FILENAME** line from the previous group to process the current group. In the following example, the input data for the second and third groups is retrieved from the input data set that is specified for the first group:

```
GROUP FIELD NAME:rdate
GROUP FIELD VALUE:05/31/00
GROUP FIELD NAME:studentID
GROUP FIELD VALUE:0012345678
GROUP OFFSET:0
GROUP LENGTH:8124
GROUP_FILENAME:ARS.GENERIC.OUTPUT
GROUP FIELD NAME:rdate
GROUP FIELD VALUE:06/30/00
GROUP FIELD NAME:studentID
GROUP FIELD VALUE:0012345678
GROUP OFFSET:8124
GROUP LENGTH:8124
GROUP_FILENAME:
GROUP FIELD NAME:rdate
GROUP FIELD VALUE:07/31/00
GROUP FIELD NAME:studentID
GROUP FIELD VALUE:0012345678
GROUP OFFSET:16248
GROUP LENGTH:8124
GROUP_FILENAME:
```

If the first **GROUP_FILENAME** line in the parameter data set is blank, then you must specify the name of the input data set when you run the ARSLOAD command.

The group lines must appear after the **CODEPAGE:** line.

Syntax

GROUP_FILENAME:fileName

Options and values

The character string **GROUP_FILENAME:** identifies the line as containing the input data set to process. The string dataSetName specifies the data set name of the input data set.

Example

The following are valid DD name lines:

```
GROUP_FILENAME:ARS.GENERIC.OUTPUT
GROUP_FILENAME:GENERIC.OUTPUT
GROUP_FILENAME:
```

GROUP_LENGTH:

Specifies the number of contiguous bytes (characters) that comprise the document to be indexed. Specify 0 (zero) to indicate the entire input file or the remainder of the input data set. Each group that you specify in the parameter data set must contain one **GROUP_LENGTH:** line. The **GROUP_LENGTH:** line must follow the **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines that comprise a group. For example:

```
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:05/31/00
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
GROUP_OFFSET:0
GROUP_LENGTH:0
```

The group lines must appear after the **CODEPAGE:** line.

Syntax

GROUP_LENGTH:value

Options and values

The character string **GROUP_LENGTH:** identifies the line as containing the byte count of the data to be indexed. The string value specifies the actual byte count. The default value is 0 (zero), for the entire (or remainder) of the data set.

Example

The following illustrates how to specify length values:

```
GROUP_LENGTH:0
GROUP_LENGTH:8124
```

GROUP_OFFSET:

Specifies the starting location (byte offset) into the input data set of the data to be indexed. Specify 0 (zero) for the first byte (the beginning) of the data set. (If you are processing AFP documents and resources with the Generic indexer, see “Processing AFP data” on page 135.) Each group that you specify in the parameter data set must contain one **GROUP_OFFSET:** line. The **GROUP_OFFSET:** line must follow the **GROUP_FIELD_NAME:** and **GROUP_FIELD_VALUE:** lines that comprise a group. For example:

```
GROUP_FIELD NAME:rdate
GROUP_FIELD VALUE:05/31/00
GROUP_FIELD NAME:studentID
GROUP_FIELD VALUE:0012345678
GROUP_OFFSET:0
```

The group lines must appear after the **CODEPAGE:** line.

Syntax

GROUP_OFFSET:value

Options and values

The character string **GROUP_OFFSET:** identifies the line as containing the byte offset (location) of the data to be indexed. The string value specifies the actual byte offset. Specify 0 (zero), to indicate the beginning of the data set.

Example

The following illustrates offset values for three documents from the same input data set. The documents are 8 KB in length.

```
GROUP_OFFSET:0  
GROUP_OFFSET:8124  
GROUP_OFFSET:16248
```

Chapter 13. Parameter examples

The following example shows how to specify indexing information for three groups (documents). Each document will be indexed using two fields. The input data for each document is contained in a different input data set.

```
COMMENT:
COMMENT: Generic Indexer Example 1
COMMENT: Different input file for each document
COMMENT:
COMMENT: Specify code page of the index data
CODEPAGE:500
COMMENT: Document #1
COMMENT: Index field #1
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:07/13/99
COMMENT: Index field #2
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
COMMENT: document data starts at beginning of file
GROUP_OFFSET:0
COMMENT: document data goes to end of file
GROUP_LENGTH:0
GROUP_FILENAME:ARS.GENERIC1.OUTPUT
COMMENT: Document #2
COMMENT: Index field #1
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:08/13/99
COMMENT: Index field #2
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:ARS.GENERIC2.OUTPUT
COMMENT: Document #3
COMMENT: Index field #1
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:09/13/99
COMMENT: Index field #2
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
GROUP_OFFSET:0
GROUP_LENGTH:0
GROUP_FILENAME:ARS.GENERIC3.OUTPUT
COMMENT:
COMMENT: End Generic Indexer Example 1
```

The following example shows how to specify indexing information for three groups (documents). Each document will be indexed using two fields. The input data for all of the documents is contained in the same input data set.

```
COMMENT:
COMMENT: Generic Indexer Example 2
COMMENT: One input file contains all documents
COMMENT:
COMMENT: Specify code page of the index data
CODEPAGE:500
COMMENT: Document #1
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:07/13/99
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
COMMENT: first document starts at beginning of file (byte 0)
GROUP_OFFSET:0
COMMENT: document length 8124 bytes
GROUP_LENGTH:8124
GROUP_FILENAME:ARS.ACCT.STUDENT.INF.LOAD.OUTPUT
COMMENT: Document #2
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:08/13/99
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
COMMENT: second document starts at byte 8124
GROUP_OFFSET:8124
COMMENT: document length 8124 bytes
GROUP_LENGTH:8124
COMMENT: use prior GROUP_FILENAME:
GROUP_FILENAME:
COMMENT: Document #3
GROUP_FIELD_NAME:rdate
GROUP_FIELD_VALUE:09/13/99
GROUP_FIELD_NAME:studentID
GROUP_FIELD_VALUE:0012345678
COMMENT: third document starts at byte 16248
GROUP_OFFSET:16248
COMMENT: document length 8124 bytes
GROUP_LENGTH:8124
COMMENT: use prior GROUP_FILENAME:
GROUP_FILENAME:
COMMENT:
COMMENT: End Generic Indexer Example 2
```

Part 3. PDF indexer reference

This part of the book provides information about the OnDemand PDF indexer. You can use the PDF indexer to extract index data from and generate index data about Adobe PDF datasets that you want to load into the system.

The PDF indexer is included with the base code. However, customers are not authorized to install or use the PDF indexer unless they first purchase the PDF indexer feature of OnDemand for z/OS.

Chapter 14. Overview

What is the PDF indexer?

The OnDemand PDF indexer is a program that you can use to extract index data from and generate index data about Adobe PDF data sets. The index data can enhance your ability to store, retrieve, and view documents with OnDemand. The PDF indexer supports PDF Version 1.3 input and output data streams. For more information about the PDF data stream, see 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 Adobe Capture, PDFWriter, Distiller, and Exchange.

You process PDF data sets on the server using standard OnDemand functions. To process a data set, you must define an OnDemand application and application group. As part of the application, you must define the indexing parameters that are used by the PDF indexer to process the data sets.

After you use the PDF indexer and the ARSLOAD program to index and load the data sets into the system, you can do the following:

- View the PDF document or documents created during the indexing and loading process.
- Print pages of the PDF document that you are viewing from the OnDemand client program.

Figure 26 on page 148 illustrates the process of indexing and loading PDF data sets.

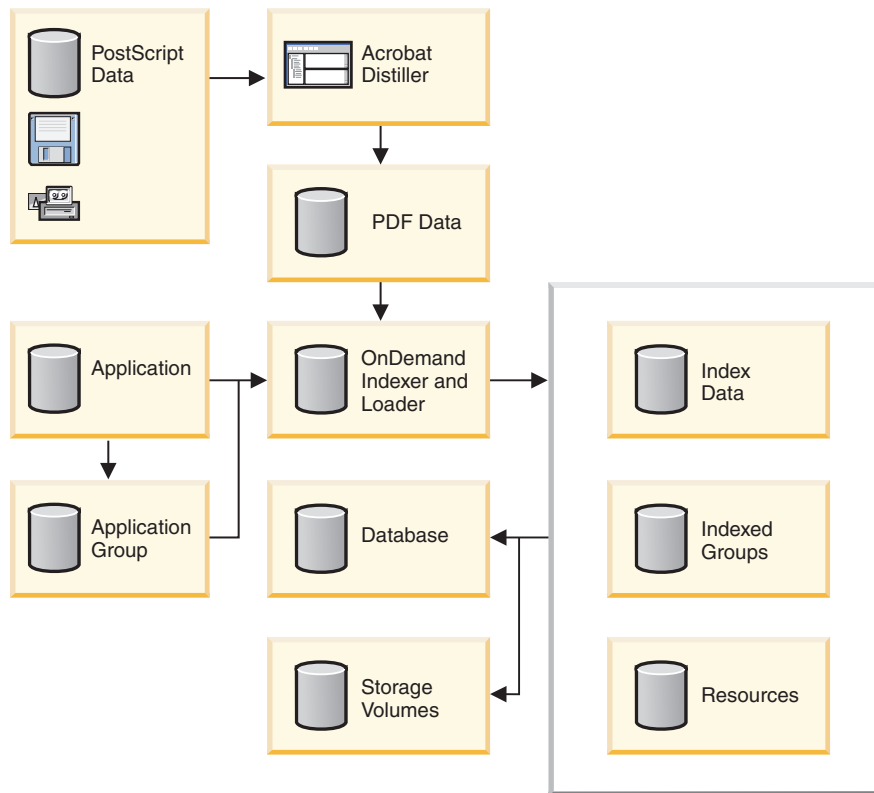


Figure 26. Processing PDF data in OnDemand

The PDF indexer processes PDF data sets. A PDF data set is a distilled version of a PostScript data set, adding structure and efficiency. A PDF data set can be created by Acrobat Distiller. See the documentation provided with Acrobat Distiller for more information about preparing input data for the Distiller.

The ARSLOAD program retrieves processing information from application and application group definitions that are stored in the database. The application definition identifies the type of input data, the indexing program used to index the input data sets, 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 the ARSLOAD program processes a PDF data set and the Indexer that is specified on the Indexing Information page in the application is PDF, the ARSLOAD program automatically calls the PDF indexer to process the data set. The PDF indexer processes the PDF data set with indexing parameters that determine the location and attributes of the index data. The PDF indexer extracts index data from the PDF data set and generates an index data set and an output data set. The output data set contains groups of indexed pages. A group of indexed pages can represent the entire input data set or, more typically, one or more pages from the input data set. If the input data set 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 data set. That way, users can retrieve a specific statement or set of statements, rather than the entire data set. After indexing the data, OnDemand stores the index data in the database and the indexed groups on storage volumes.

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 load a PDF data set into OnDemand, the ARSLOAD program invokes the PDF indexer to process the indexing parameters and create the index data. The ARSLOAD program then loads the index data into the database, storing the group-level attribute values that the PDF indexing program extracted from the data into their corresponding database fields. Figure 27 illustrates the index creation and data loading process.

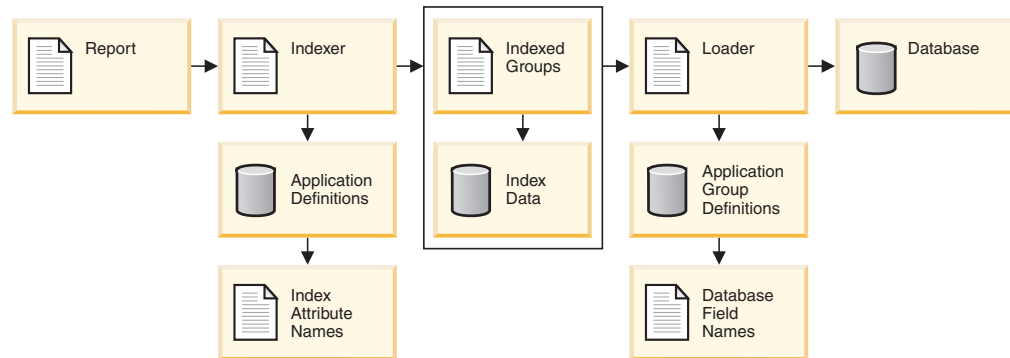


Figure 27. Indexing and loading data

You typically create an application for each report that you plan to store 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 into which 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 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 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 storage.

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 in the OnDemand Generic indexer format. See Part 2, “Generic indexer reference,” on page 133 for more information about the OnDemand Generic indexer.

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 program 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 data set that contains the information specified in the trigger, it can begin to look for index information.

- The PDF indexer compares words in the input data set 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 sixteen 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 the sixteen 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. IBM strongly recommends that you 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 28 on page 152 depicts a portion of a page from a sample input data set. The text strings that determine the beginning of a group and the index values are enclosed in rectangles.

[illegible]

Figure 28. Indexing data with the PDF indexer

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 data set. 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 28. See Chapter 16, “Parameter reference,” on page 161 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, you might define four fields:

- FIELD1 identifies the location of customer name index values.

```
FIELD1=u1(1,1),lr(3.25,1.25),0
```

- FIELD2 identifies the location of statement date index values.

```
FIELD2=u1(2,2),lr(2.75,2.25),0
```

- FIELD3 identifies the location of account number index values.

```
FIELD3=u1(2,2.25),1r(3.25,2.5),0
```

- FIELD4 identifies the location of the balance index values.

FIELD4=u1(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 to 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 that the PDF indexer uses to identify groups and locate index data. Then, create the indexing parameters using the administrative client.

OnDemand provides the ARSPDUMP program to help you determine the location of trigger and field string values in the input data. The ARSPDUMP program processes one or more pages of sample report data and generates an output data set. The output data set 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 data set, beginning with the number 1 (one), for the first page in the data set. 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 program. See Chapter 19, “Using ARSPDUMP in z/OS,” on page 177 for examples.
- In the output data set, 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 that you can use to maintain indexing parameters for the application.

Processing PDF input files with the graphical indexer

This section describes how to use the graphical indexer to create indexing information for PDF input data sets.

Important: If you plan to use the report wizard or the graphical indexer to process PDF input data sets, then you must first install Adobe Acrobat on the PC from which you plan to run the administrative client. OnDemand provides the ARSPDF32.API file to enable PDF viewing from the client.

If you install the client after you install Adobe Acrobat, then the installation program will copy the API file to the Acrobat plug-in directory. If you install the client before you install Adobe Acrobat, then you must copy the API file to the Acrobat plug-in directory. Also, if you upgrade to a new version of Acrobat, then you must copy the API file to the new Acrobat plug-in directory. The default location of the API file is \Program Files\IBM\OnDemand32\PDF. The default Acrobat plug-in directory is \Program Files\Adobe\Acrobat x.y\Acrobat\Plug_ins, where x.y is the version of Acrobat, for example, 4.0, 5.0, and so forth.

You can define indexing information in a visual environment. You begin by opening a sample input file with the graphical indexer. You can run the graphical indexer from the report wizard or by choosing the sample data option from the Indexing Information page of the application. After you open an input file in the graphical indexer, you define triggers, fields, and indexes. The PDF indexer uses the triggers, fields, and indexes to locate the beginning of a document in the input data and extract index values from the input data. Once you have defined the triggers, fields, and indexes, you can save them in the application so that OnDemand can use them later on to process the input files that you load into the system.

You define a trigger, field, or index by drawing a box around a text string with the mouse and then specifying properties. For example, to define a trigger that identifies the beginning of a document, you could draw a box around the text string Account Number on the first page of a statement in the input file. Then, on the Add a Trigger dialog box, you would accept the default values provided, such as the location of the text string on the page. When processing an input file, the PDF indexer attempts to locate the specified string in the specified location. When a match occurs, the PDF indexer knows that it has found the beginning of a document. The fields and indexes are based on the location of the trigger.

The PDF file that you open with the graphical indexer should contain a representative sample of the type of input data that you plan to load into the system. For example, the sample input file must contain at least one document. A good sample should contain several documents so that you can verify the location of the triggers, fields, and indexes on more than one document. The sample input file must contain the information that you need to identify the beginning of a document in the input file. The sample input file should also contain the information that you need to define the indexes. When you load an input file into the system, the PDF indexer will use the indexing information that you create to locate and extract index values for each document in the input file.

The following example describes how to use the graphical indexer from the report wizard to create indexing information for an input file. The indexing information consists of a trigger that uniquely identifies the beginning of a document in the input file and the fields and indexes for each document.

1. To begin, start the administrative client.
2. Log on to a server.
3. Start the report wizard by clicking the Report Wizard icon on the toolbar. The report wizard opens the Sample Data dialog box.
4. Click **Select Sample Data** to open the Open dialog box.
5. Type the name or full path name of a file in the space provided or use the Look in or Browse commands to locate a file.

6. Click **Open**. The graphical indexer opens the input file in the report window.
7. Press F1 to open the main help topic for the report window. The main help topic contains general information about the report window and contains links to other topics that describe how to add triggers, fields, and indexes. Under Options and Commands, click Indexer Information page to open the Indexing Commands topic. (You can also use the content help tool to display information about the icons on the toolbar.) Under Tasks, Indexer Information page, click Adding a trigger (PDF).
8. Close any open help topics and return to the report window.
9. Define a trigger.
 - Find a text string that uniquely identifies the beginning of a document. For example, Account Number, Invoice Number, Customer Name, and so forth.
 - Using the mouse, draw a box around the text string. Start just outside of the upper left corner of the string. Click and hold using your mouse. Drag the mouse towards the lower right corner of the string. As you drag the mouse, the graphical indexer uses a dotted line to draw a box. When you have enclosed the text string completely inside of a box, release the mouse button. The graphical indexer highlights the text string inside of a box.
Tip: If you are having difficulty selecting text strings, try including as much white space as possible to get the highlighting to stick. Also, try magnifying the text string area when selecting it.
 - Click the **Define a Trigger** icon on the toolbar to open the Add a Trigger dialog box. If the Trigger icon is greyed out, make sure you have highlighted a text string. You cannot use Trigger until you highlight a text string. In the Add a Trigger dialog box, verify the attributes of the trigger. For example, the text string that you selected in the report window should be displayed under Value; for Trigger1, the Pages to Search should be set to Every Page. Click **Help** for assistance with the other options and values that you can specify.
 - Click **OK** to define the trigger. The Index icon becomes available for you to use after you define the trigger.
 - To verify that the trigger uniquely identifies the beginning of a document, first put the report window in display mode. Then click the Select tool to open the Select dialog box. Under Triggers, double click the trigger. The graphical indexer highlights the text string in the current document. Double click the trigger again. The graphical indexer should highlight the text string on the first page of the next document. Use the Select dialog box to move forward to the first page of each document and return to the first document in the input file.
 - Put the report window in add mode.
10. Define a field and an index.
 - Find a text string that can be used to identify the location of the field. The text string should contain a sample index value. For example, if you want to extract account number values from the input file, then find where the account number is printed on the page.
 - Using the mouse, draw a box around the text string. Start just outside of the upper left corner of the string. Click and hold using your mouse. Drag the mouse towards the lower right corner of the string. As you drag the mouse, the graphical indexer uses a dotted line to draw a box. When you have enclosed the text string completely inside of a box, release the mouse button. The graphical indexer highlights the text string inside of a box. The Define a Trigger and Define a Field icon become active.

Tip: If you are having difficulty selecting text strings, try including as much white space as possible to get the highlighting to stick. Also, try magnifying the text string area when selecting it.

- Click the Define a Field icon on the toolbar to open the Add a Field dialog box.
 - On the Field Information page, verify the attributes of the index field. For example, the text string that you selected in the report window should be displayed under Reference String; the Trigger should identify the trigger on which the field is based. Click Help for assistance with the options and values that you can specify.
 - On the Database Field Attributes page, verify the attributes of the database field. In the Database Field Name space, enter the name of the application group field into which you want OnDemand to store the index value. In the Folder Field Name space, enter the name of the folder field that will appear on the client search screen. Click Help for assistance with the other options and values that you can specify.
 - Click OK to define the field and index.
 - To verify the locations of the fields, first put the report window in display mode. The fields should have a blue box drawn around them. Next, click the Select tool to open the Select dialog box. Under Fields, double-click Field 1. The graphical indexer highlights the text string in the current document. Double click Field 1 again. The graphical indexer should move to the next document and highlight the text string. Use the Select dialog box to move forward to the each document and display the field. Then return to the first document in the input file.
 - Put the report window in add mode.
11. Click the Display Indexer Parameters tool to open the Display Indexer Parameters dialog box. The Display Indexer Parameters dialog box lists the indexing parameters that the PDF indexer will use to process the input files that you load into the application. At a minimum, you need one trigger, one field, and one index. See theChapter 16, "Parameter reference," on page 161 for details about the indexing parameters.
 12. When you have finished defining all of the triggers, fields, and indexes, close the report window.
 13. Click **Yes** to save the changes to the indexer parameters.
 14. On the Sample Data window, click **Next** to continue with the report wizard.

After you create your indexing parameter, transfer a sample PDF file to the host using the same parameters that you just defined in the previous steps. You should FTP the file in binary to an MVS data set, not to HFS. Because you want to use the PDF Indexer, you need to transfer the data to an MVS data set and then use the `-s` parameter for the DDNAME to send to the indexer in ARSLOAD.

Review the *IBM Content Manager OnDemand for z/OS and OS/390 Administration Guide Version 7.1* (publication SC27-1374) for ARSLOAD parameters. **Important:** The PDF Indexer also requires the ADOBERES, ADOBEFNT, TEMPATTR, and SYSTMP01 DDs be specified in the ARSLOAD job. See the "PDF Indexer" section in *IBM Content Manager OnDemand for z/OS and OS/390 Configuration Guide Version 7.1* (publication GC27-1373) for more information about these DDs. Be aware that the TEMPATTR DD DSN should be `1recl=255, recfm=VB, and blocksize=27998`. The file should contain the following line:

```
unit=vio,cyl,spaceround,primary=5,secondary=5,recfm=f,1recl=13030,blksize=13030
```

For more details on specifying the required DDs for the PDF Indexer, refer to the OnDemand guides at: <http://www-3.ibm.com/software/data/ondemand/390/library.html> Also, review the following Redbook for additional assistance:
<http://www.redbooks.ibm.com/redbooks/pdfs/sg246915.pdf>.

Troubleshooting

If you have set up your data sets for the PDF Indexer and still experience errors when running ARSLOAD, then check for the following settings:

- Make sure you specify NUM OFF in the PDFs you create for the Adobe DDs. Ensure that the members do not contain line numbering in columns 72-80. Having line numbering in columns 72-80 causes errors about an invalid DSN format.
- The text in the ADOBERES DD library member should be in mixed mode. You should set CAPS OFF in the member and follow the samples in Chapter 18, “Using ARSPDOCI in z/OS,” on page 173 to specify the values. For instance, specify PS-Resources-1.0 rather than PS-RESOURCES-1.0.
- Ensure you transferred the sample PDF file to the host in binary. If you did not, you might see errors indicating the ACRODIST module was not found (S806).
- As a general practice, ensure you have the latest maintenance available for ARSPDOCI and ARSLOAD, including the latest builds of the Adobe libraries available and correct known issues.
- Open the PDF file you are attempting to index in Adobe Acrobat or Acrobat Reader and scroll through the pages. **Note:** If you experience difficulties viewing the document in the viewer on the PC, then the problem is most likely located with the PDF structure, and therefore, the Indexer is also likely to experience problems with this file.

Chapter 15. System considerations

System requirements

The OnDemand PDF indexer is included with the base code. However, customers are not authorized to install or use the OnDemand PDF indexer unless they first purchase the PDF indexer feature of OnDemand for z/OS and OS/390.

To process PDF data with the administrative client or to view the PDF documents that you load into the system, you must obtain Adobe Acrobat PDF viewing software from Adobe. IBM recommends that you obtain one copy of Adobe Acrobat for each user that needs to process or view PDF documents. See <http://www.adobe.com> to find out more about Adobe Acrobat.

System limitations

If you are using the OnDemand PDF indexer to generate index data for PDF data sets that are created by user-defined programs, then you need to keep the following in mind:

- The PDF indexer can process data sets that are up to 4 GB in size.
- The PDF indexer supports DBCS languages. However, IBM does not provide any DBCS fonts. You can purchase DBCS fonts from Adobe. The PDF indexer supports all DBCS fonts, except encrypted Japanese fonts.
- Input data delimited with PostScript Passthrough markers cannot be indexed
- The Adobe Toolkit does not validate link destinations or bookmarks to other pages in a document or to other documents. Links or bookmarks may or may not resolve correctly, depending on how you segment your documents.
- If a font is referenced but not embedded in a PDF data set, the Adobe Acrobat viewing software attempts to find the font using information contained in the PDF font descriptor. If the Acrobat viewing software finds the font, then it uses the font to display the text. If the Acrobat viewing software does not find the font, then 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. (IBM does not provide Acrobat Distiller.) The documentation provided with Acrobat Distiller describes methods that you can use to generate PDF data.

If you plan to automate the data indexing and loading process on the OnDemand server, then the input data set name must identify the application group and application to load. Use the following convention to name your input data sets:

```
MVS.JOBNAME.DATASET.FORMS.YYDD.HHMMSS.PDF
```

By default, the ARSLOAD program uses the FORMS part of the data set name to identify the application group to load. You can use the -G parameter to specify a different part of the name that identifies the application group. For example, `arsload -G JOBNAME`. If the application group contains more than one application, then you must identify the application to load. Otherwise the load will fail. For example, to use the DATASET part of the data set name to identify the application,

run the ARSLOAD program with the -A DATASET parameter. Choose one of the MVS, JOBNAME, DATASET, and FORMS parts of the data set name to identify the application group and application.

Note: The case of the identifier PDF is ignored. However, application group and application names are case sensitive and may include special characters such as the blank character.

NLS considerations

The PDF indexer supports DBCS languages. However, IBM does not provide any DBCS fonts. You can purchase DBCS fonts from Adobe. The PDF indexer supports all DBCS fonts, except encrypted Japanese fonts.

Data values that you specify 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, then any data values that you specify on TRIGGER and FIELD parameters must be encoded in code page 500. Examples of data values that you might specify include TRIGGER string values and FIELD default and constant values.

Important: When loading data using the PDF indexer, the locale must be set appropriately for the code page of the documents. For example, if the code page of the documents is 273, set the LC_ALL environment variable to De_DE.IBM-273 or some other locale that correctly identifies upper and lower case characters in code page 273.

For more information about NLS in OnDemand, see the *IBM DB2 Content Manager OnDemand for z/OS and OS/390 Configuration Guide*, GC27-1373.

Chapter 16. Parameter reference

This parameter reference assumes that you will use the OnDemand application to process your input data sets. When you use the OnDemand application to process input data, the PDF indexer ignores any values that you may provide for the INDEXDD, INPUTDD, MSGDD, OUTPUTDD, and PARMDD parameters. If you process your input data sets outside of the OnDemand application, then you must provide values for the INPUTDD, OUTPUTDD, and PARMDD parameters and verify that the default values for the INDEXDD and MSGDD parameters are correct. If you plan to process your input data sets outside of the OnDemand application, please see Chapter 18, “Using ARSPDOCI in z/OS,” on page 173.

COORDINATES

Identifies the metrics used for *x,y* coordinates in the FIELD and TRIGGER parameters.

Required?

No

Default Value

IN

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 32 fields. You can define two types of fields: a *trigger field*, which is based on the location of a trigger string value and a *constant field*, which provides the actual index value that is 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 (zero) 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 (zero) 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, then 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='fieldMask'

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=u1(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. Specify BASE=0 because the field string box always appears in a specific location on the page.

`TRIGGER1=u1(0,0),lr(.75,.25),*, 'Page 0001'`

`FIELD1=u1(1,1),lr(3.25,1.25),0,(TRIGGER=1,BASE=0)`

Constant field syntax

FIELDn='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 (one) 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=u1(2,2),lr(2.5,2.25),0,(TRIGGER=1,BASE=0)  
INDEX1='acct_num',FIELD1,FIELD2
```

Related parameters

INDEX parameter on page 164.

TRIGGER parameter on page 168.

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, IBM strongly recommends that you name the index the same as the application group database field name.

Required?

Yes

Default Value

<none>

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 *acctNumber* 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 (one) to 250 bytes in length. IBM strongly recommends that you name the index the same as the application group database field name.

FIELDnn

The name of the field parameter or parameters that the PDF indexer uses to locate the index. You can specify a maximum of 32 field parameters. Separate the 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='reportDate',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='acctNumber',FIELD2
```

Related parameters

FIELD parameter on page 161.

INDEXDD

Specifies the file name or full path name of the index object file in the HFS. The PDF indexer writes indexing information to the index object file.

Required?

No

Note: When you process input files with the ARSLOAD program, the PDF indexer ignores any value that you may specify for the INDEXDD parameter. (The ARSLOAD program sets the file name for the PDF indexer.) If you process input files by any other method, for example, by running the ARSPDOCI program from the command line, verify the value of the INDEXDD parameter.

Default Value

INDEX

Syntax

```
INDEXDD=HFS:filename
```

Options and values

The *filename* is a valid file name or full path name in the HFS. For example:

```
INDEXDD=HFS:/ars/tmp/po301005.ind
```

Notes:

1. File names and path names are case sensitive.
2. If you do not specify the INDEXDD parameter, the PDF indexer uses INDEX as the default file name.
3. If you specify the file name without a path (or you do not specify the INDEXDD parameter), the PDF indexer writes the index object file to the current directory.

INDEXSTARTBY

Determines the page number by which the PDF indexer must locate the first group (document) within the input data set. The first group is identified when all of the triggers and fields are found. For example, with the following parameters:

```
TRIGGER1=u1(4.72,1.28),lr(5.36,1.45),*, 'ACCOUNT '  
TRIGGER2=u1(6.11,1.43),lr(6.79,1.59),1, 'SUMMARY '  
INDEX1='Account',FIELD1,FIELD2  
FIELD1=u1(6.11,1.29),lr(6.63,1.45),2  
FIELD2=u1(6.69,1.29),lr(7.04,1.45),2  
INDEX2='Total',FIELD3  
FIELD3=u1(6.11,1.43),lr(6.79,1.59),2  
INDEXSTARTBY=3
```

The word ACCOUNT must be found on a page in the location described by TRIGGER1. The word SUMMARY must be found on a page following the page on which ACCOUNT was found, in the location specified by TRIGGER2. In addition, there must be one or more words found for fields FIELD1, FIELD2, and FIELD3 in the locations specified by FIELD1, FIELD2, and FIELD3 which are located on a page that is two pages after the page on which TRIGGER1 was found.

In the example, the first group in the data set must start on either page one, page two, or page three. If TRIGGER1 is found on page one, then TRIGGER2 must be found on page two and FIELD1, FIELD2, and FIELD3 must be found on page three.

The PDF indexer stops processing if it does not locate the first group by the specified page number. This parameter is optional, but the default is that the PDF indexer must locate the first group on the first page of the input data set. This parameter is helpful if the input file contains header pages. For example, if the input data set 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 the first group 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 data set or store them in OnDemand. For example, if you specify INDEXSTARTBY=3 and the first group is found on page three, then pages one and two are not copied to the output data set or stored in OnDemand; if you specify INDEXSTARTBY=3 and the first group is found on page two, then page one is not copied to the output data set or stored in OnDemand.

Required?

No

Default Value

1

Syntax

INDEXSTARTBY=*value*

Options and values

The *value* is the page number by which the PDF indexer must locate the first group (document) in the input data set.

INPUTDD

Specifies the DD name for the input data set that the PDF indexer will process.

Required?

No

Note: When you process input data sets with the OnDemand application, the PDF indexer ignores any value that you may supply for the INPUTDD parameter. If you process input data sets outside of the OnDemand application, then you must specify a value for the INPUTDD parameter.

Default Value

<none>

Syntax

INPUTDD=*DD name*

Options and values

The *DD name* is the one- to eight-byte character DD name for the input data set.

MSGDD

Specifies the DD name for the data set where the PDF indexer writes error messages. If you do not specify the MSGDD parameter, then the PDF indexer writes messages to SYSPRINT.

Required?

No

Note: When you process input data sets with the OnDemand application, the PDF indexer ignores any value that you may supply for the MSGDD parameter. If you process input data sets outside of the OnDemand application, then verify the value of the MSGDD parameter.

Default Value

SYSPRINT

Syntax

Options and values

The *DD name* is the one- to eight-byte character DD name for the message data set.

OUTPUTDD

Specifies the file name or the full path name of the output file in the HFS.

Required?

No

Note: When you process input files with the ARSLOAD program, the PDF indexer ignores any value that you may supply for the OUTPUTDD parameter. (The ARSLOAD program sets the output file name for the PDF indexer.) If you process input files by any other method, for

example, by running the ARSPDOCI program from the command line, then you must specify a value for the OUTPUTDD parameter.

Default Value

<none>

Syntax

OUTPUTDD=HFS:*filename*

Options and values

The *filename* is the file name or full path name of the output file in the HFS. For example:

OUTPUTDD=HFS:/ars/tmp/po301005.out

Notes:

1. File names and path names are case sensitive.
2. If you specify the file name without a path, the PDF indexer writes the output file to the current directory.

PARMDD

Specifies the DD name for the data set that contains the indexing parameters that the PDF indexer uses to process the input data.

Required?

No

Note: When you process input data sets with the OnDemand application, the PDF indexer ignores any value that you may supply for the PARMDD parameter. If you process input data sets outside of the OnDemand application, then you must specify a value for the PARMDD parameter.

Default Value

<none>

Syntax

PARMDD=*DD name*

Options and values

The *DD name* is the one- to eight-byte character DD name for the parameter data set.

TRACEDD parameter

The TRACEDD parameter is new in SPE 2. For more information, see Chapter 20, "Trace facility," on page 179.

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 sixteen triggers.

Required?

Yes

Default Value
<none>

Syntax

TRIGGER n =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) to 16 (sixteen).

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 (zero) 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 (zero) to 45, page width and length, in inches.

page

The page number in the input file on which the trigger string value must be located.

- For TRIGGER1, the *page* value must be an asterisk (*), to specify that the trigger string value can be located on any page in the input file. The PDF indexer begins searching on the first page in the input file. The PDF indexer continues searching until the trigger string value is located, the INDEXSTARTBY value is reached, or the last page of the input file is searched, whichever occurs first. If the PDF indexer reaches the INDEXSTARTBY value or the last page and the trigger string value is not found, then an error occurs and indexing stops.
- For all other triggers, the *page* value can be 0 (zero) to 16, relative to TRIGGER1. For example, the page value 0 (zero) means that the trigger is located on the same page as TRIGGER1; the value 1 (one) means that the trigger is located on the page after the page that contains TRIGGER1; and so forth. For TRIGGER2 through TRIGGER16, the trigger string value can be a maximum of 16 pages from 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=u1(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, a one by four inch trigger string box is defined, 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=u1(4,4),lr(5,8),1,'Total'  
FIELD2=u1(1,0),lr(2,1),0,(TRIGGER=2,BASE=TRIGGER)
```

Related parameters

The FIELD parameter on page 161.

Chapter 17. Message reference

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**

When the PDF indexer processes a data set, 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**

The PDF indexer issues a warning message and return code of 4 (four) when the fidelity of the document may be in question.

- **Error**

The PDF indexer issues an error message and return code of 8 (eight) or 16 (sixteen) and terminates processing the current input data set. 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 to the message data set that you specified when you invoked the PDF indexer. 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 program, you can use the **MSGDD** parameter to specify the DD name for the data set to hold processing messages.

- **Adobe Toolkit**

- **Internal Error**

The PDF indexer issues an error message and return code of 16 (sixteen) and terminates processing the current input data set.

Note: Please see *IBM Content Manager OnDemand Version 7.1 Messages and Codes*, SC27-1379 for a list of the messages that can be generated by the PDF indexer, along with explanations of the messages and actions that you can take to respond to the messages. The messages that are generated by the PDF indexer are listed in the Common Server section of the *Messages and Codes* publication. You can get a copy of the *Messages and Codes* publication from <http://www.ibm.com/software/data/OnDemand/390/library.html>.

Chapter 18. Using ARSPDOCI in z/OS

This chapter provides information about using the ARSPDOCI program in the z/OS environment, outside of the OnDemand application.

Note: The statements in this chapter about using the ARSPDOCI program in z/OS apply equally to using the ARSPDOCI program in OS/390.

The ARSPDOCI program uses the identified locations of text strings on a page of a PDF document to produce a text index file as well as a byte offset indexed PDF document. You can use the ARSPDUMP program to list the locations of text strings in a document. See Chapter 19, “Using ARSPDUMP in z/OS,” on page 177 for more information.

The ARSPDOCI program requires two input files: a PDF document and a parameter file.

If a font is referenced in a PDF file, but not embedded, then the ARSPDOCI program attempts to find the font using information provided with the //ADOBERES and //ADOBEFNT DD statements. If the ARSPDOCI program does not find the font, then it uses a substitute Adobe Type 1 font.

Sample JCL

Figure 29 shows sample JCL used to run the ARSPDOCI program.

```
//MSTEPHCA JOB 12345678,
//          MSTEPHE,REGION=0M,
//          NOTIFY=&SYSUID,CLASS=A,
//          MSGLEVEL=(1,1)
//PROCESS OUTPUT DEFAULT=YES,CLASS=*,JESDS=ALL,OUTDISP=HOLD
//RUNIT EXEC PGM=ARSPDOCI,REGION=0M,
//          PARM='parmdd=DDN:PAR'
//STEPLIB DD DSN=ARS.V7R1M0.SARSL0AD,DISP=SHR
//PAR DD DSN=MSTEPHE.PAR,DISP=SHR
//ADOBERES DD DSN=ADOBE.PDFLIB.RESOURCE.INDEX(ADOBERES),DISP=SHR
//ADOBEFNT DD DSN=ADOBE.PDFP405.PLUSP1C.ADOBEFNT.LST,DISP=SHR
//TEMPATTR DD DSN=ADOBE.PDFP405F.PLUSP1C.TEMPATTR,DISP=SHR
//SYSTM01 DD UNIT=VIO,DSN=&&SYSTM1,DISP=(NEW,PASS),SPACE=(CYL,(5,5))
//SYSTEM DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 29. Sample JCL for the ARSPDOCI Program

About the JCL statements

The JCL statements in Figure 29 are explained as follows.

//RUNIT EXEC PGM=ARSPDOCI

Specifies the name of the program – ARSPDOCI.

// PARM=

Parameter values for the ARSPDOCI program. The parameter values are:

- **parmdd**

A required parameter that specifies either the DD name or the data set name of the file that contains the indexing parameters. To specify the DD name, use the following format of the parameter:

parmdd=DDN:PAR, where **PAR** is the DD name. To specify the data set name, use the following format of the parameter:

parmdd=DSN:MSTEPHE.PARM, where **MSTEPHE.PARM** is the data set name.

You must specify the **parmdd** parameter in the JCL.

“Sample parameter file” on page 175 lists the indexing parameters for this example.

- **inputdd**

An optional parameter that specifies either the DD name that contains the data set name and other attributes of the input file or the data set name of the input file. To specify the DD name, use the following format of the parameter: **inputdd=DDN:IN**, where **IN** is the DD name. To specify the data set name, use the following format of the parameter: **inputdd=DSN:MSTEPHE.PDF**, where **MSTEPHE.PDF** is the data set name.

If you do not specify the **inputdd** parameter in the JCL, then you must specify the **INPUTDD** parameter in the parameter file. If you specify both the **inputdd** parameter in the JCL and the **INPUTDD** parameter in the parameter file, the ARSPDOCI program uses the value of the **INPUTDD** parameter to determine the input data set to process.

The input file for this example is specified in the parameter file. (See “Sample parameter file” on page 175.)

- **outputdd**

An optional parameter that specifies the file name or full path name of the output file in the HFS. To specify the file name, use the following format of the parameter: **outputdd=HFS:/path/filename.out**, where **/path/filename.out** is a valid file name or full path name in the HFS.

If you do not specify the **outputdd** parameter in the JCL, then you must specify the **OUTPUTDD** parameter in the parameter file. If you specify both the **outputdd** parameter in the JCL and the **OUTPUTDD** parameter in the parameter file, the ARSPDOCI program uses the value of the **OUTPUTDD** parameter to determine the name for the output document file.

The output file for this example is specified in the parameter file. (See “Sample parameter file” on page 175.)

- **indexdd**

An optional parameter that specifies the file name or full path name of the index object file in the HFS. To specify the file name, use the following format of the parameter: **indexdd=HFS:/path/filename.ind**, where **/path/filename.ind** is a valid file name or full path name in the HFS.

If you do not specify the **indexdd** parameter in the JCL, then you must specify the **INDEXDD** parameter in the parameter file. If you specify both the **indexdd** parameter in the JCL and the **INDEXDD** parameter in the parameter file, the ARSPDOCI program uses the value of the **INDEXDD** parameter to determine the name for the index object file.

The index file for this example is specified in the parameter file. (See “Sample parameter file” on page 175.)

//PAR Specifies the data set name and other attributes of the parameter file.

//ADOBERES

Specifies the data set name of the library that contains the Adobe fonts.

//ADOBEFNT

Specifies the data set name for the list of Adobe fonts that the ARSPDUMP program found in the Adobe font library.

Sample parameter file

Note: The parameter file must be in EBCDIC for z/OS. If you use the report wizard or the edit window in the administrative client to create indexing parameters, remember to use the ASCII option when you transfer the parameter file from the PC to the z/OS system. Otherwise, the ARSPDOCI program will fail when attempting to read the parameter file.

Figure 30 shows the sample parameter file used for this example.

```
TRIGGER1=UL(4.83,1.21),LR(7.64,1.54),*, 'Page      1'
FIELD1=UL(2.57,2.06),LR(5.39,2.31),0, (TRIGGER=1, BASE=0)
INDEX1='Account Number', FIELD1, (TYPE=GROUP, BREAK=NO)
INDEXSTARTBY=1
INDEXDD=HFS:/ars/tmp/user.ind
INPUTDD=DSN:USER.PDF
OUTPUTDD=HFS:/ars/tmp/user.out
```

Figure 30. Sample Parameter File for the ARSPDOCI Program

About the parameters

The parameters in Figure 30 are explained as follows.

TRIGGER1

Identifies the location of the string value that uniquely identifies the beginning of a group. You must specify at least one TRIGGER parameter. See “TRIGGER” on page 168 for details about the TRIGGER parameter.

FIELD1

Identifies the location of the index data, based on the TRIGGER. You must specify at least one FIELD parameter. See “FIELD” on page 161 for details about the FIELD parameter.

INDEX1

Identifies the index name and the field on which the index is based. You must specify at least one INDEX parameter. See “INDEX” on page 164 for details about the INDEX parameter.

INDEXSTARTBY

Determines the page number by which the PDF indexer must locate the first group within the input data set. The first group is identified when all of the triggers and fields are found. See “INDEXSTARTBY” on page 166 for details about the INDEX parameter.

INDEXDD

Specifies the file name name for the index object file. The PDF indexer writes indexing information to the index object file. See “INDEXDD” on page 165 for details about the INDEXDD parameter.

The following shows how to specify a file name:

```
INDEXDD=HFS:/path/filename.ind
```

Where /path/filename.ind is a valid file name or full path name in the HFS.

If you do not specify the **indexdd** parameter in the JCL, then you must specify the INDEXDD parameter in the parameter file. If you specify both the **indexdd** parameter in the JCL and the INDEXDD parameter in the parameter file, the ARSPDOCI program uses the value of the INDEXDD parameter to determine the name for the index object file.

INPUTDD

Specifies the data set name of the input PDF file to process. See “INPUTDD” on page 167 for details about the INPUTDD parameter.

The following shows how to specify a data set name:

```
INPUTDD=DSN:xxxxxxx.PDF
```

Where xxxxxxx.PDF is the data set name.

If you do not specify the **inputdd** parameter in the JCL, then you must specify the INDEXDD parameter in the parameter file. If you specify both the **inputdd** parameter in the JCL and the INPUTDD parameter in the parameter file, the ARSPDOCI program uses the value of the INPUTDD parameter to determine the data set name of the input PDF file to process.

OUTPUTDD

Specifies the file name for the output file. The PDF indexer writes the indexed groups (documents) to the output file. See “OUTPUTDD” on page 167 for details about the OUTPUTDD parameter.

The following shows how to specify a file name:

```
OUTPUTDD=HFS:/path/filename.out
```

Where /path/filename.out is a valid file name or full path name in the HFS.

If you do not specify the **outputdd** parameter in the JCL, then you must specify the OUTPUTDD parameter in the parameter file. If you specify both the **outputdd** parameter in the JCL and the OUTPUTDD parameter in the parameter file, the ARSPDOCI program uses the value of the OUTPUTDD parameter to determine the name for the output document file.

Chapter 19. Using ARSPDUMP in z/OS

This chapter provides information about using the ARSPDUMP program in the z/OS environment, outside of the OnDemand application.

Note: The statements in this chapter about using the ARSPDUMP program in z/OS apply equally to using the ARSPDOCI program in OS/390.

The ARSPDUMP program lists the locations of text strings on a page in a PDF file. 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 program contains a list of the text strings on the page and the coordinates for each string. You can use the information that is generated by the ARSPDUMP program to create the parameter file that is used by the ARSPDOCI program to index PDF files. See Chapter 18, “Using ARSPDOCI in z/OS,” on page 173 for more information.

If a font is referenced in a PDF file, but not embedded, then the ARSPDUMP program attempts to find the font using information provided with the //ADOBERES and //ADOBEFNT DD statements. If the ARSPDUMP program does not find the font, then it uses a substitute Adobe Type 1 font.

Sample JCL

Figure 31 shows sample JCL used to run the ARSPDUMP program.

```
//MSTEPHCA JOB 12345678,
//          MSTEPHE,REGION=0M,
//          NOTIFY=&SYSUID,CLASS=A,
//          MSGLEVEL=(1,1)
//PROCESS  OUTPUT  DEFAULT=YES,CLASS=*,JESDS=ALL,OUTDISP=HOLD
//RUNIT    EXEC PGM=ARSPDUMP,REGION=0M,
// PARM='-f DDN:IN -o DDN:OUT'
//STEPLIB DD DSN=ARS.V7R1M0.SARSLoad
//IN      DD DSN=MSTEPHE.PDF,DISP=SHR
//OUT     DD DSN=MSTEPHE.OUT,DISP=SHR
//ADOBERES DD DSN=ADOBE.PDFLIB.RESOURCE.INDEX(ADOBERES),DISP=SHR
//ADOBEFNT DD DSN=MSTEPHE.PDFP405F.ADOBEFNT.LST,DISP=SHR
//TEMPATTR DD DSN=ADOBE.PDF405.PLUSP1C.TEMPATTR,DISP=SHR
//SYSTMP01 DD UNIT=VIO,DSN=&&SYSTM1,DISP=(NEW,PASS),SPACE=(CYL,(5,5))
//SYSTEM  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

Figure 31. Sample JCL for the ARSPDUMP Program

About the JCL statements

The JCL statements in Figure 31 are explained as follows.

//RUNIT EXEC PGM=ARSPDUMP

Specifies the name of the program – ARSPDUMP.

// PARM='-f DDN:IN -o DDN:OUT'

The -f DDN: parameter specifies the DD name that contains the data set

name and other attributes of the input file (**//IN** for this example). The **-o DDN:** parameter specifies the DD name that contains the data set name and other attributes for the output file (**//OUT** for this example).

Note: If you specify the **-o PARM** value and omit the DD name, the ARSPDUMP program writes the output to STDOUT.

//IN Specifies the data set name and other attributes of the input file.

//OUT Specifies the data set name and other attributes of the output file.

//ADOBERES

Specifies the data set name of the library that contains the Adobe fonts.

//ADOBEFNT

Specifies the data set name for the list of Adobe fonts that the ARSPDUMP program found in the Adobe font library.

Chapter 20. Trace facility

Beginning with SPE 2, an enhanced tracing capability for the PDF indexer is now available. The tracing capability provides assistance to users attempting to debug problems, such as when the system fails during the indexing and loading of PDF documents.

To trace or debug a problem with the PDF indexer, the following is required:

- The parameter file, which specifies the fields, triggers, indexes and other indexing information
- The PDF input file to process

The parameter file and PDF input file can be processed by running the PDF indexer from the command line. For example:

```
arspdoci parmdd=filen.parms inputdd=filen.pdf outputdd=filen.out indexdd=filen.ind  
tracedd=filen.trace
```

Where:

arspdoci is the name of the command-line version of the PDF indexer program

parmdd= specifies the name of the input file that contains the indexing parameters

inputdd= specifies the name of the PDF input file to process

outputdd= specifies the name of the output file that contains the indexed PDF documents created by the PDF indexer

indexdd= specifies the name of the output file that contains the index information that will be loaded into the database

tracedd= specifies the name of the output file that contains the trace information

Note: See Chapter 18, “Using ARSPDOCI in z/OS,” on page 173 for more information about the parameters that may be specified when running the ARSPDOCI program.

After running the PDF indexer with the trace, the output file specified by the tracedd= parameter will contain detailed information about the processing that took place and where the PDF indexer is failing during the process. The trace information will identify whether a trigger was not found, a field was not found, the PDF data is corrupted, there was a problem extracting a PDF page from the document, or even if there is not enough memory or disk space to complete the required operations. Figure 32 on page 180 shows an example of the trace information that may be generated by the PDF indexer.

```

COORDINATES=IN
ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
TRIGGER1=UL(7.00,0.25),LR(7.70,0.57),*, 'Page:'
ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
ARSPDOCI completed code parse_trigger <-----
ARSPDOCI completed code parse_quoted_parm <-----
ARSPDOCI completed code parse_quoted_parm 001 ----->
ARSPDOCI completed code parse_trigger 001 ----->
FIELD1=UL(7.00,0.48),LR(7.90,0.77),0,(TRIGGER=1,BASE=0)
ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
ARSPDOCI completed code parse_field <-----
ARSPDOCI completed code parse_subfields <-----
ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
ARSPDOCI completed code parse_subfields 001 ----->
ARSPDOCI completed code parse_field 001 ----->
FIELD2=UL(6.11,1.39),LR(7.15,1.57),0,(TRIGGER=1,BASE=0)
ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
ARSPDOCI completed code parse_field <-----
ARSPDOCI completed code parse_subfields <-----
ARSPDOCI completed code get_keyword <-----

.
.
.

ARSPDOCI completed code get_keyword <-----
ARSPDOCI completed code get_keyword 003 ----->
ARSPDOCI completed code arsparm_final_sanity_check <-----
ARSPDOCI completed code arsparm_final_sanity_check 001 ----->
ARSPDOCI completed code ArspProcessOpt <-----
ARSPDOCI completed code ArspOpenIndex <-----
ARSPDOCI completed code ArspOpenIndex 001 ----->
Adobe PDF Library version -732512488.-1
Editing is : -1
Number of input pages = 130
ARSPDOCI completed code ArspProcessOpt:Calling ArspSearchDocPages()
ARSPDOCI completed code ArspSearchDocPages <-----
ARSPDOCI completed code ArspSearchDocPages: ArspCreateWordFinder()
ARSPDOCI completed code ArspSearchDocPages: PDWordFinderAcquireWordList()
ARSPDOCI completed code ArspSearchDocPages: PDDocAcquirePage()
ARSPDOCI completed code ArspSearchDocPages: ArspSearchPage()
ARSPDOCI completed code ArspSearchDocPages: PDPageRelease()
ARSPDOCI completed code ArspSearchDocPages: PDWordFinderReleaseWordList()
Trigger(s) not found by page 1
ARSPDOCI completed code ArspSearchDocPages 004 ----->
ARSPDOCI completed code ArspProcessOpt:Calling ArspCloseIndex()
ARSPDOCI completed code ArspCloseIndex <-----
ARSPDOCI completed code ArspCloseIndex 001 ----->
ARSPDOCI completed code ArspProcessOpt:Calling PDDocClose()
ARSPDOCI completed code ArspProcessOpt 002 ----->
ARSPDOCI completed code 1
ARSPDOCI completed code ArspFreeParms ()

```

Figure 32. Trace Information for the PDF indexer

Part 4. Xenos transform

Note: In this publication, the term Xenos transform refers to the Xenos d2e Platform. Xenos d2e Platform is a trademark of Xenos Group Inc.

This part of the book provides information about the Xenos transforms. When you load AFP, Metacode/DJDE, or PCL print files into the system, you can use the Xenos transforms to extract index data from the input data and convert the input data into AFP (Metacode/DJDE input only), Metacode (Metacode/DJDE input only), or PDF documents. If you use the OnDemand Web Enablement Kit (ODWEK) to retrieve AFP or Metacode documents from the system, you can use the Xenos transforms to convert the documents into AFP (Metacode only), HTML, PDF, or XML files before sending the documents to the browser.

Chapter 21. Understanding Xenos

Beginning with OnDemand Version 7.1, you can process input files and documents with transforms that are provided by Xenos. This section provides an overview of the Xenos transforms, explains the functions that the Xenos transform can perform, and describes different scenarios for processing your input files and documents.

Important: Before you attempt to use the Xenos transforms on your system, you must obtain the transform programs, license, and documentation from your IBM representative. Your IBM representative can also provide education that is available and other types of help and support for processing data with the transform programs.

The Xenos transforms are batch application programs that let you process several different types of input print files or documents that are stored in the system. The Xenos transforms provide converting, indexing, and resource collecting capabilities that let you archive, retrieve, and view and print documents.

The Xenos transforms can be used in an OnDemand system:

- When loading input files into the system.
- When retrieving documents from the system by using ODWEK.

When loading input files into the system, you can use the Xenos transforms to:

- Convert AFP input files to PDF.
- Convert Metacode/DJDE input files to AFP, Metacode, or PDF.
- Convert PCL input files to PDF.
- Index input files to enhance your ability to view, archive, or retrieve individual pages or groups of pages from large print files.
- For Metacode/DJDE to Metacode, collect the resources needed for printing or viewing a document, so that you can print and view the exact document, possibly years after its creation.

The Xenos transforms process the input print data and resources and produce these files:

- Index file
- Document file
- Resource file (Metacode/DJDE to Metacode only)

With the files that the Xenos transforms create, you can store the data into OnDemand and then use the Windows client to search for and retrieve, view, and print the documents. You can also use ODWEK to retrieve and transform documents.

When retrieving documents from the system by using ODWEK, you can use the Xenos transforms to:

- Convert AFP documents to HTML, PDF or XML files.
- Convert Metacode documents to AFP, HTML, PDF, or XML files.

The files that the Xenos transforms creates can then be sent to a Web browser for viewing and printing.

Figure 33 shows a high-level overview of how the Xenos transform fits into an OnDemand system for creating, indexing, viewing, and printing documents. The figure shows the resources and the AFP, Metacode/DJDE, or PCL print data, which can be provided by various products, that can flow into the ARSLOAD program for processing. If the Indexer that is specified in OnDemand is Xenos, then the ARSLOAD program calls the Xenos transform with parameter and script files that you create. The files that the Xenos transform produces can then be processed by the ARSLOAD program for archiving and the client programs for viewing and printing. If you plan to use ODWEK, you can transform AFP and Metacode documents that are stored in the system before sending them to the Web browser. For example, you could use the Xenos transform with the ARSLOAD program to process and load Metacode/DJDE print files. Then, you could use ODWEK to retrieve a Metacode document from the system, call the Xenos transform to convert the Metacode document into a PDF file (by using other parameter and script files that you create), and send the PDF file to the browser.

Note: The Metacode output file should have {MVS-LF2XM} specified on the dm_MetaGenOpen statement of the script.

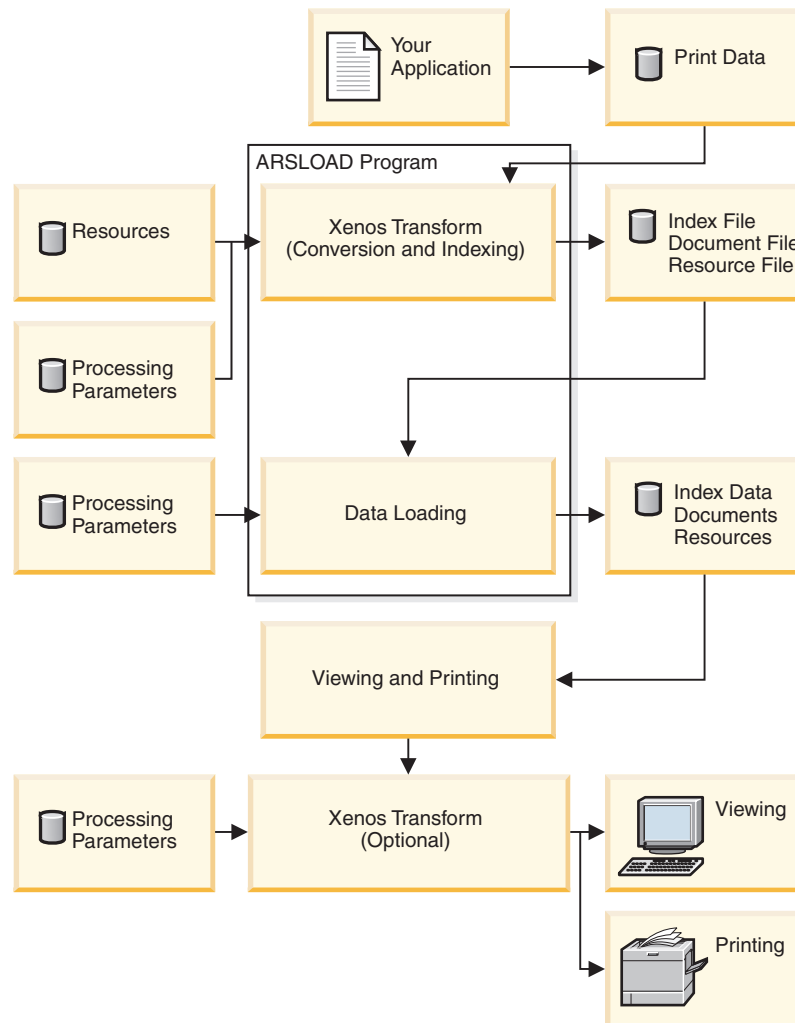


Figure 33. How the Xenos Transform fits into the OnDemand Environment

Chapter 22. Xenos transform

You can use the Xenos transform to perform these functions:

- Convert data streams
- Index documents
- Collect resources

Convert data streams

OnDemand customers can use the Xenos transforms for the following types of data conversions:

- Convert AFP documents to HTML
- Convert AFP data to PDF
- Convert AFP documents to XML
- Convert Metacode/DJDE data to AFP
- Convert Metacode documents to HTML
- Convert Metacode/DJDE data to PDF
- Convert Metacode/DJDE data to Metacode documents
- Convert Metacode documents to XML
- Convert PCL input files to PDF documents

The following sections describe each type of data. Please see the Xenos documentation to better understand the data formats and for more information about the transforms.

AFP documents to HTML

The AFP to HTML transform is used when retrieving AFP documents from an OnDemand server by using ODWEK.

There are two AFP to HTML transforms:

- The AFP to HTML/CSS transform converts AFP documents into HTML files with linked Cascading Style Sheets (CSS) for display with a Web browser. The output uses fonts from the user's system to display the pages, so it may not match the original as closely as other formats. Graphics are displayed using linked PNG image files. A separate HTML file is generated for each page in the original document. An optional HTML frameset file can also be generated to permit easier navigation between pages. The frameset is an HTML page that displays the individual HTML pages in a large frame at the top, and has links to the other pages in a small frame at the bottom.
- The AFP to HTML/Template Merger allows data from the input document to be inserted in a predefined template file. This allows the pages to be viewed to have a different format and layout than the printed page for redesigning the page to look and fit better on a standard computer screen. Multiple templates can be used in a single application, as needed, to suit the variety of information that is found from page to page in the input document.

AFP data to PDF

The AFP to PDF transform can be used when loading AFP input files into the system or retrieving AFP documents from the system by using ODWEK.

When loading AFP input files into the system, the transform can process linedata and mixed mode files that print on AFP printers through Infoprint or PSF as well as fully-composed MO:DCA-P files. The AFP to PDF transform converts AFP data streams into Adobe PDF documents. The PDF output can be viewed at the Windows client by using Adobe Acrobat or Adobe Acrobat Approval. To run the AFP to PDF transform when loading AFP input files into the system, set the Data Type on the View Information page to PDF and set the Indexer on the Indexing Information page to Xenos. Specify processing parameters on the Indexing Information page.

When retrieving AFP documents from the system by using ODWEK, the transform processes fully-composed MO:DCA-P files. The PDF output can be viewed at the Web browser by using an Adobe PDF viewer. To run the AFP to PDF transform when retrieving AFP documents from the system, set the AFPVIEWING parameter to XENOS in the ARSWWW.INI file and specify PDF as the OUTPUTTYPE in the ARSXENOS.INI file.

AFP documents to XML

The AFP to HTML transform is used when retrieving AFP documents from an OnDemand server by using ODWEK.

The AFP data to XML transform uses the AFP to HTML/Template Merger to allow data from the input document to be inserted in a predefined template file. This allows the pages to be viewed to have a different format and layout than the printed page, such as separating summary information from transaction details. The template is written in XML, for applications such as an online payment system. Multiple templates can be used in a single application, as needed, to suit the variety of information that is found from page to page in the input document.

Metacode to AFP

The Metacode to AFP transform can be used when loading Metacode/DJDE input files into the system or retrieving Metacode documents from the system by using ODWEK.

When loading Metacode/DJDE input files into the system, the transform converts Xerox Metacode and Linedata Conditioned Data Streams (LCDS) into AFP documents. Linedata and mixed mode files that print on Metacode Printers are supported, as well as pure Metacode files. The AFP output can be viewed at the Windows client. To run the Metacode to AFP transform when loading Metacode input files into the system, set the Data Type on the View Information page to AFP and set the Indexer on the Indexing Information page to Xenos. Specify processing parameters on the Indexing Information page.

When retrieving Metacode documents from the system by using ODWEK, the transform processes pure Metacode documents. The AFP output can be viewed at the Web browser by using the AFP Web Viewer. To run the Metacode to AFP transform when retrieving Metacode documents from the system by using ODWEK, set the METAVIEWING parameter to XENOS in the ARSWWW.INI file and specify AFP as the OUTPUTTYPE in the ARSXENOS.INI file.

Metacode documents to HTML

The Metacode to HTML transform is used when retrieving Metacode documents from an OnDemand server by using ODWEK.

There are two Metacode to HTML transforms:

- The Metacode to HTML/CSS transform converts Metacode documents into HTML files with linked Cascading Style Sheets (CSS) for display with a Web browser. The output uses fonts from the user's system to display the pages, so it may not match the original as closely as other formats. Graphics are displayed using linked PNG image files. A separate HTML file is generated for each page in the original document. An optional HTML frameset file can also be generated to permit easier navigation between pages. The frameset is an HTML page that displays the individual HTML pages in a large frame at the top, and has links to the other pages in a small frame at the bottom.
- The Metacode to HTML/Template Merger transform allows data from the input document to be inserted in a predefined template file. This allows the pages to be viewed to have a different format and layout than the printed page for redesigning the page to look and fit better on a standard computer screen. Multiple templates can be used in a single application, as needed, to suit the variety of information that is found from page to page in the input document.

Metacode to Metacode

The Metacode to Metacode transform is used when loading Metacode/DJDE input files into the system.

The Metacode to Metacode transform converts Xerox Metacode and Linedata Conditioned Data Streams (LCDS) to Metacode documents. Linedata and mixed mode files that print on Metacode Printers are supported, as well as pure Metacode files. At first, the Metacode to Metacode transform might seem a bit redundant, converting a format that already prints on the destination printer to the same format. But the input to the transform can be inefficient linedata or very obscure metacode, where the resulting output Metacode is efficient and in a predictable format, which allows individual pages or groups of pages in the output Metacode to be indexed and retrieved.

To run the Metacode to Metacode transform when loading Metacode input files into the system, set the Data Type on the View Information page to Metacode and set the Indexer on the Indexing Information page to Xenos. Specify processing parameters on the Indexing Information page.

Metacode to PDF

The Metacode to PDF transform can be used when loading Metacode input files into the system or retrieving Metacode documents from the system by using ODWEK.

When loading Metacode input files into the system, the transform converts Xerox Metacode and Linedata Conditioned Data Streams (LCDS) into PDF documents. Linedata and mixed mode files that print on Metacode Printers are supported, as well as pure Metacode files. The PDF output can be viewed at the Windows client by using Adobe Acrobat or Adobe Acrobat Approval. To run the Metacode to PDF transform when loading Metacode input files into the system, set the Data Type on the View Information page to PDF and set the Indexer on the Indexing Information page to Xenos. Specify processing parameters on the Indexing Information page.

When retrieving Metacode documents from the system by using ODWEK, the transform processes pure Metacode documents. The PDF output can be viewed at the Web browser by using an Adobe PDF viewer. To run the Metacode to PDF transform when retrieving Metacode documents from the system by using ODWEK, set the METAVIEWING parameter to XENOS in the ARSWWW.INI file and specify PDF as the OUTPUTTYPE in the ARSXENOS.INI file.

Metacode documents to XML

The Metacode to XML transform is used when retrieving Metacode documents from an OnDemand server by using ODWEK.

The Metacode to XML transform uses the Metacode to HTML/Template Merger to allow data from the input document to be inserted in a predefined template file. This allows the pages to be viewed to have a different format and layout than the printed page, such as separating summary information from transaction details. The template is written in XML, for applications such as an online payment system. Multiple templates can be used in a single application, as needed, to suit the variety of information that is found from page to page in the input document.

PCL to PDF

The PCL to PDF transform is used when loading PCL input files into the system.

The PCL to PDF transform converts Hewlett Packard Printer Control Language (PCL) print files into Adobe PDF documents. The term PCL refers to the compound data stream used by the Hewlett Packard (HP) printers. The transform accepts most PCL 4 or 5 designed for HP desktop printers; the transform does not support the HP PGL or HP Deskjet formats. The PDF output can be viewed at the Windows client by using Adobe Acrobat or Adobe Acrobat Approval. The PDF output can also be viewed at a Web browser by using an Adobe PDF viewer.

To run the PCL to PDF transform when loading PCL input files into the system, set the Data Type on the View Information page to PDF and set the Indexer on the Indexing Information page to Xenos. Specify processing parameters on the Indexing Information page.

Index documents

The Xenos transforms can index input files. When indexing with the Xenos transforms, you can divide a large print file into smaller, uniquely identifiable units, called *groups*. For example, you can use the Xenos transforms to divide a large print file that was created by a bank statement application into individual groups by generating group indexes that define the group boundaries in the file. A group is a named collection of sequential pages, which, in this example, consists of the pages describing a single customer's account. For example, a bank statement application probably produces a large printout consisting of thousands of individual customer statements. You can think of these statements as smaller, separate units, each uniquely identifying an account number, date, Social Security number, or other attributes.

Using the Xenos transforms, you can create an OnDemand generic index file. The index file lets you:

- Retrieve individual statements from storage volumes, based on an account number or any other attribute.

- More rapidly access the statements for viewing by, for example, the Windows client.
- Archive individual statements or the entire indexed print file for long-term storage and subsequent data management and reprinting, even years after its creation.

The Xenos transforms can create a generic index file for the following types of input files:

- AFP data
- Metacode
- PCL data

The Xenos transforms let you generate the group indexes by extracting values that are present in the input data itself, when the data has been formatted so that the Xenos transforms can reliably locate the values. This kind of indexing is called *indexing with data values*.

Indexing with data values

Some applications such as payroll or accounting statements contain data that might be appropriate to use for indexing tags. In the bank statement example, the account number is a type of data value that you might want to tag. You can then store a single customer's account statement using the account number, and you can retrieve and view the same statement using the account number. If the data value that you want to use in an indexing tag is consistently located in the same place for each statement, then you can specify parameters to the Xenos transform that create a separate group of pages for each statement.

Collect resources

Resources can be collected only when running the Metacode/DJDE to Metacode transform.

The Xenos transform can determine the list of resources needed to view the print file and collect the resources from the specified libraries. After you load the document and the resources into the system, you can view the document with fidelity.

Summary

Figure 34 shows the first part of the load process – you run the ARSLOAD program to process an input file and the Indexer that is specified to OnDemand is Xenos.

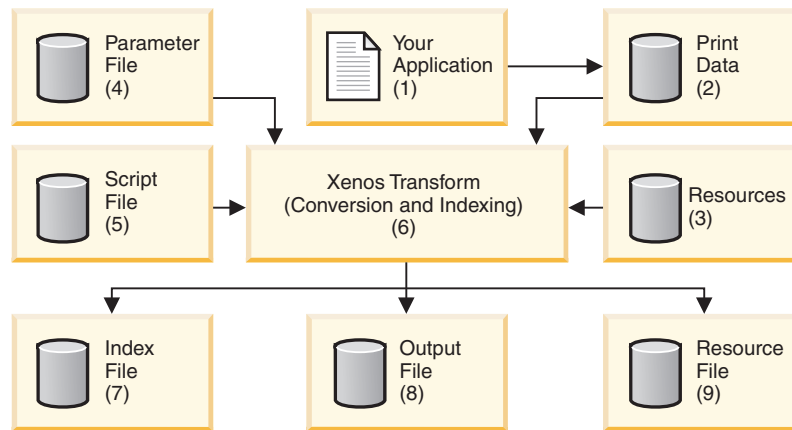


Figure 34. Using the Xenos Transforms to Prepare Files for Loading and Viewing

1. The process begins with your application, which is the program that generates your print data.
2. The print data can be AFP, Metacode, or PCL.
3. Resources are stored in resource libraries.
4. You create the Parameter File, which the Xenos transform uses to locate index data in the input file, convert the input to a specified type of output file, and so forth. See Figure 41 on page 204 for an example of a parameter file.
5. You create the Script File, which the Xenos transform uses to create the OnDemand generic index file and the other output files. See Figure 42 on page 206 for an example of a script file.
6. The ARSLOAD program calls the Xenos transform, to index the print data, convert the input file to the specified type of output, and collect the resources (Metacode output only).
7. The Index File contains the index data that is in the OnDemand generic index format.
8. The Output File contains the indexed groups of pages, also known as documents in OnDemand.
9. The Resource File (Metacode output only) contains the resources that are required to view and print the converted documents.

Chapter 23. Loading data

Figure 35 shows the second part of the load process – the ARSLOAD program processes the files that were created by the Xenos transform.

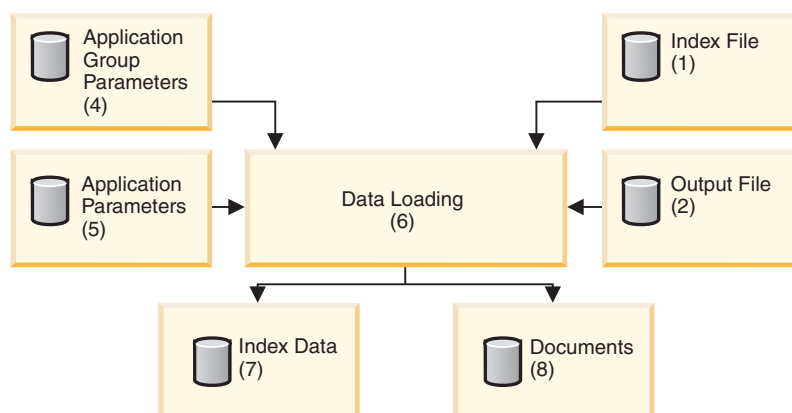


Figure 35. Loading Data into OnDemand

1. The Index File contains the index data that is in the OnDemand generic index format.
2. The Output File contains the indexed groups of pages, also known as documents in OnDemand.
3. The Resource File (Metacode output only) contains the resources that are required to view and print the documents.
4. You create the application group parameters, which include the database and storage management information that the ARSLOAD program uses to process the input files. For example, the database fields that you define for the application group will hold the index field values that the Xenos transform extracted from the original print data.
5. You create the application parameters, which include the type of data and the indexer information that the ARSLOAD program uses to process the input files. For example, on the View Information page in the application, specify the type of data that will be stored in OnDemand, that is, the output from the Xenos transform (AFP, Metacode, or PDF); on the Indexing Information page, specify Xenos as the indexing program and specify the parameters that are used to process the input file and create the output file, collect the resources, and generate the index data. See “Specifying parameters to OnDemand” on page 198 for information about creating the application parameters. Figure 38 on page 200 and Figure 39 on page 200 show examples of the Indexing Information page.
6. The ARSLOAD program stores the index data into the database and loads the documents and resources on storage volumes.
7. The Index Data is loaded into the database.
8. The Documents are loaded on to storage volumes.
9. The Resources (Metacode output only) are loaded on to storage volumes.

Chapter 24. Scenarios for using Xenos

You can use the Xenos transform to process your files for:

- Viewing and printing locally with the Windows client
- Viewing and printing locally with Web Enablement Kit feature

Viewing and printing with the Windows client

Figure 36 shows the steps that you take to view and print documents locally with the Windows client.

1. The process begins with the Index Data (1a) and Documents (1b) that you loaded into the database and on to storage volumes with the ARSLOAD program (AFP or PDF documents).
2. Using the Windows client program, your users search for and view the indexed documents. They can also print documents on local printers from the Windows client.

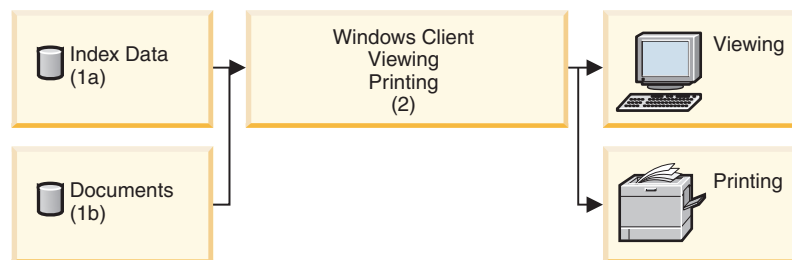


Figure 36. Viewing and Printing with the Windows Client

Viewing and printing with the OnDemand Web Enablement Kit

Note: See the *IBM DB2 Content Manager OnDemand for z/OS: Web Enablement Kit Implementation Guide*, SC27-1376 for information about how to configure ODWEK to use the Xenos transforms.

Figure 37 on page 196 shows the steps that you take to view and print documents locally with ODWEK.

1. The process begins with the Index Data (1a), Documents (1b), and Resources (1c) that you loaded into the database and on to storage volumes with the ARSLOAD program (AFP, Metacode, or PDF documents).
2. In the ARSWWW.INI file, you specify information about the documents that the WEK retrieves from the server. For AFP and Metacode documents, you can specify that a different type of output should be sent to the Web browser. For example, you can specify that Metacode documents that are retrieved from the system should be transformed into PDF files that are sent to the Web browser.
3. In the ARSXENOS.INI file, you specify the output type and the name of the parameter file and the script file that are used by the Xenos transform to process the input document and create the output document that is sent to the Web browser.
4. When ODWEK retrieves a document, it checks the ARSWWW.INI file to determine if document conversion is required.
5. If document conversion is required, then ODWEK sends the document to the Xenos transform; otherwise, ODWEK sends the document to the Web browser.
6. You create the Parameter File, which provides information about the type of conversion that the Xenos transform performs.
7. You create the Script file, which the Xenos transform uses to create the output document.
8. The Xenos transform converts the document from the format in which it was stored on the server to a format that can be viewed at the Web browser.
9. The user views the document from a Web browser. The user can also print documents on local printers from the Web browser.

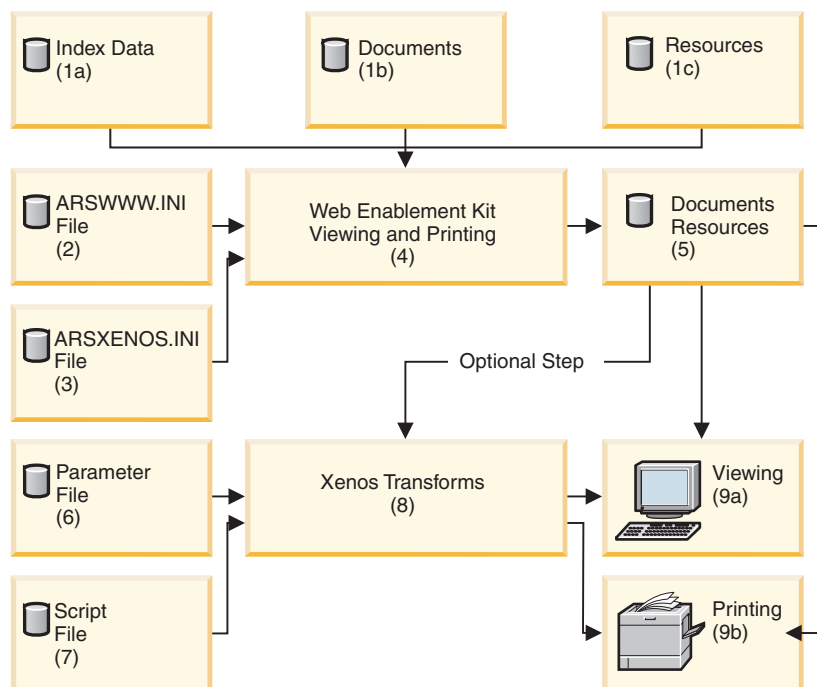


Figure 37. Viewing and Printing with the Web Enablement Kit

Chapter 25. How to specify parameters to the Xenos transform

This section describes how to specify parameters to the Xenos transform to assist you with developing indexing parameters and to specify the parameters that are used by the ARSLOAD program and the Xenos transform to load input data into the system. See *IBM DB2 Content Manager OnDemand for z/OS: Web Enablement Kit Implementation Guide*, SC27-1376 for information about how to specify parameters when you use the WEK to run the Xenos transform.

Xenos provides a graphical interface, called Developer's Studio, to automate the creation of parameter and script files needed to run a transform. Please see the Xenos documentation for instructions about how to use Developer's Studio. If you do not have Developer's Studio, see "Processing sample data" for a discussion of how you can manually determine indexing parameters and see the examples of parameter and script files in "Specifying parameters to OnDemand" on page 198.

There are two parts to specifying the parameters that are used by the ARSLOAD program and the Xenos transform.

- First, you should process some sample input data to determine the locations of the text strings that the Xenos transform uses to identify groups and locate the field index values. You can do this either by using Developer's Studio or by running the Xenos transform from the command line and specifying the parameters for this step.
- Then you should specify the parameters that are used by the ARSLOAD program to call the Xenos transform and specify the parameter file and script file that the Xenos transform uses to process the input data and create the output files. You can specify all of the parameters for this step by using the administrative client.

Processing sample data

You can use the Xenos transform (the JSMAIN program) to help you determine the location of the text strings in the input data. The JSMAIN program processes one or more pages of an input file and generates an output file. The output file contains one record for each text string on a page. Each record contains the 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 input file.
- Identify the string values that you want to use to locate fields
- Identify the number of the page on which 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 0 (zero), 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 input file with the JSMAIN program.
- In the output file (that is identified by the FDDLOUTPUT parameter), locate the records that contain the string values and make a note of the coordinates.
- Create FIELD parameters using the coordinates, page number, and string value.

See “Generating the locations of text strings” on page 201 for an example of how to run the JSMAIN program to process sample data.

For information about the parameters that you can specify, including options and data values, and the script file, see your Xenos documentation.

Specifying parameters to OnDemand

The indexing parameters and other processing parameters that are used by the ARSLOAD program and the Xenos transform are part of the OnDemand application. The administrative client provides an edit window that you can use to maintain the parameters for the application. The parameters are:

- The license file. You must always specify the name of the Xenos license file by using the `OnDemandXenosLicenseFile` parameter. For example:

```
OnDemandXenosLicenseFile=XENOS.LICENSE
```

The file that you specify must be a valid license file that you obtained from Xenos.

- The warning level. The `OnDemandXenosWarningLevel` parameter determines how the ARSLOAD program will handle return codes from the Xenos transform. For example:

```
OnDemandXenosWarningLevel=4
```

The Xenos transform sets a return code after it converts an input file. Use this parameter to specify the maximum return code that the ARSLOAD program will consider good and continue with the load process. For example, if you specify 4 (four), then the return code that is set by the Xenos transform must be four or less; otherwise, the load will fail. The default value is zero. If you do not specify this parameter, then the Xenos transform must set a return code of zero. Otherwise, the load will fail. See the Xenos documentation for details about return codes.

- The parameter file. You can use one of two methods to specify the parameters: the *file name* method and the *details* method.

With the file name method, you specify the name of the parameter file. The parameter file that you specify must contain all of the indexing and conversion parameters and other parameters that the Xenos transform uses to process the input data. Use the `OnDemandXenosParmFile` parameter to specify the name of the parameter file. For example:

```
OnDemandXenosParmFile=XENOS.PARMPFILE(META2PDF)
```

Figure 38 on page 200 shows an example of the Indexing Information page when you use the file name method.

With the details method, you must specify all of the parameters in the edit window within the application. Enclose the parameters inside of the `OnDemandXenosParmBegin` and `OnDemandXenosParmEnd` parameters. For example:

```
OnDemandXenosParmBegin
fddmslib = XENOS.SCRIPTS(DMSL)
scriptvar = ( 'Parser', 'META' )
scriptvar = ( 'Generator', 'PDF' )
scriptvar = ( 'NumberOfFields', 2 )
.
.
.
OnDemandXenosParmEnd
```

Figure 39 on page 200 shows an example of the Indexing Information page when you use the details method. **Note:** The example is for demonstration purposes only; not all of the parameters are shown in the edit window.

- The script file. As with the parameter file, you can use the file name method or the details method to specify the script. The script file that you specify must contain all of the code that the Xenos transform uses to generate the OnDemand generic index file and the other output files. Use the OnDemandXenosScriptFile parameter to specify the name of the parameter file. For example:

```
OnDemandXenosScriptFile=XENOS.SCRIPTS(INDEXDMS)
```

Figure 38 on page 200 shows an example of the Indexing Information page when you use the file name method.

With the details method, you must specify all of the script statements in the edit window within the application. Enclose the script statements inside of the OnDemandXenosScriptBegin and OnDemandXenosScriptEnd parameters. For example:

```
OnDemandXenosScriptBegin
TRUE = 1;
FALSE = 0;
call dm_Initialize
rc = dm_SetParm( par_h, 'fdinput', inputfile )
.
.
.
OnDemandXenosScriptEnd
```

Figure 39 on page 200 shows an example of the Indexing Information page when you use the details method. **Note:** The example is for demonstration purposes only; not all of the script statements are shown in the edit window.

See “Indexing and converting input data” on page 207 for an example of a parameter file and a script file to convert and index data and generate the OnDemand generic index file and the other output files.

For information about the parameters that you can specify, including options and data values, and the script file, see your Xenos documentation.

Important: When loading data using Xenos, the locale must be set appropriately for the code page emitted by the indexing script (in the example, dm_DASDWrite(index_h, "CODEPAGE:500"). For example, if CODEPAGE:273 is emitted, set the LC_ALL environment variable to De_DE.IBM-273 or some other locale that correctly identifies upper and lower case characters in code page 273

Add an Application

Load Information | Logical View Fields | Logical Views | Print Options

General | View Information | Indexer Information

Indexer: **Xenos**

Parameters Source:
☒ Keyboard
☐ Sample Data
☐ Parameter File **Modify...**

Details...

```

OnDemandXenosLicenseFile = XENOS.LICENSE
OnDemandXenosWarningLevel = 4
OnDemandXenosParmFile = XENOS.PARMFILE (META2PDF)
OnDemandXenosScriptFile = XENOS.SCRIPTS (DMSL)

```

OK Cancel Apply Help

Figure 38. Application Indexing Information – Specifying File Names

Add an Application

Load Information | Logical View Fields | Logical Views | Print Options

General | View Information | Indexer Information

Indexer: **Xenos**

Parameters Source:
☒ Keyboard
☐ Sample Data
☐ Parameter File **Modify...**

Details...

```

OnDemandXenosLicenseFile = XENOS.LICENSE
OnDemandXenosWarningLevel = 4
OnDemandXenosParmBegin
fdmslib = XENOS.SCRIPTS (DMSL)
scriptvar = ( 'Parser', 'META' )
scriptvar = ( 'Generator', 'PDF' )
scriptvar = ( 'NumberOfFields', 0 )
.
.
OnDemandXenosParmEnd

OnDemandXenosScriptBegin
TRUE = 1;
FALSE = 0;
call dm_initialize
rc = dm_SetParm( par_h, 'fdinput', inputfile );
.
.
OnDemandXenosScriptEnd

```

OK Cancel Apply Help

Figure 39. Application Indexing Information – Specifying Details

Chapter 26. Using JSMAIN in z/OS

This chapter provides information about using the JSMAIN program in the z/OS environment, outside of the OnDemand application.

Note: The statements in this chapter about using the JSMAIN program in z/OS apply equally to using the JSMAIN program in OS/390.

Xenos provides a graphical interface, called Developer's Studio, to automate the creation of parameter and script files needed to run a transform. Please see the Xenos documentation for instructions about how to use Developer's Studio. If you do not have Developer's Studio, see "Processing sample data" on page 197 for a discussion of how you can manually determine indexing parameters and see the examples of parameter and script files in "Specifying parameters to OnDemand" on page 198.

The JSMAIN program can be used to process AFP, Metacode/DJDE, and PCL print data. The JSMAIN program can also extract index data from the print data and for Metacode output, collect the resources that are required to view and print the data. The JSMAIN program can be used to do the following:

- Print the locations of the text strings found on the pages of an input file. You can use the JSMAIN program to help define the indexing fields for your input data. When you define indexing fields, you must specify the location of the string value used to locate the field as a coordinate system imposed on the page. For each string value, you must identify the upper left and lower right position on the page. To help you create the indexing parameters, you can run the JSMAIN program to process a sample input file and list the text strings found on the pages of the input file and the locations of the text strings. When you run the JSMAIN program to print the locations of the text strings, you must specify the FDDLOUTPUT parameter. The FDDLOUTPUT parameter identifies the name of the file that will contain the text strings and the location information. See "Generating the locations of text strings" for an example.
- Convert input data, generate an index file, and for Metacode output, collect resources. The ARSLOAD program will automatically call the JSMAIN program if the Indexer that is specified on the Indexer Information page under applications is Xenos. You can also run the JSMAIN program from a batch job to generate an index file manually. See "Indexing and converting input data" on page 207 for an example.

Generating the locations of text strings

This example shows how to use the JSMAIN program to process a Metacode/DJDE input file and produce the text strings and location information. You can use the information generated in this step to specify the indexing information that is used by the JSMAIN program to extract index data from the input files that you want to load on the system (see "Indexing and converting input data" on page 207).

- Figure 40 on page 203 shows the sample JCL used for this example.
- Figure 41 on page 204 shows the processing parameters used for this example. Note that all of the parameters that are required to process an input file and produce the text strings and the location information are not shown in the

example. See your Xenos documentation for a complete list of indexing parameters, options, and data values and more detailed information.

Notes:

1. The script variables Parser and Generator determine the type of conversion taking place. The script variable Parser represents the input file format. The script variable Generator represents the output file format. Table 5 lists the possible values for the Parser and Generator variables when loading input data into the system.

Table 5. Parser and Generator Variables

Input	Output	Parser	Generator
AFP	PDF	scriptvar=('Parser', 'AFP')	scriptvar=('Generator', 'PDF')
Metacode / DJDE	AFP	scriptvar=('Parser', 'META')	scriptvar=('Generator', 'AFP')
Metacode / DJDE	Metacode	scriptvar=('Parser', 'META')	scriptvar=('Generator', 'META')
Metacode / DJDE	PDF	scriptvar=('Parser', 'META')	scriptvar=('Generator', 'PDF')
PCL	PDF	scriptvar=('Parser', 'PCL')	scriptvar=('Generator', 'PDF')

2. The script variable NumberOfFields must be set to 0 (zero) when running the JSMAIN program to produce text string coordinates.
3. The STARTPAGE and STOPPAGE parameters determine how many pages of the input file to process. In the example, the entire input file will be processed. To process a range of pages, specify the number of the starting page to process and the number of pages to process. For example, if you specify STARTPAGE=0 and STOPPAGE=10, then the JSMAIN program will process the first eleven pages of the input file.

Note: The input data must contain the information that the JSMAIN program uses to determine when one page ends and another page begins. For example, if the input data contains carriage control characters, then a Skip-to-Channel One carriage control character signals the beginning of a new page.

4. The FDDLOUTPUT parameter determines the file name or full path name of the file that will contain the text strings and the location information. The FDDLOUTPUT parameter is required when you run the JSMAIN program to generate the text strings and the locations.
- Figure 42 on page 206 shows the script file used for this example. **Note:** The script file does not create an index file, because an index file is not needed for this step.

Sample JCL

Figure 40 on page 203 shows the sample JCL used to run the JSMAIN program for this example.

```

//XXXXXX@ JOB (SYS10000),'D2E - IDC 4.1',NOTIFY=XXXXXX,
//          CLASS=A,MSGCLASS=X,REGION=0M
//*****
//*      Run this JCL to create the display list
//*
//*      CHANGE THE FOLLOWING:
//*
//* 'prefix' TO YOUR INSTALLATION DEFINED DATASET NAME PREFIX
//* use the M2PDOUT if you need to preallocate the output dataset
//*****
//ST01      EXEC PGM=JSMMAIN,PARM='-PARM=DSN:prefix.XENOS.PARM(METADL)'
//STEPLIB DD DSN=XENOS.D2E.DOCS.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTEM   DD SYSOUT=*
//MSGDUMP  DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDERR   DD SYSOUT=*
//JSLOG     DD SYSOUT=*                                JS Log File
//M2PDOUT DD DISP=(NEW,CATLG),DSN=prefix.SAMPLOUT,        Output file
//          DCB=(DSORG=PS,RECFM=VB,BLKSIZE=23476,LRECL=259),
//          SPACE=(CYL,(10,10),RLSE)
//FDDLOUT DD DISP=(NEW,CATLG),DSN=prefix.XENOS.SAMPLEDL,    Display list
//          DCB=(DSORG=PS,RECFM=VB,BLKSIZE=32760,LRECL=32756),
//          SPACE=(CYL,(5,5),RLSE)
//

```

Figure 40. Sample JCL for the JSMMAIN Program

About the JCL statements

The JCL statements in Figure 40 are explained as follows.

PARM='-PARM=DSN:prefix.XENOS.PARM(METADL)'

Specifies the data set name of the parameter file. The parameter file contains the parameters that are used to index and convert the input data, and, in this example, also the name of the script file. The script file creates the OnDemand generic index file and the other output files. See Figure 41 on page 204 for the parameter file used in this example. See Figure 42 on page 206 for the script file used in this example.

Note: For information about the parameters that you can specify, including options and data values, see your Xenos documentation.

JSLOG

Specifies the data set name of the file to which a job report will be written. The report will contain a list of the parameters used to process the input file, any error messages, and a list of resources used to process the input file.

M2PDOUT

Specifies the data set name and attributes of the file that will contain the converted output data.

Note: If you are using the JSMMAIN program to generate the text strings and locations, then you can omit this DD statement or discard the output file.

FDDLOUT

Specifies the data set name and attributes of the file that will contain the text strings and the location information.

Sample parameter file

Figure 41 shows the sample parameter file used for this example.

```
/* Sample processing parameters for Meta2PDF transform */

JS:
/* Documorph DME job parameter file */
fdJobParm = 'DSN:XENOS.PARM(METADL)'
fdDmScript = 'DSN:XENOS.SCRIPTS(NOINDDMS)'
scriptvar=('inputfile','DSN:XENOS.INPUTMTA')
scriptvar=('outputfile','XXXXXXXX.SAMPLOUT')
fdLicense='DSN:XENOS.LICENSES(M020831)'

/* DM Script Library - XG supplied functions */
fddmslib = 'DSN:XENOS.SCRIPTS(DMSL)'

scriptvar=('Parser', 'META')
scriptvar=('Generator', 'PDF')
scriptvar=('NumberOfFields', 0)

METADL-METAP:
/* Metacode Parser Options */
startjdl   = GTIJDL
startjde   = START
StartPage  = 0
StopPage   = 5
Position   = WORD
Native     = NO
CC         = YES
Shade      = NONE
DashLine   = NO
LDMethod   = LARGEST
CCTran     = NONE
/* File Defs */
FDfnt      = 'DSN:XENOS.METARES(%s)'
FDjdl      = 'DSN:XENOS.METARES(%s)'
RespectShift = NO
/* MTA2PDF Options */
FDfonttab  = 'DSN:XENOS.D2E.AUXFILE(AGENTFCT)'
```

Figure 41. Sample Parameter File for the JSMAIN Program (Part 1 of 2)

```

PDFGEN-PDFOUT:
/* PDF Out Generator Options */
/* output file name being set in the script */
offset      = (0,0)
scaleby     = 100
border      = NONE
Compress    = (NONE,NONE,NONE)
orient      = AUTO
PDFAuthor   = 'Xenos Group'
PDFOpenAct  = '/FitH 800'
BMOrder     = (AsIs,AsIs,AsIs)
drawdlfields = yes

METADL-DLIST:
parmdpi     = 300
pagefilter  = all
resfilter   = all
fdDLoutput  = 'DSN:XXXXXXXX.XENOS.SAMPLEDL'

/* for field 1 */
FieldName   = (PAGE)
FieldWord   = (20, and, %20)
FieldPhrase = (%500, OneSpace)
FieldPara   = (%500)

```

Figure 41. Sample Parameter File for the JSMAIN Program (Part 2 of 2)

About the parameters

The parameters in Figure 41 on page 204 are explained as follows.

fdJobParm = 'DSN:XENOS.PARM(METADL)'

Specifies the data set name of the parameter file.

fdDmScript = 'DSN:XENOS.SCRIPTS(NOINDDMS)'

Specifies the data set name of the script file.

scriptvar=('inputfile','DSN:XENOS.INPUTMTA')

Specifies the data set name of the input file.

scriptvar=('outputfile','XXXXXXXX.SAMPLOUT')

Specifies the data set name of the file that will contain the converted output data.

fdLicense='DSN:XENOS.LICENSES(M020831)'

Specifies the Xenos license file. You must obtain a valid licence file from Xenos.

fdDLoutput='DSN:XXXXXXXX.XENOS.SAMPLEDL')

Specifies the data set name of the file that will contain the text strings and the location information.

Sample script file

Figure 42 on page 206 shows the sample script file used for this example.

```

TRUE = 1;
FALSE = 0;

call dm_Initialize

par_h = dm_StartParser(Parser);
gen_h = dm_StartGenerator(Generator);

rc = dm_SetParm(par_h, 'fdinput', inputfile);

/* start the DASD Distributors */
dasd_h = dm_StartDistributor("DASD");

/* open output file */
rc = dm_DASDOpen(dasd_h, '{GROUPFILENAME}'"DSN:"outputfile);

/* initialize */
file_open = FALSE

dlpage = dm_GetDLPage(par_h);

do while(dlpage <> 'EOF')
  if file_open = FALSE then do
    select
      when generator = 'PDF' then
        rc = dm_PDFFGenOpen(gen_h, '{GROUPFILEENTRY}'"DSN:"outputfile);
      when generator = 'AFP' then
        rc = dm_AFPGenOpen(gen_h, '{GROUPFILEENTRY}'"DSN:"outputfile);
      when generator = 'META' then
        rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}'"DSN:"outputfile);
      otherwise do
        say 'Invalid generator'
        return 12
      end
    end
  if rc = 0 then do
    file_open = TRUE
  end
end
end

```

Figure 42. Sample Script File for the JSMAIN Program (Part 1 of 2)

```

select
  when generator = 'PDF' then rc = dm_PDFGenWrite(gen_h, dlpage );
  when generator = 'AFP' then rc = dm_AFPGenWrite(gen_h, dlpage );
  when generator = 'META' then rc = dm_METAGenWrite(gen_h, dlpage );
end

dlpage = dm_GetDLPage(par_h);
end

if file_open = TRUE then do
  select
    when generator = 'PDF' then rc = dm_PDFGenClose( gen_h )
    when generator = 'AFP' then rc = dm_AFPGenClose( gen_h )
    when generator = 'META' then rc = dm_METAGenClose( gen_h )
  end
end

rc = dm_DASDClose( dasd_h )
return;

```

Figure 42. Sample Script File for the JSMAIN Program (Part 2 of 2)

Indexing and converting input data

This example shows how to use the JSMAIN program to process a Metacode/DJDE input file and produce an index file and a converted output file that can be loaded into OnDemand.

- Figure 43 on page 208 shows the sample JCL used for this example.
- Figure 44 on page 209 shows the processing parameters used for this example. Note that all of the parameters that are required to process an input file are not shown in the example. See your Xenos documentation for a complete list of processing parameters, options, and data values and more detailed information.

Notes:

1. The script variable NumberOfFields determines the number of index fields.
2. The script variables Field.1 and Field.2 identify the names of the index fields.
3. The script variables Break.1 and Break.2 tell Xenos whether to start a new group if the corresponding field value changes. In this example, when the value of the index field Field.2 changes, Xenos will start a new group. For more information about breaking indexes, see the description of BREAK in the ACIF parameter reference ("INDEX" on page 65).
4. The fieldlocate parameter determines the *anchor* character string that the Xenos transform will search for. The Xenos transform will locate the index fields based on the location of the anchor string.
5. The fieldbase parameters identify the locations of index fields using the specified offset from the anchor string, using the text string coordinates that were generated in "Generating the locations of text strings" on page 201.
6. The fieldname parameters identify the names of the index fields.
7. The STARTPAGE and STOPPAGE parameters determine how many pages of the input file to process. In the example, the entire input file is processed.

Note: The input data must contain the information that the JSMAIN program uses to determine when one page ends and another page begins. For example, if the input data contains carriage control characters, then a Skip-to-Channel One carriage control character signals the beginning of a new page.

8. The absence of the FDDLOUTPUT parameter means that the JSMAIN program will not generate the text strings and the location information.
 9. For information regarding the FieldLocate, FieldName, and FieldBase parameters, refer to your Xenos documentation.
- Figure 45 on page 211 shows the script file used for this example.
 - Figure 46 on page 215 shows the OnDemand generic index file that was created by the script file for this example.

Sample JCL

Figure 43 shows the sample JCL used to run the JSMAIN program for this example.

```
//XXXXXX@ JOB (SYS10000),'D2E - IDC 4.1',NOTIFY=XXXXXX,
//          CLASS=A,MSGCLASS=X,REGION=0M
//*****
//*      Run this JCL to convert Sample Metacode file to PDF
//*
//*      CHANGE THE FOLLOWING:
//*
//* 'prefix' TO YOUR INSTALLATION DEFINED DATASET NAME PREFIX
//*****
//ST01     EXEC PGM=JSMAIN,PARM='-PARM=DSN:prefix.XENOS.PARM(META2PDF)'
//STEPLIB  DD DSN=XENOS.D2E.DOCS.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTEM   DD SYSOUT=*
//MSGDUMP  DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//STDERR   DD SYSOUT=*
//JSLOG    DD SYSOUT=*                                JS Log File
//M2PDOUT  DD DISP=(NEW,CATLG),DSN=prefix.SAMPLOUT,      Output file
//          DCB=(DSORG=PS,RECFM=VB,BLKSIZE=23476,LRECL=259),
//          SPACE=(CYL,(10,10),RLSE)
//*
```

Figure 43. Sample JCL for the JSMAIN Program

About the JCL statements

The JCL statements in Figure 43 are explained as follows.

PARM='-PARM=DSN:prefix.XENOS.PARM(META2PDF)'

Specifies the data set name of the parameter file. The parameter file contains the parameters that are used to index and convert the input data, and, in this example, also the name of the script file. The script file creates the OnDemand generic index file and the other output files. See Figure 44 on page 209 for the parameter file used in this example. See Figure 45 on page 211 for the script file used in this example.

Note: For information about the parameters that you can specify, including options and data values, see your Xenos documentation.

JSLOG

Specifies the data set name of the file to which a job report will be written. The report will contain a list of the parameters used to process the input file, any error messages, and a list of resources used to process the input file.

M2PDOUT

Specifies the data set name and attributes of the file that will contain the converted output data.

Sample parameter file

Figure 44 shows a sample parameter file used for this example.

```
JS:
/* Documorph DME job parameter file */
fdJobParm = 'DSN:XXXXXXXX.XENOS.PARM(META2PDF)'
fdDmScript = 'DSN:XENOS.SCRIPTS(INDEXDMS)'
scriptvar=('inputfile','DSN:XENOS.INPUTMTA')
scriptvar=('outputfile','XXXXXXXX.SAMPLOUT')
scriptvar=('indexfile','[MVSALLOC:ALCUNIT=CYL,SPACE=5]DSN:XXXXXXXX.SAMPLIND')
fdLicense='DSN:XENOS.LICENSES(M020831)'

/* DM Script Library - XG supplied functions */
fddmslib = 'DSN:XENOS.SCRIPTS(DMSL)'

scriptvar=('Parser','META')
scriptvar=('Generator','PDF')
scriptvar=('NumberOfFields', 2)
scriptvar=('Field.1','Name')
scriptvar=('Field.2','Policy')
scriptvar=('Break.1','no')
scriptvar=('Break.2','yes')

METADL-METAP:
/* Metacode Parser Options */
startjdl   = GTIJDJL
startjde   = START
StartPage  = 0
StopPage   = 0
Position   = WORD
Native     = NO
CC         = YES
Shade      = NONE
DashLine   = NO
LDMethod   = LARGEST
CCTran     = NONE
/* File Defs */
FDfnt      = 'DSN:XENOS.METARES(%s)'
FDjsl      = 'DSN:XENOS.METARES(%s)'
RespectShift = NO
/* MTA2PDF Options */
FDfonttab  = 'DSN:XENOS.D2E.AUXFILE(AGENTFCT)'
```

Figure 44. Sample Parameter File for the JSMAIN Program (Part 1 of 2)

```

PDFGEN-PDFOUT:
/* PDF Out Generator Options */
/* output file name being set in the script */
offset      = (0,0)
scaleby     = 100
border      = NONE
Compress    = (NONE,NONE,NONE)
orient      = AUTO
PDFAuthor   = 'Xenos Group'
PDFOpenAct  = '/FitH 800'
BMOrder     = (AsIs,AsIs,AsIs)
drawdlfields = no

METADL-DLIST:
parmdpi = 300
pagefilter = all
resfilter = all
/*fdLoutput = 'DSN:XENOS.SAMPLEDL' */

/* for field 1 */
FieldName = (PAGE)
FieldWord = (20, and, %20)
FieldPhrase = (%500, OneSpace)
FieldPara = (%500)

/* extract name */
FieldLocate = ('InsName', 'Insured')
FieldName = ('Name')
FieldPhrase = (1000)
FieldBase = ('InsName',+275,'=',-35,'=',+800,'=',+30)

/* for field 2 */
FieldName = (PAGE)
FieldWord = (20, and, %20)
FieldPhrase = (%500, OneSpace)
FieldPara = (%500)

/* extract policy number */
FieldName = ('Policy')
FieldBase = ('InsName', +300,'=',+100,'=',+800,'=',+170)

FieldWord = (20, and, %20)
FieldPhrase = (%500, OneSpace)
FieldPara = (%500)

```

Figure 44. Sample Parameter File for the JSMAIN Program (Part 2 of 2)

About the parameters

The parameters in Figure 44 on page 209 are explained as follows.

fdJobParm = 'DSN:XXXXXXXX.XENOS.PARM(META2PDF)'

Specifies the data set name of the parameter file.

fdDmScript = 'DSN:XENOS.SCRIPTS(INDEXDMS)'

Specifies the data set name of the script file.

scriptvar=('inputfile','DSN:XENOS.INPUTMTA')

Specifies the data set name of the input file.

scriptvar=('outputfile','XXXXXXXXX.SAMPLOUT')

Specifies the data set name of the file that will contain the converted output data.

**scriptvar=('indexfile','[MVSALLOC:ALCUNIT=CYL,SPACE=5]
DSN:XXXXXXXXX.SAMPLIND')**

Specifies the space allocation for and the data set name of the file that will contain the index data.

fdLicense='DSN:XENOS.LICENSES(M020831)'

Specifies the name of the Xenos license file. You must obtain a valid licence file from Xenos.

Sample script file

Figure 45 shows a sample script file used for this example.

```
TRUE = 1;
FALSE = 0;

call dm_Initialize

par_h = dm_StartParser(Parser);
gen_h = dm_StartGenerator(Generator);

rc = dm_SetParm(par_h, 'fdinput', inputfile);

/* start the DASD Distributors */
dasd_h = dm_StartDistributor("DASD");
index_h = dm_StartDistributor("DASD");

/* open output and index files */
rc = dm_DASDOpen(dasd_h, '{GROUPFILENAME}'DSN:"outputfile");
/*rc = dm_DASDOpen(dasd_h, outputfile); */
rc = dm_DASDOpen(index_h, indexfile );

/* initialize */
do i = 1 to NumberOfFields
    fieldvaluesave.i = ""
    if Break.i \= "no" & Break.i \= "NO" then
        do
            Break.i = "yes"
        end
    end
end
file_open = FALSE
save_BytesWritten = 0
/* crlf = '0d0a'X */

/* write preamble to the index file */
rc = dm_DASDWrite(index_h, "COMMENT: OnDemand Generic Index File Format");
rc = dm_DASDWrite(index_h, "COMMENT: This file has been generated by the xenos process");
dt = DATE('N');
ts = TIME('N');
rc = dm_DASDWrite(index_h, "COMMENT:" dt ts );
rc = dm_DASDWrite(index_h, "COMMENT:" );
rc = dm_DASDWrite(index_h, "CODEPAGE:500" );
```

Figure 45. Sample Script File for the JSMAIN Program (Part 1 of 4)

```

dlpage = dm_GetDLPage(par_h);

do while(dlpage \= 'EOF')
  if file_open = FALSE then do
    select
      when generator = 'PDF' then
        rc = dm_PDFGenOpen(gen_h, '{GROUPFILEENTRY}' "DSN:"outputfile);
      when generator = 'AFP' then
        rc = dm_AFPGenOpen(gen_h, '{GROUPFILEENTRY}' "DSN:"outputfile);
      when generator = 'META' then
        rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}' "DSN:"outputfile);
      otherwise do
        say 'Invalid generator'
        return 12
      end
    end
    if rc = 0 then do
      file_open = TRUE
    end
  end

  do i = 1 to NumberOfFields
    fieldvalue.i = dm_GetText( dlpage, field.i, First )
  end

  docbreak = 0
  do i = 1 to NumberOfFields
    if fieldvalue.i \= "" then do
      /* if there is no previous value, save the current value */
      if fieldvaluesave.i = "" then do
        fieldvaluesave.i = fieldvalue.i
      end
    else
      /* if there is a previous value, see if the new value is different */
      if fieldvaluesave.i \= fieldvalue.i then do
        if Break.i = "yes" then
          docbreak = 1
        end
      end
    end
  end
end
end

```

Figure 45. Sample Script File for the JSMAIN Program (Part 2 of 4)

```

if docbreak = 1 then do
  select
    when generator = 'PDF' then rc = dm_PDFGenClose( gen_h )
    when generator = 'AFP' then rc = dm_AFPGenClose( gen_h )
    when generator = 'META' then rc = dm_METAGenClose( gen_h )
  end
  file_open = FALSE

  /* write out index values to the index file */
  do i = 1 to NumberOfFields
    field_name = "GROUP_FIELD_NAME:" || field.i
    rc = dm_DASDWrite( index_h, field_name )
    field_value = "GROUP_FIELD_VALUE:" || fieldvaluesave.i
    rc = dm_DASDWrite( index_h, field_value )
  end

  /* replace index values with the new values */
  do i = 1 to NumberOfFields
    if fieldvalue.i \= "" then do
      fieldvaluesave.i = fieldvalue.i
    end
  end

  rc = dm_DASDSize(dasd_h)
  BytesWritten = dm_size
  length = BytesWritten - save_BytesWritten
  offset = BytesWritten - length
  save_BytesWritten = BytesWritten

  group_offset = "GROUP_OFFSET:" || offset
  rc = dm_DASDWrite( index_h, group_offset )
  group_length = "GROUP_LENGTH:" || length
  rc = dm_DASDWrite( index_h, group_length )
  group_filename = "GROUP_FILENAME:" || outputfile
  rc = dm_DASDWrite( index_h, group_filename )

  select
    when generator = 'PDF' then
      rc = dm_PDFGenOpen(gen_h, '{GROUPFILEENTRY}'"DSN:"outputfile);
    when generator = 'AFP' then
      rc = dm_AFPGenOpen(gen_h, '{GROUPFILEENTRY}'"DSN:"outputfile);
    when generator = 'META' then
      rc = dm_METAGenOpen(gen_h, '{GROUPFILEENTRY}'"DSN:"outputfile);
  end
end

```

Figure 45. Sample Script File for the JSMAIN Program (Part 3 of 4)

```

        if rc = 0 then do
            file_open = TRUE
        end
    end /* end docbreak = 1 */

select
    when generator = 'PDF' then rc = dm_PDFGenWrite(gen_h, dlpage );
    when generator = 'AFP' then rc = dm_AFPGenWrite(gen_h, dlpage );
    when generator = 'META' then rc = dm_METAGenWrite(gen_h, dlpage );
end

dlpage = dm_GetDLPage(par_h);
end

if file_open = TRUE then do
    select
        when generator = 'PDF' then rc = dm_PDFGenClose( gen_h )
        when generator = 'AFP' then rc = dm_AFPGenClose( gen_h )
        when generator = 'META' then rc = dm_METAGenClose( gen_h )
    end
end

/* write out final index values to the index file */
do i = 1 to NumberOfFields
    field_name = "GROUP_FIELD_NAME:" || field.i
    rc = dm_DASDWrite( index_h, field_name )
    field_value = "GROUP_FIELD_VALUE:" || fieldvaluesave.i
    rc = dm_DASDWrite( index_h, field_value )
end

rc = dm_DASDSize(dasd_h)
BytesWritten = dm_size
length = BytesWritten - save_BytesWritten
offset = BytesWritten - length
save_BytesWritten = BytesWritten

group_offset = "GROUP_OFFSET:" || offset
rc = dm_DASDWrite( index_h, group_offset )
group_length = "GROUP_LENGTH:" || length
rc = dm_DASDWrite( index_h, group_length )
group_filename = "GROUP_FILENAME:" || outputfile
rc = dm_DASDWrite( index_h, group_filename )

rc = dm_DASDClose( dasd_h )
rc = dm_DASDClose( index_h )
return;

```

Figure 45. Sample Script File for the JSMAIN Program (Part 4 of 4)

Generic index file

Figure 46 on page 215 shows the Generic index file that was generated by the sample script file for this example.

```
COMMENT: OnDemand Generic Index File Format
COMMENT: This file has been generated by the xenos process
COMMENT: 20 Aug 2001 14:17:40
COMMENT:
CODEPAGE:500
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:Ward T. Jones, MD.
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:0030-334-33
GROUP_OFFSET:0
GROUP_LENGTH:2854891
GROUP_FILENAME:XXXXXXXXX.SAMPLOUT
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:Mike R. Smith.
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:0333-888-45
GROUP_OFFSET:2854891
GROUP_LENGTH:3002685
GROUP_FILENAME:XXXXXXXXX.SAMPLOUT
GROUP_FIELD_NAME:Name
GROUP_FIELD_VALUE:Barbara L. Schuster
GROUP_FIELD_NAME:Policy
GROUP_FIELD_VALUE:4567-001-77
GROUP_OFFSET:5857576
GROUP_LENGTH:3152841
GROUP_FILENAME:XXXXXXXXX.SAMPLOUT
```

Figure 46. Generic Index File Generated by JSMAIN Program Script

Part 5. OS/390 indexer reference

This part of the book provides information about the OS/390 indexer. You can use the OS/390 indexer to extract index data from and generate index data about line data and AFP reports. In addition, other data types, such as TIF images, can be captured using the Anystore Exit.

Chapter 27. Overview

Note: Statements in this section about the OS/390 indexer in OS/390 apply equally to the OS/390 indexer in the z/OS environment.

The OS/390 indexer extracts indexes and stores documents all in one pass of reading the input data. The OS/390 indexer indexes reports based on the organization of the data in the report. The OS/390 indexer processes two input sources:

- Indexing parameters that specify how the data should be indexed. You can create the indexing parameters when you define an OnDemand application. The parameters are of the same form as used by ACIF, along with some extensions which are unique to the OS/390 indexer.
- The print data stream.

The OS/390 indexer indexes input data based on the organization of the data:

- AFP reports. For AFP reports, the index values are already specified within the AFP data stream.
- Document organization. For reports made up of logical items, such as statements, policies, and invoices. The OS/390 indexer can generate index data for each logical item in the report.
- Report organization. For reports that contain line data with sorted values on each page, such as a transaction log or general ledger. The OS/390 indexer can divide the report into groups of pages and generate index data for each group of pages.
- Anystore Exit. This exit point allows for generating your own index values for each document.
- Large Object. Large object support is designed to provide enhanced usability and better retrieval performance for reports that contain very large documents by segmenting the documents into groups of pages and downloading only the page groups that the users request to view.

Before you can index a report with the OS/390 indexer, you must create a set of *indexing parameters*. The indexing parameters describe the physical characteristics of the input data, identify where in the data stream that the OS/390 indexer can locate index data, and provide other directives to the OS/390 indexer. Collecting the information needed to develop the indexing parameters requires a few steps. For example:

1. Examine the input data to determine how users use the report, including what information they need to retrieve a report from the system (indexing requirements).
2. Create parameters for indexing.

You run the OS/390 indexer as part of the OnDemand load process with the ARSLOAD program. The OnDemand application retrieves the indexing parameters from the OnDemand database and uses the parameters to process the input data.

The OS/390 indexer can logically divide reports into individual items, such as statements, policies, and bills. You can define up to 32 index fields for each item in a report.

Chapter 28. Indexing concepts

Indexing parameters include information that allow the OS/390 indexer to identify key items in the input data stream so they can be extracted from the report and stored in the OnDemand database. OnDemand uses these index values for efficient, structured search and retrieval.

The OS/390 indexer makes use of three different methods for determining the index values to use for each document within a report.

1. **AFP Reports.** The OS/390 indexer can capture fully resolved AFP data streams (AFPDS). The AFPDS must contain the index values either in the form of TLE or NOP records. See the INDEXSTYLE=AFP parameter on page 233 for details on these record types.

2. **Line Print Reports.** Line Print Reports consist of text formatted print streams. Column one of each record contains a carriage control character.

You specify the index information that allows the OS/390 indexer to segment the print 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. The OS/390 indexer 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.

An indexing parameter is made up of an *attribute name* (for example, Customer Name) and an *attribute value* (for example, Earl Hawkins). The parameters include pointers that tell the OS/390 indexer where to locate the attribute information in the data stream. For example, the tag Account Number with the pointer 1,21,16 means that the OS/390 indexer can expect to find Account Number values starting in column 21 of specific input records. The OS/390 indexer collects 16 bytes of information starting at column 21 and adds it to a list of attribute values found in the input. For each group that is identified by the OS/390 indexer, a set of index values that are associated with the group are stored by the OnDemand load process into the OnDemand database.

3. **Anystore Exits.** The use of an Anystore Exit allows for the capture of any type of data. The exit is responsible for reading the data to be captured, breaking it into documents, and determining the index values. A sample Anystore Exit is provided which captures TIF images using a pre-generated set of indexing instructions read from a separate file.
4. **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 many 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. Large Objects are only supported for Line data report types. To enable large object support, you must specify INDEXOBJ=ALL in the indexing parameters.

Chapter 29. OS/390 indexer parameters

To provide the segmentation and indexing instructions, the OS/390 indexer process uses standard ACIF parameters, as described in Part 1, “ACIF reference,” on page 1, as well as some parameters unique to the OS/390 indexer. This allows you to define the indexing parameters by using the graphical indexer from the administration client.

ACIF parameters used with the OS/390 indexer

The following subset of ACIF parameters are recognized by the OS/390 indexer process. See Chapter 3, “ACIF parameter reference,” on page 49 for detailed descriptions of these parameters.

Note: Other ACIF parameters can be specified, however they are ignored by the OS/390 indexer.

Important: The only code page supported by the OS/390 indexer is code page 500. All index values provided by the OS/390 indexer are assumed to be in code page 500. When loading data using the OS/390 indexer, the locale must be set appropriately for code page 500.

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 eight triggers.

INDEX

Identifies the index name and the field on which the index is based. You must define at least one index parameter. You can define up to 32 index parameters.

Notes® for INDEX usage:

- Only one field can be specified for each index.

- Only *group* and *grouprange* indexes are supported.

- Grouprange indexes can only be used with INDEX1 or INDEX2.

- Group indexes can be used with any of the 32 possible indexes.

FIELD

Identifies the location of index data. You must define at least one field. You can define up to 32 fields.

Note: Neither *default* nor *constant* field values are supported.

GROUPMAXPAGES

Determines the maximum number of pages to be put into a group. This allows the OS/390 indexer to logically segment a large report into groups of pages and create indexes for each group. You can specify a number from 1 (one) and 9999. Use this parameter when specifying a grouprange index. If no value is specified, the defaults is 100 (pages).

This parameter can only be used when the INDEXSTYLE is not set to AFP and when the ANYEXIT is not specified.

FILEFORMAT

Used by the OS/390 indexer to identify the logical record length of the input file. Only the record format of this parameter is supported:

FILEFORMAT=RECORD,n

Where n is the logical record length value.

This parameter can only be used when the INDEXSTYLE is not set to AFP and when the ANYEXIT is not specified.

INDEXOBJ

Specifies that documents are to be stored as large objects. This is indicated by setting INDEXOBJ=ALL (which is the only valid setting for the OS/390 indexer). This parameter is automatically included when the Large Object option is selected on the application Load Information page (by using the administrative client). If the documents are not to be stored as large objects, then this parameter is omitted. This parameter is only recognized by the OS/390 indexer when capturing line print data. It can optionally be used in combination with an Input Exit or an Anystore Exit, as long as the data being captured is line print data.

Other Parameters used with the OS/390 indexer

The following parameters are unique to the OS/390 indexer and are described here.

INPEXIT

Overview

The Input Exit is provided to allow additional processing of the report input before the report is stored. This exit can only be used when the INDEXSTYLE is not set to AFP and when the ANYEXIT is not specified. The exit is called dynamically during the report capture process. The report capture routine calls the exit when the indexing parameters specify an input exit name in the INPEXIT parameter. The report administrator provides a program name for this parameter.

There are no restrictions as to the type of processing that can be performed in an input exit with the exception that the exit must pass the standard parameter list back to the report capture program. Values must be supplied for all parameters.

A sample exit is provided in ARSEXINP.

Parameters

Figure 47 on page 225 shows the parameters that are required by ARSEXINP.


```

01 PAGE-BUFFER.
  02 PAGE-BUFFER-LINE OCCURS 256 TIMES.
    05 PAGE-BUFFER-CHAR PIC X(512).
01 LINE-COUNT COMP PIC S9(8).
01 RECORD-STATUS PIC X(3).
  88 END-OF-FILE VALUE 'EOF '.
01 ARSEXINP-DDNAME PIC X(8).

```

Figure 47. ARSEXINP Parameter List from ARSINPBK Cobol Copybook

Parameter Description

PAGE-BUFFER	Set by Exit. One data page of the report input. The buffer dimensions are: 512 characters wide and 256 lines per page. The dimensions must not be changed.
LINE-COUNT	Set by Exit. The number of lines in the Page Buffer.
RECORD-STATUS	Set by Exit. The end of file (EOF) indicator.
ARSEXINP-DDNAME	Set by calling routine. The DD Name of the input file. This is particularly useful when the load process is initiated by the Spool Capture process, because the DD Name will be different each time.

Notes

The exit processes two files. The report input file, called OBJINPT in the sample exit, and an error message file, called INPSTATS.

The sample exit routine performs the following: open the input file, read the input file, build the page buffer, perform report unique processing, and return control to the report capture program. All processing is controlled by the MAINLINE section of the exit. Changes related to file control processing may be required. **Care should be taken when changing the routines that build the page buffer. Do not alter the size of the INPUT-DATALINE.**

The sample code found in paragraph REPORT-UNIQUE-LOGIC is an example of the kind of processing that can be performed in an input exit. **The REPORT-UNIQUE-LOGIC routine must be customized or removed for the exit to function properly with your report input.**

Developing an input exit

The following approach is recommended for developing an input exit:

1. Create report definition with an input exit name specified in the INPEXIT parameter.
2. Create a new member in the Sample library corresponding to the input exit name.
3. Copy the sample exit into the new member.
4. Modify the exit as required.
5. Compile and link the exit.
6. Test.

INDEXEXIT

Overview

The Index Exit is provided to allow the report indexes to be modified prior to insertion into the Application Group Data Table. This exit can be used with any type of report captured by the OS/390 indexer. The exit is called dynamically during the capture process. The capture program calls the exit when the indexing instructions for the application include the INDEXEXIT parameter. The report administrator provides a program name for the Index Exit.

There are no restrictions as to the type of processing that can be performed in an index exit with the exception that the exit must pass the standard parameter list back to the capture program. A sample exit is provided in member ARSEXNDX.

Parameters

Figure 48 shows the parameters that are required by ARSEXNDX.

```
01 WS-HEADER-RECORD.
   05 FIRST-FIVE                      PIC X(5).
   05 WS-KEY-IND                      PIC X(3).
   05 WS-FIELDS.
       10 WS-FIELD OCCURS 32 TIMES    PIC X(256).
   05 WS-APPLICATION-NAME            PIC X(60).
```

Figure 48. ARSEXNDX Parameter List from ARSINDBK Cobol Copybook

Parameter Description

FIRST-FIVE	Set by capture program . Control field.
WS-KEY-IND	Set by capture program . Control field.
WS-FIELD	Set by capture program . The value for the 32 Keys extracted from the report input. The exit may return modified values.
WS-APPLICATION-NAME	Set by capture program . The value of the Application Name for this report.

Notes

All parameter fields are set before the Index Exit is called. The exit can alter the WS-FIELD array.

The sample exit routine performs the following: receives the parameter list, performs report unique index modifications and returns control to the capture process. All processing is controlled by the MAINLINE section of the exit. Error messages are written to file INDSTATS.

The sample code performed in paragraph REPORT-UNIQUE-LOGIC is an example of the kind of processing that may be performed in an index exit. The REPORT-UNIQUE-LOGIC routine must be customized or removed for the exit to function properly with your report.

Developing an index exit

The following approach is recommended for developing an index exit:

- Create report definition with an index exit name specified in the INDEXEXIT parameter.

- Create a new member in the Sample library corresponding to the index exit name.
- Copy the ARSEXNDX sample exit into the new member.
- Modify exit as required.
- Compile and link the exit.
- Test.

ANYEXIT

Overview

The Anystore Batch Capture Exit can be used to provide all segment and index data to the report capture program. This exit program is called from the report capture process.

The exit is called dynamically during the capture process. The capture program calls the exit when the indexing instructions for the application include the ANYEXIT parameter. The report administrator provides a program name for the Anystore Exit.

The report capture program expects the Anystore exit to pass back all segment data and the associated index information. The capture program will perform only the data management functions required for the capture process (document compression, document store, index management and store, etc.)

A sample Anystore exit program is provided in the ARSEXANY member.

Parameters

Figure 49 shows the parameters that are required by ARSEXANY. The parameters can be found in the Cobol Copybook member ARSANYBK.

```

01 ANY-HEADER-RECORD.
   02 ANY-APPLICATION-NAME          PIC X(60).
   02 FILLER.
      05 ANY-FIELD OCCURS 32 TIMES   PIC X(256).
   02 FILLER.
      05 ANY-DISPL OCCURS 32 TIMES   PIC S9(4) COMP.
   02 FILLER.
      05 ANY-LEN  OCCURS 32 TIMES    PIC S9(4) COMP.
   02 ANY-DDNAME                     PIC X(8).

01 ANY-DOC-SPACE.
   02 CURRENT-DOC-BUFFER OCCURS 1048320 PIC X.

01 ANY-DOC-SIZE                      COMP PIC S9(8).

01 ANY-STATUS                       PIC XXX.
   88 ANY-EOF                        VALUE 'EOF'.
   88 ANY-OVERFLOW                   VALUE 'OVF'.
   88 ANY-INDEX                      VALUE 'IDX'.
   88 ANY-SEG                        VALUE 'END'.
   88 ANY-ERR                        VALUE 'ERR'.

```

Figure 49. ARSEXANY Parameter List from ARSANYBK Cobol Copybook

Parameter Description

ANY-APPLICATION-NAME Set by capture program. The value of the Application Name.

ANY-FIELD	Set by exit. The values for the 32 Keys.
ANY-DISPL	Set by capture program. The column displacement value. This field represents an array of values that correspond with the Field parameter value from the Indexing Parameters.
ANY-LEN	Set by capture program. The field length value. This field represents an array of values that correspond with the Field parameter value from the Indexing Parameters.
ANY-DDNAME	Set by capture program. The DD Name of the input data file. This is particularly useful when the load process is initiated by the Spool Capture facility, because the DD Name changes each time.
ANY-DOC-SPACE	Set by exit. The segment buffer to be passed to the capture program. Maximum size of 1048320 allowed. (If larger sizes are required, the exit must break them into multiple physical segments no larger than 1048320.)
ANY-DOC-SIZE	Set by exit. The size of the current segment that is to be passed to the capture program.
ANY-STATUS	<p>Set by exit. Processing status to be passed back to the capture program.</p> <p>Possible values are:</p> <p>EOF End of run, no data nor indexes returned</p> <p>OVF Data returned, with more to follow; no indexes returned</p> <p>IDX Indexes returned, no data</p> <p>END Data and indexes returned, end of this document</p> <p>ERR An unexpected error occurred, end the run</p>

Sample scenarios

- Document is .5 MB in size, one set of indexes

ANY-STATUS = END	Both the data and the index values are returned.
-------------------------	--
- Document is .5 MB in size, three sets of indexes for this document

ANY-STATUS = END	Data and first set of indexes returned
ANY-STATUS = IDX	No data, second set of indexes returned
ANY-STATUS = IDX	No data, third set of indexes returned
- Document is 2.5 MB in size, one set of indexes

ANY-STATUS = OVF	First MB of data returned, no indexes returned
ANY-STATUS = OVF	Second MB of data returned, no indexes returned
ANY-STATUS = END	Remainder of data and indexes returned
- Document is 2.5 MB in size, three sets of indexes for this document

ANY-STATUS = OVF	First MB of data returned, no indexes returned
-------------------------	--

ANY-STATUS = OVF	Second MB of data returned, no indexes returned
ANY-STATUS = END	Remainder of data returned, first set of indexes returned
ANY-STATUS = IDX	No data returned, second set of indexes returned
ANY-STATUS = IDX	No data returned, third set of indexes returned

Eventually, an ANY-STATUS of EOF is returned. At that time, no data nor any indexes are returned.

Developing an ANYSTORE exit

The sample exit routine stores TIFF data and also creates a second index (for illustration purposes) for each logical document. The following approach is recommended for developing an Anystore exit:

- Create report definition with an Anystore exit name specified in the ANYEXIT parameter.
- Create a new member in the Sample library corresponding to the Anystore exit name.
- Copy the ARSEXANY sample exit into the new member.
- Modify exit as required.
- Compile and link the exit.
- Test.

A minimal set of indexing instructions can be provided for Applications using an ANYSTORE exit. Figure 50 shows an example.

```

TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'D7C5D9E2D6D5C1D3D5E4D4C2C5D9',FIELD1,(TYPE=GROUP,BREAK=YES) /* PERSONALNUMBER */
INDEX2=X'D9C5C7C9D6D5',FIELD1,(TYPE=GROUP,BREAK=NO) /* REGION */
INDEX3=X'C4C1E3E4D4',FIELD1,(TYPE=GROUP,BREAK=NO) /* DATUM */
INDEX4=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD1,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
ANYEXIT=ARSEXANY

```

Figure 50. Indexing instructions for Applications using an ANYSTORE exit

The TRIGGER and FIELD parameters are needed to meet the syntax checking requirements of the graphical indexer, but are not used by the OS/390 Indexer. The INDEX names are required to match the Application Group field names. This allows each index value to be mapped to the correct Application Group data table column.

The exit is responsible for ensuring the data returned for the index values match the expected data types.

INDEXSTYLE

Overview

This parameter exists to aid in the migration of customers from OnDemand for OS/390 Version 2.1 to OnDemand for z/OS and OS/390, Version 7.1. However, this parameter may be used by any customer. If this parameter is specified, certain rules must be followed regarding the indexes defined in the OnDemand application group to which the OnDemand application is associated.

Parameters

Valid values and the rules associated with them follow:

DOC DOC reports are traditional document reports, such as statements, invoices, and so forth. No indexes of type GROUPRANGE can be specified.

Figure 51 lists typical indexing parameters and values for DOC reports. Indexing parameters are specified on the Indexer Information page in the OnDemand application.

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,133
TRC=YES
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=*,3,X'D7C1C7C540F140',(TYPE=GROUP) /* PAGE 1 */
TRIGGER3=*,73,X'C1C3C3D6E4D5E3',(TYPE=FLOAT) /* ACCOUNT */
TRIGGER4=*,3,X'C3D6D5E3C5D5E3E2',(TYPE=FLOAT) /* CONTENTS */
TRIGGER5=*,19,X'C5D5C4C9D5C740C2C1D3',(TYPE=FLOAT) /* ENDING BAL */
FIELD1=-1,89,9,(TRIGGER=3,BASE=0)
FIELD2=1,87,11,(TRIGGER=4,BASE=0)
FIELD3=7,19,12,(TRIGGER=1,BASE=0)
FIELD4=0,87,16,(TRIGGER=5,BASE=0)
FIELD5=0,37,8,(TRIGGER=5,BASE=0)
INDEX1=X'C1C3C3D6E4D5E36DD5E4D4C2C5D9',FIELD1,(TYPE=GROUP,BREAK=YES) /* ACCOUNT_NUMBER */
INDEX2=X'E2E2D56D6D6DE3C1E76DC9C4',FIELD2,(TYPE=GROUP,BREAK=NO) /* SSN_TAX_ID */
INDEX3=X'C3E4E2E36DD5C1D4C5',FIELD3,(TYPE=GROUP,BREAK=NO) /* CUST_NAME */
INDEX4=X'C5D5C4C9D5C76DC2C1D3C1D5C3C5',FIELD4,(TYPE=GROUP,BREAK=NO) /* ENDING_BALANCE */
INDEX5=X'C3D3D6E2C56DC4C1E3C5',FIELD5,(TYPE=GROUP,BREAK=NO) /* CLOSE_DATE */
INDEX6=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD5,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=DOC
```

Figure 51. Typical indexing parameters and values for DOC reports

PAGE PAGE reports are *transaction* type reports. The entire report is sorted by some column value. This sort key is used in the first and second indexes. The GROUPMAXPAGES parameter can be used to determine the number of pages included in each segment. If no GROUPMAXPAGES value is specified, the default is 100 (pages).

The indexes defined for the OnDemand application group must be as follows:

First Index Must be the start value for a GROUPRANGE index.

Second Index Must be the end value for the GROUPRANGE index.

Third Index Must be the start value for a GROUPRANGE index on Page Number. Must be defined as an integer.

Note: The value for the third index is set by the OS/390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Fourth Index Must be the end value for the GROUPRANGE index on Page Number. Must be defined as an integer.

Note: The value for the fourth index is set by the OS/390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Additional indexes

May be defined, but cannot be GROUPRANGE indexes.

Figure 52 lists typical indexing parameters and values for PAGE reports. Indexing parameters are specified on the Indexer Information page in the OnDemand application.

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=0,2,X'D9C5D7D6D9E3',(TYPE=GROUP) /* REPORT */
FIELD1=*,*,10,(OFFSET=(3:12),MASK='#####',ORDER=BYROW)
FIELD2=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'D3D6C1D56DD5E4D4C2C5D9',FIELD1,(TYPE=GROUPRANGE) /* LOAN_NUMBER */
INDEX2=X'D7C1C7C56DD5D66D',FIELD1,(TYPE=GROUPRANGE) /* PAGE_NO */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD2,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=PAGE
```

Figure 52. Typical indexing parameters and values for PAGE reports

Notes:

1. The **FIELD_n** value pointed to by the INDEX2 Page Number index is needed to meet the syntax checking requirements of the graphical indexer, but is not used by the OS/390 indexer. Any valid **FIELD_n** value may be specified for INDEX2.
2. The GROUPRANGE specification for INDEX2 will cause the beginning page number value to be stored in the third application group index field, while the ending page number value gets stored in the fourth application group index field. INDEX3 (posting date from above) gets stored in the fifth application group index field.

PDOC PDOC reports are transaction type reports, but have a high level index. For example, a bank may have a report that is organized by Branch Number. Within each Branch, the report is sorted on some column. The GROUPMAXPAGES parameter can be used to determine the number of pages included in each segment. If no GROUPMAXPAGES value is specified, the default is 100 (pages). A new segment is started when either the high level index changes or the GROUPMAXPAGES value is reached.

The indexes defined for the OnDemand application group must be as follows:

- | | |
|---------------------|--|
| First Index | Must be the high level index. |
| Second Index | Must be the start value for the GROUPRANGE index by which the report is sorted within the first index. |
| Third Index | Must be the end value for the GROUPRANGE index by which the report is sorted within the first index. |

Additional indexes

May be defined, but cannot be GROUPRANGE indexes.

Figure 53 on page 232 lists typical indexing parameters and values for PDOC reports. Indexing parameters are specified on the Indexer Information page in the OnDemand application.


```

CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=100
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
TRIGGER2=*,2,X'C2C1D5D2',(TYPE=GROUP) /* BANK */
FIELD1=1,11,3,(TRIGGER=1,BASE=0)
FIELD2=*,*,10,(OFFSET=(3:12),MASK='#####',ORDER=BYROW)
FIELD3=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'C2C1D5D26DC2D9C1D5C3C8',FIELD1,(TYPE=GROUP,BREAK=YES) /* BANK_BRANCH */
INDEX2=X'D3D6C1D56DD5E4D4C2C5D9',FIELD2,(TYPE=GROUPRANGE) /* LOAN_NUMBER */
INDEX3=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD3,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=PDOC

```

Figure 53. Typical indexing parameters and values for PDOC reports

Note: The GROUPRANGE specification for INDEX2 will cause the beginning loan number value to be stored in the second application group index field, while the ending loan number value gets stored in the third application group index field. INDEX3 (posting date from above) gets stored in the fourth application group index field.

NODX

NODX (no index) reports are ones which either do not have obvious index values, or which are very short and do not need to be broken up into documents. The GROUPMAXPAGES parameter can be used to determine the number of pages included in each segment. If no GROUPMAXPAGES value is specified, the default is 100 (pages).

The indexes defined for the OnDemand application group must be as follows:

First Index Must be defined as Segment Number. Must be defined as an integer. The OS/390 indexer assigns values to this index by sequentially counting each segment (document) as it is created.

Second Index Must be defined as Report Date. Must be defined as a string of length 8 (eight). The OS/390 indexer assigns this value, based on the Posting Date of the report. The value will have the format of MM/DD/YY.

Note: A separate index defined on Posting Date with a data type of DATE should also be defined. This index should be marked as being the *segment field* for the application group.

Third Index Must be the start value for a GROUPRANGE index on Page Number. Must be defined as an integer.

Note: The value for the third index is set by the OS/390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Fourth Index Must be the end value for the GROUPRANGE index on Page Number. Must be defined as an integer.

Note: The value for the fourth index is set by the OS/390 indexer by counting the pages as they are stored, not from values extracted from the report data.

Additional indexes

May be defined, but cannot be GROUPRANGE indexes.

Figure 54 lists typical indexing parameters and values for NODX reports. Indexing parameters are specified on the Indexer Information page in the OnDemand application.

```
CC=YES
CCTYPE=A
FILEFORMAT=RECORD,90
GROUPMAXPAGES=50
TRIGGER1=*,1,X'F1',(TYPE=GROUP) /* 1 */
FIELD1=0,83,8,(TRIGGER=1,BASE=0)
INDEX1=X'E2C5C7D4C5D5E36DD5E4D4C2C5D9',FIELD1,(TYPE=GROUP,BREAK=NO) /* SEGMENT_NUMBER */
INDEX2=X'D9C5D7D6D9E36DC4C1E3C5',FIELD1,(TYPE=GROUP,BREAK=NO) /* REPORT_DATE */
INDEX3=X'D7C1C7C56DD5E4D4C2C5D9',FIELD1,(TYPE=GROUPRANGE) /* PAGE_NUMBER */
INDEX4=X'D7D6E2E3C9D5C76DC4C1E3C5',FIELD1,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE */
INDEXSTYLE=NODX
```

Figure 54. Typical indexing parameters and values for NODX reports

Notes:

1. The **FIELD n** value pointed to by the INDEX1, INDEX2, and INDEX3 indexes are needed to meet the syntax checking requirements of the graphical indexer, but are not used by the OS/390 Indexer. Any valid **FIELD n** value may be specified for these indexes.
2. The GROUPRANGE specification for INDEX3 will cause the beginning page number value to be stored in the third application group index field, while the ending page number value gets stored in the fourth application group index field. INDEX4 (posting date from above) gets stored in the fifth application group index field.

AFP AFP (Advanced Function Printing) reports captured via the OS/390 indexer must already have been formatted into an AFP Data Stream (AFPDS). This can be done by using ACIF (AFP Conversion and Indexing Facility) or by any third party program. The OS/390 indexer looks for index values within the AFPDS, either in TLE or NOP records. ACIF, and other programs, can automatically generate the TLE records. The NOP records for use by the OS/390 indexer have a fixed format.

For details on the TLE record formats, refer to the *MO:DCA Reference*, SC31-6802.

The NOP record format provides space for 32 indexes of up to 256 characters each. One ODZOSSEG record must exist for each document. This identifies that a new document is starting, and provides one complete set of index values for the document.

One or more ODZOSDIR records can exist for each document. This record provides an additional set of index values for the document.

Note: NOP record formats for earlier releases of OnDemand for OS/390 and R/DARS-ESA are supported by the OS/390 indexer for compatibility purposes.

Table 6 on page 234 shows the layout of the ODZOSSEG and ODZOSDIR NOP record.

Table 6. Layout of the ODZOSSEG and ODZOSDIR NOP record

Position	Description
1	X'5A'
2 - 3	Length of this record - 1
4 - 6	X'D3EEEE'
7 - 9	X'000000'
10 - 17	'ODZOSSEG' or 'ODZOSDIR'
18 - 273	Value of field 1 (256 bytes)
274 - 529	Value of field 2 (256 bytes)
530 - 785	Value of field 3 (256 bytes)
786 - 1041	Value of field 4 (256 bytes)
1042 - 1297	Value of field 5 (256 bytes)
1298 - 1553	Value of field 6 (256 bytes)
1554 - 1809	Value of field 7 (256 bytes)
1810 - 2065	Value of field 8 (256 bytes)
2066 - 2321	Value of field 9 (256 bytes)
2322 - 2577	Value of field 10 (256 bytes)
2578 - 2833	Value of field 11 (256 bytes)
2834 - 3089	Value of field 12 (256 bytes)
3090 - 3345	Value of field 13 (256 bytes)
3346 - 3601	Value of field 14 (256 bytes)
3602 - 3857	Value of field 15 (256 bytes)
3858 - 4113	Value of field 16 (256 bytes)
4114 - 4369	Value of field 17 (256 bytes)
4370 - 4625	Value of field 18 (256 bytes)
4626 - 4881	Value of field 19 (256 bytes)
4882 - 5137	Value of field 20 (256 bytes)
5138 - 5393	Value of field 21 (256 bytes)
5394 - 5649	Value of field 22 (256 bytes)
5650 - 5905	Value of field 23 (256 bytes)
5906 - 6161	Value of field 24 (256 bytes)
6162 - 6417	Value of field 25 (256 bytes)
6418 - 6673	Value of field 26 (256 bytes)
6674 - 6929	Value of field 27 (256 bytes)
6930 - 7185	Value of field 28 (256 bytes)
7186 - 7441	Value of field 29 (256 bytes)
7442 - 7697	Value of field 30 (256 bytes)
7698 - 7953	Value of field 31 (256 bytes)
7954 - 8209	Value of field 32 (256 bytes)

Figure 55 lists typical indexing parameters and values for AFP reports using TLE records. Indexing parameters are specified on the Indexer Information page in the OnDemand application.

```

TRIGGER1=*,1,X'5A',(TYPE=GROUP) /* AFP x'5A' */
FIELD1=-0,1,14,(TRIGGER=1,BASE=0)
FIELD2=0,1,24,(TRIGGER=1,BASE=0)
FIELD3=0,1,18,(TRIGGER=1,BASE=0)
INDEX1=X'D796938983A8',FIELD1,(TYPE=GROUP,BREAK=YES) /* Policy */
INDEX2=X'C39695A38595A3A2',FIELD2,(TYPE=GROUP,BREAK=NO) /* Contents */
INDEX3=X'C995A2A4998584',FIELD3,(TYPE=GROUP,BREAK=NO) /* Insured */
INDEXSTYLE=AFP

```

Figure 55. Indexing parameters and values for AFP reports using TLE records

In Figure 55, the TRIGGER record is not used by the OS/390 indexer and is specified only to meet syntax checking requirements of the graphical indexer. The only value used from the FIELD records is the length value. The name fields of the INDEX values must match the Attribute Name field of the TLE records, and is used to map the index values back to the Application Group data table columns. The BREAK parameter of the INDEX record is not used.

Figure 56 lists typical indexing parameters and values for AFP reports using NOP records. Indexing parameters are specified on the Indexer Information page in the OnDemand application.

```

TRIGGER1=*,1,X'5A',(TYPE=GROUP) /* AFP x'5A' */
FIELD1=-0,1,15,(TRIGGER=1,BASE=0) /* Length of data to extract = 15 */
FIELD2=0,1,11,(TRIGGER=1,BASE=0) /* Length of data to extract = 11 */
FIELD3=0,1,8,(TRIGGER=1,BASE=0) /* Length of data to extract = 8 */
FIELD4=0,1,8,(TRIGGER=1,BASE=0) /* Length of data to extract = 8 */
INDEX1=X'F1',FIELD1,(TYPE=GROUP,BREAK=NO) /* CUST_NAME in NOP Field 1 */
INDEX2=X'F2',FIELD2,(TYPE=GROUP,BREAK=YES) /* ACCOUNT_NUM in NOP Field 2 */
INDEX3=X'F5',FIELD3,(TYPE=GROUP,BREAK=NO) /* REPORT_DATE in NOP Field 5 */
INDEX4=X'F6',FIELD4,(TYPE=GROUP,BREAK=NO) /* POSTING_DATE in NOP Field 6 */
INDEXSTYLE=AFP

```

Figure 56. Indexing parameters and values for AFP reports using NOP records

In Figure 56, the TRIGGER record is not used by the OS/390 indexer and is specified only to meet syntax checking requirements of the graphical indexer. The only value used from the FIELD records is the length value. The name fields of the INDEX values are character representations of numbers which point to the position within the NOP record where the index value is to be found for each index. The BREAK parameter of the INDEX record is not used.

Note: Special consideration is needed when dealing with the Posting Date field with the older style of NOP records.

- OnDemand for OS/390 V2.1 used NOP record types of OD390SEG and OD390DIR. The INDEX record for the Posting Date field must specify a name value of X'F1F7'. For example:

```

INDEX4=X'F1F7',FIELD4,(TYPE=GROUP,BREAK=NO) /*POSTING_DATE in OD390SEG NOP*/

```

- OnDemand for OS/390 V1.1 and R/DARS-ESA used NOP record types of RDARSSEG and RDARSDIR. The INDEX record for the Posting Date field must specify a name value of X'F6'. For example:

```
INDEX4=X'F6',FIELD4,(TYPE=GROUP,BREAK=NO) /*POSTING_DATE in RDARSSEG NOP*/
```

If the INDEXSTYLE parameter is not specified in the indexing instructions, it defaults to a value of DOC. The INDEXSTYLE parameter is ignored when the ANYEXIT parameter is specified.

Note: All application groups should have an index defined on Posting Date, with a data type of DATE. This index should be marked as being the *segment field* for the application group.

Chapter 30. Using the OS/390 indexer

The OnDemand application

You must specify to the ARSLOAD program that the OS/390 indexer process is to be used to capture a report. You specify the name of the indexer on the Indexer Information page in the OnDemand application. The name of the OS/390 indexer in the OnDemand application is OS/390. For example:

1. Start the administrative client.
2. Log on to the server.
3. Add an application.
4. Click on the Indexer Information tab.
5. On the Indexer Information page, specify OS/390 in the Indexer field. Figure 57 shows an example.

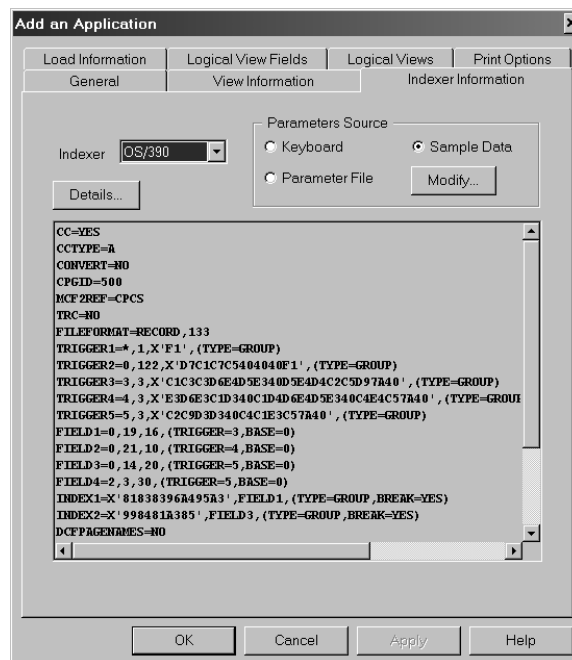


Figure 57. Specifying the OS/390 indexer on the Indexer Information page

The ARSLOAD program

When using JCL to run the ARSLOAD program to capture reports:

- Specify the `-s ddname` parameter to indicate the DD statement that points to the input report file that is being captured.
- Specify the name of a temporary file as the last parameter. The ARSLOAD program uses the temporary file for work space during the load process.

Figure 58 on page 238 shows an example of a parameter list for ARSLOAD. The DD statement that points to the input report file that is being captured is OBJINPT,

the name of the application group is CHKOPL1 01, and tempname is the name of the file that is used for temporary work space.

```
//      PARM=('/-u SYSADMIN -p SYSADMIN -h ARCHIVE -n -v -s OBJINPT  
//      -g "CHKOPL1 01" tempname')
```

Figure 58. Example of parameter list for ARSLOAD

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM	Domino.Doc	OS/2
ADSTAR	EDMSuite	OS/390
Advanced Function Presentation	Enterprise Storage Server	OS/400
Advanced Function Printing	FlowMark	Presentation Manager
AIX	ImagePlus	Print Services Facility
AIX/6000	Infoprint	RACF
AFP	Intelligent Printer Data Stream	RS/6000
AS/400	IPDS	S/390
BookManager	iSeries	SecureWay
CICS	Language Environment	SET
Cryptolope	Lotus	System/370
DataJoiner	Lotus Notes	Tivoli
DB2	MQSeries	Ultrastar
DB2 Universal Database	MVS	VideoCharger
DFSMSdfp	MVS/DFP	VisualInfo
DFSMS/MVS	Notes	WebSphere
Domino	OpenEdition	z/OS

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Adobe, the Adobe logo, Acrobat and the Acrobat logo are trademarks of Adobe Systems Incorporated, which may be registered in certain jurisdictions.

Portions of the OnDemand Windows client program contain licensed software from Pixel Translations Incorporated, © Pixel Translations Incorporated 1990, 2002. All rights reserved.

Other company, product or service names may be trademarks or service marks of others.

Index

A

accessing reports 13, 21, 32, 44

ACIF

about 3

AFP resources 8, 54, 61, 75, 76, 78, 80, 81, 83, 89

bycol option 59

byrow option 59

CC parameter 49, 102

CCTYPE parameter 49, 102

CHARS parameter 50

constant field 57

CONVERT parameter 51

CPGID parameter 52

DCFPAGENAMES parameter 53

default index value 57

EBCDIC data 10

error messages 93

examples 13

exit, index 97

exit, input file 95

exit, output record 98

exit, resource retrieval 100

exit, user programming 95

extended options 53

EXTENSIONS parameter 53

FDEFLIB parameter 54

field mask 59

field order 59

FIELD parameter 55

fields 18, 27, 38, 55

FILEFORMAT parameter 60

FONTLIB parameter 61

fonts 51

FORMDEF parameter 61, 103

group indexes 4, 9, 63, 65

GROUPMAXPAGES parameter 63

GROUPNAME parameter 64

IMAGEOUT parameter 64

index exits 97

INDEX parameter 65

index, exit 95

INDEXDD parameter 68

indexes 19, 28, 40, 65

indexing 95

indexing parameter reference 49

indexing parameters

about 5

example of 15, 24, 33, 46

INDEXOBJ parameter 69

INDEXSTARTBY parameter 70

INDEXEXIT parameter 70

inline resources 111

INPEXIT parameter 71

input exits 95

input file, exit 95

input, user exit 95

INPUTDD parameter 71

INSERTIMM parameter 72

introduction 3

Invoke Medium Map structured field 109

invoking program to index input file 129

ACIF (continued)

JCL statement defined 128

large object support 69

line data 6

LINECNT parameter 72

mask 90

mask option 59

MCF2REF parameter 73

message file 128

messages 93

MSGDD parameter 73

NEWPAGE parameter 74

order option 59

OS/390 JCL statement 127

OS/390 requirements 4, 127

OUTEXIT parameter 74

output file format 124

output record exits 98

OUTPUTDD parameter 75

overview 3

OVLYLIB parameter 75

page indexes 3, 10, 65

PAGEDEF parameter 76, 103

parameter file 128

parameter reference 49

parameters

about 5

example of 15, 24, 33, 46

parameters syntax 129

parameters, OS/390 129

parameters, z/OS 129

PARMDD parameter 78

PDEFLIB parameter 78

print file attributes 102

print file attributes, CC parameter 102

print file attributes, CCTYPE parameter 102

print file attributes, FORMDEF parameter 103

print file attributes, PAGEDEF parameter 103

print file attributes, PRMODE parameter 103

print file attributes, TRC parameter 103

PRMODE parameter 79, 103

program requirements 4, 127

PSEGLIB parameter 80, 89

reference 1

requirements 4, 127

RESEXIT parameter 81

RESFILE parameter 81

RESOBJDD parameter 82

resource provided with ACIF 100

resource retrieval 100

resources 8, 54, 61, 75, 76, 78, 80, 81, 83, 89, 111

RESTYPE parameter 83

syntax rules, OS/390 129

syntax rules, z/OS 129

TRACE parameter 85

transaction field 18, 58

TRC parameter 85, 103

trigger field 55

TRIGGER parameter 86

triggers 17, 26, 35, 86

UNIQUEBNGS parameter 89

- ACIF (*continued*)
 - usage 13
 - user exit input 95
 - user exit provided with ACIF 95
 - user exit, print file attributes 102
 - user programming exit 95
 - USERLIB parameter 89
 - USERMASK parameter 90
 - using 4, 127
 - z/OS JCL statement 127
 - z/OS requirements 4, 127
- ACIF command
 - See* ACIF
- Adobe PDF documents
 - See* PDF indexer
- ADOBEFNT JCL statement 173, 175, 177, 178
- ADOBERES JCL statement 173, 175, 177, 178
- Advanced Function Printing (AFP)
 - See* AFP
- AFP
 - about 6
 - converting line data to 6
 - converting with Xenos transform 181
 - example of converting line data to 21, 31
 - example of indexing 44
 - fonts 51, 61, 83
 - form definitions 54, 61
 - generic indexer, processing with 135
 - IMM structured fields 72
 - indexing data that contains TLEs 44
 - indexing with the generic indexer 135
 - line data
 - converting 6
 - MCF2 structured fields 73
 - overlays 75
 - page definitions 76, 78
 - page segments 80
 - processing with the generic indexer 135
 - processing with Xenos transform 181
 - resources 8, 54, 61, 75, 76, 78, 80, 81, 83, 89
 - Set Coded Font Local structured fields 79
 - SOSI 79
 - user-defined resources 89
- AFP API
 - See* AFP Application Programming Interface
- AFP Application Programming Interface
 - Tag Logical Element 113
- AFP Conversion and Indexing Facility (ACIF)
 - See* ACIF
- ANSI carriage-control characters
 - See* carriage-control characters
- ANYEXIT parameter
 - OS/390 indexer 227
- Anystore Batch Capture Exit
 - OS/390 indexer 227
- archiving, ACIF
 - indexing considerations 109, 110
- ARSACIF
 - See* ACIF
- ARSLoad program
 - specifying parameters for the OS/390 indexer 237
- ARSPDOCI
 - Adobe fonts DD statement 175
 - COORDINATES parameter 161
 - error messages 171
 - examples 173
 - FIELD parameter 161

- ARSPDOCI (*continued*)
 - INDEX parameter 164
 - INDEXDD parameter 165
 - INDEXSTARTBY parameter 166
 - INPUTDD parameter 167
 - JCL statement defined 173
 - messages 171
 - MSGDD parameter 167
 - OS/390 JCL statement 173
 - OUTPUTDD parameter 167
 - parameter file 175
 - parameter file defined 175
 - PARMDD parameter 168
 - reference 161
 - TRIGGER parameter 168
 - z/OS JCL statement 173
- ARSPDOCI command
 - See* ARSPDOCI
- ARSPDUMP
 - Adobe fonts DD statement 178
 - examples 177
 - JCL statement defined 177
 - OS/390 JCL statement 177
 - z/OS JCL statement 177
- attributes
 - print file 102

B

- Begin Document Index structured field
 - defined 118
- Begin Document structured field
 - defined 123
- Begin Named Group structured field
 - defined 123
- Begin Page structured field
 - defined 124
- Begin Resource Group structured field
 - described 116
- Begin Resource structured field
 - defined 116
- bookmarks
 - PDF indexer 159
- BTD
 - See* Begin Document structured field
- bycol option
 - FIELD parameter option 59
- byrow option
 - FIELD parameter option 59

C

- carriage controls 49, 72
- carriage-control characters
 - indexing considerations 110
- CC parameter
 - flags and values 49
 - print file attributes 102
 - user exits 102
- CCTYPE parameter
 - flags and values 49
 - print file attributes 102
 - user exits 102
- CHARS parameter
 - flags and values 50

- CMS commands
 - invoking ACIF program to index input file 129
- code page
 - ACIF 52
 - generic indexer 137
 - OS/390 indexer 223
 - PDF indexer 160
 - Xenos indexer 199
- CODEPAGE: parameter 137
- COMMENT: parameter 138
- comments
 - in parameter file 129
- Composed Text Control (CTC) structured field
 - obsolete 124
- concatenation
 - OS/390 files 131
 - resource group to document 110
 - z/OS files 131
- constant field 57, 163
- conversion 6, 51
- CONVERT parameter
 - flags and values 51, 72
- converting
 - AFP data to PDF 185
 - AFP to HTML 185
 - AFP to XML 185
 - Metacode to AFP 185
 - Metacode to HTML 185
 - Metacode to Metacode 185
 - Metacode to PDF 185
 - Metacode to XML 185
 - PCL to PDF 185
- converting line data to AFP 6
- coordinate system 150
- coordinates
 - on FIELD parameter for PDF indexer 162
 - on fieldbase parameter for Xenos transform 207
 - on TRIGGER parameter for PDF indexer 169
- COORDINATES parameter
 - flags and values 161
- CPGID parameter
 - flags and values 52

D

- data
 - format 60
- DCB characteristics
 - index object file 68
 - output file 75
 - resource group file 83
- DCB requirements
 - message file, OS/390 128
 - message file, z/OS 128
 - output file, OS/390 128
 - output file, z/OS 128
- DCFPAGENAMES parameter
 - flags and values 53
- default index value
 - FIELD parameter option 57, 163
- defining fields 18, 27, 38
- defining indexes 19, 28, 40
- defining triggers 17, 26, 35
- diagnostic trace information 85
- DOC reports and OS/390 indexer 230
- document
 - DD statement for ARSPDOCI, PDF indexer 174

- document (*continued*)
 - DD statement for ARSPDUMP, PDF indexer 178
 - DD statement for, OS/390 127
 - DD statement for, z/OS 127
 - generic indexer parameter 139, 141
 - INPUTDD parameter defined for ARSPDOCI, PDF indexer 176
 - inputfile parameter defined for JSMAIN, Xenos transform 205, 210
 - JCL parameter defined for ARSPDOCI, PDF indexer 174
 - output format 121

E

- EBCDIC data
 - CCTYPE parameter 49
 - CPGID parameter 52
 - example of 10
 - indexing 10
 - parameter file for PDF indexer 175
 - specifying 10
 - TRIGGER parameter 86, 168
 - USERMASK parameter 90
- EDI
 - See* End Document Index structured field
- EDT
 - See* End Document structured field
- End Document Index structured field
 - defined 120
- End Document structured field
 - defined 124
- End Named Group structured field
 - defined 124
- End Page structured field
 - defined 124
- End Resource Group structured field
 - defined 116
- End Resource structured field
 - defined 116
- ENG
 - See* End Named Group structured field
- EPG
 - See* End Page structured field
- ER
 - See* End Resource structured field
- ERG
 - See* End Resource Group structured field
- error messages
 - ACIF 93
 - ARSPDOCI program 171
 - PDF indexer 171
- examples 130, 131
 - AFP data, indexing 21, 31, 44
 - AFP document output formats 121
 - generic indexer 143
 - indexing 13, 21, 31
 - indexing data that contains TLEs 44
 - invoking ACIF program to index input file 129
 - JCL and ACIF processing parameters 129
 - JSMAIN 201, 207
 - line data, converting to AFP 21, 31
 - line data, indexing 13, 21, 31
 - OS/390 JCL to invoke ACIF 127
 - OS/390 JCL to invoke ARSPDOCI 173
 - OS/390 JCL to invoke ARSPDUMP 177
 - OS/390 JCL to invoke JSMAIN 202, 208
 - OS/390 JCL to invoke PDF indexer 173, 177

- examples (*continued*)
 - OS/390 JCL to invoke Xenos transform 202, 208
 - print file attributes 102
 - TLEs, indexing 44
 - Xenos transform 193, 201, 207
 - z/OS JCL to invoke ACIF 127
 - z/OS JCL to invoke ARSPDOCI 173
 - z/OS JCL to invoke ARSPDUMP 177
 - z/OS JCL to invoke JSMAIN 202, 208
 - z/OS JCL to invoke PDF indexer 173, 177
 - z/OS JCL to invoke Xenos transform 202, 208
- exits
 - index 97
 - input 95
 - output 98
 - print file attributes provided 102
 - resource, provided with ACIF 100
- extended options 53
- EXTENSIONS parameter
 - flags and values 53

F

- FDDLOUT JCL statement
 - for JSMAIN, Xenos transform 203
- fddloutput parameter
 - defined for JSMAIN, Xenos transform 205
- fddmscript parameter
 - defined for JSMAIN, Xenos transform 205, 210
- FDEFLIB
 - form definitions 62
- FDEFLIB parameter
 - flags and values 54
- fdjobparm parameter
 - defined for JSMAIN, Xenos transform 205, 210
- fdlicense parameter
 - defined for JSMAIN, Xenos transform 205, 211
- FIELD parameter
 - bycol option 59
 - byrow option 59
 - constant field 57, 163
 - default index value 57, 163
 - flags and values 55, 161
 - mask option 59, 162
 - order option 59
 - OS/390 indexer 223
 - transaction field 58
 - trigger field 55, 161
- fields
 - about 5, 150
 - ACIF parameter 55
 - bycol option 59
 - byrow option 59
 - constant field 57, 163
 - default index value 57, 163
 - defining 18, 27, 38
 - generic indexer parameter 138, 139
 - mask option 59, 162
 - order option 59
 - OS/390 indexer 223
 - PDF indexer parameter 161
 - transaction field 18, 58
 - trigger field 55, 161
 - Xenos transform 207
- file
 - message, ACIF 128
 - parameter, ACIF 128

- FILEFORMAT parameter
 - flags and values 60
 - OS/390 indexer 224
- files
 - format 60
 - OS/390 indexer 224
 - PDF indexer 159
- floating triggers 64, 88
- FONTLIB parameter
 - flags and values 61
- fonts
 - CHARS parameter 50
 - converting 73
 - DD statement for ARSPDOCI, PDF indexer 175
 - DD statement for ARSPDUMP, PDF indexer 178
 - directory 61
 - library 61
 - line data 51
 - location 61
 - Map Coded Font Format 2 structured fields 73
 - MCF2 structured fields 73
 - NLS 79
 - PDF indexer 159
 - resources 83
 - Set Coded Font Local structured fields 79
 - SOSI 79
 - specifying 50
 - TRCs 85
- form definitions 54, 61
- FORMDEF parameter
 - flags and values 61
 - print file attributes 103
 - user exits 103

G

- generic indexer
 - about 133, 135
 - AFP data, processing 135
 - application group field names 138
 - code page 137
 - CODEPAGE: parameter 137
 - COMMENT: parameter 138
 - document 139, 141
 - examples 143
 - field names 138
 - field values 139
 - group indexes, defining 138, 139
 - GROUP_FIELD_NAME: parameter 138
 - GROUP_FIELD_VALUE: parameter 139
 - GROUP_FILENAME: parameter 139
 - GROUP_LENGTH: parameter 141
 - GROUP_OFFSET: parameter 141
 - input file 139, 141
 - introduction 133
 - national language support (NLS) 137
 - NLS 137
 - overview 133
 - parameter file 143
 - parameters 137
 - using 133
- graphical indexer
 - PDF input files 153
- group indexes
 - about 4, 9
 - defining 65, 138, 164
 - defining for generic indexer 139

group indexes (*continued*)

pages in a group 63

GROUP_FIELD_NAME: parameter 138

GROUP_FIELD_VALUE: parameter 139

GROUP_FILENAME: parameter 139

GROUP_LENGTH: parameter 141

GROUP_OFFSET: parameter 141

GROUPMAXPAGES parameter

flags and values 63

OS/390 indexer 223

GROUPNAME parameter

flags and values 64

grouprange index 66

H

header pages

skipping 70, 166

I

IEL

See Index Element structured field

IMAGEOUT parameter

flags and values 64

IMM structured fields 72

Index Element structured field

considerations 109

defined 119

group-level 117

index object file 109

index exit 97

index exit parameter

OS/390 indexer 226

index file

indexfile parameter defined for JSMAIN, Xenos
transform 211

index object file

archiving considerations 110

DCB characteristics 68

DD statement for, OS/390 128

DD statement for, z/OS 128

INDEXDD parameter defined for ARSPDOCI, PDF
indexer 175

JCL parameter defined for ARSPDOCI, PDF indexer 174

INDEX parameter

flags and values 65, 164

JCL statement, OS/390 128

JCL statement, z/OS 128

OS/390 indexer 223

OS/390, JCL statement 128

z/OS, JCL statement 128

index record exit 70

index user exit 70

indexdd

JCL parameter defined for ARSPDOCI, PDF indexer 174

INDEXDD parameter

defined for ARSPDOCI, PDF indexer 175

flags and values 68, 165

indexer information

specifying for OS/390 indexer 237

indexes

about 5, 150

ACIF parameter 65

defining 19, 28, 40

generic indexer parameter 139

indexes (*continued*)

OS/390 indexer 223

PDF indexer parameter 164

Xenos transform 207

indexfile parameter

defined for JSMAIN, Xenos transform 211

indexing

Adobe PDF documents 145

AFP documents 181

bycol option 59

byrow option 59

concepts 4

constant field 57, 163

default index value 57, 163

EBCDIC data 10, 49, 52, 86, 90, 168

effect on document 121

examples 13

field mask 59, 162

field order 59

fields 5, 18, 27, 38, 55

fields for PDF indexer 161

file format 60

floating triggers 88

generic indexer 133

graphical indexer 153

group indexes 65, 164

groups 4, 9, 63

header pages 70, 166

helpful hints 109

index exit 95

indexes 5, 19, 28, 40, 65, 164

large object support 69, 224

line data 217

mask 90

mask option 59, 162

order option 59

OS/390 indexer 217

page indexes 3, 10, 65

parameters 5, 150

PDF indexer 145

recordrange triggers 87

reports

example of 13

skipping header pages 70, 166

TIFF images 217

transaction field 18, 58

trigger field 55, 161

triggers 5, 17, 26, 35, 86, 168

Xenos transform 181, 188

indexing parameters

example of 15, 24, 33, 46

INDEXOBJ parameter

flags and values 69

OS/390 indexer 224

INDEXSTARTBY parameter

flags and values 70, 166

INDEXSTYLE parameter

DOC reports and OS/390 indexer 230

NODX reports and OS/390 indexer 232

OS/390 indexer 229

PAGE reports and OS/390 indexer 230

PDOC reports and OS/390 indexer 231

INDEXEXIT parameter

flags and values 70

OS/390 indexer 226

inline resources

order of 111

- inline resources (*continued*)
 - writing to output file 111
- INPEXIT parameter
 - flags and values 71
 - OS/390 indexer 224
- input
 - OS/390 128
 - z/OS 128
- input exit parameter
 - OS/390 indexer 224
- input file
 - DD statement for ARSPDOCI, PDF indexer 174
 - DD statement for ARSPDUMP, PDF indexer 178
 - exit 95
 - generic indexer parameter 139, 141
 - INPUTDD parameter defined for ARSPDOCI, PDF indexer 176
 - inputfile parameter defined for JSMAIN, Xenos transform 205, 210
 - JCL parameter defined for ARSPDOCI, PDF indexer 174
- input record exit 71
- input user exit 71
- inputdd
 - JCL parameter defined for ARSPDOCI, PDF indexer 174
- INPUTDD parameter
 - defined for ARSPDOCI, PDF indexer 176
 - flags and values 71, 167
- inputfile parameter
 - defined for JSMAIN, Xenos transform 205, 210
- Invoke Medium Map
 - structured field 109

J

- JCL
 - ACIF JCL statement defined 128
 - concatenating ACIF files, OS/390 131
 - concatenating ACIF files, z/OS 131
 - concatenation example, OS/390 131
 - concatenation example, z/OS 131
 - DD:ddname parameter 237
 - example, OS/390 130
 - example, z/OS 130
 - for ACIF job, OS/390 130
 - for ACIF job, z/OS 130
 - for ACIF OS/390 jobs 127
 - for ACIF z/OS jobs 127
 - for ARSLOAD job, OS/390 indexer 237
 - for concatenating OS/390 files 131
 - for concatenating z/OS files 131
 - invoking ACIF program to index input file 129
 - OS/390 example 130, 131
 - OS/390 indexer example 237
 - OUTPUT JCL statement defined 128
 - PRINTOUT JCL statement defined 127
 - specifying application group 237
 - specifying temporary file for work space 237
 - statement defined, ACIF JCL 128
 - statement defined, OUTPUT JCL 128
 - statement defined, PRINTOUT JCL 127
 - z/OS example 130, 131
- job report file
 - DD statement for JSMAIN, Xenos transform 203, 209
- JSLOG JCL statement
 - for JSMAIN, Xenos transform 203, 209
- JSMAIN
 - examples 201, 207

- JSMAIN (*continued*)
 - FDDLOUT JCL statement 203
 - fddloutput parameter 205
 - fddmscript 205, 210
 - fdjobparm 205, 210
 - fdlicense parameter 205, 211
 - generic index file defined 214
 - index file 214
 - indexfile parameter 211
 - inputfile parameter 205, 210
 - JCL statement defined 203, 208
 - job report file 203, 209
 - JSLOG JCL statement 203, 209
 - license file defined 205, 211
 - M2PDOUT JCL statement 209
 - OS/390 JCL statement 202, 208
 - output file 209
 - outputfile parameter 205, 211
 - parameter file 204, 205, 209, 210
 - parameter file defined 205
 - script file 205, 210
 - script file defined 205, 211
 - text string location file 203, 205
 - z/OS JCL statement 202, 208

L

- large object support
 - ACIF 69
 - OS/390 indexer 224
- license file
 - fdlicense parameter defined for JSMAIN, Xenos transform 205, 211
- limitations
 - PDF indexer 159
- line data
 - AFP
 - converting to 6, 21, 31
 - converting to AFP 6, 21, 31
 - example of indexing 13, 21, 31
 - groups 63
 - indexing 13, 21, 31
 - indexing with the OS/390 indexer 217
 - pages in a group 63
- LINECNT parameter
 - flags and values 72
- links
 - PDF indexer 159

M

- M2PDOUT JCL statement
 - for JSMAIN, Xenos transform 209
- Map Coded Font Format 1 structured field
 - converted 124
- Map Coded Font Format 2 structured field
 - archival, document integrity 125
 - converting 73
 - including fonts 84
- mask 90
 - FIELD parameter option 59, 162
- maximum pages in a group 63
- MCF-1
 - See Map Coded Font Format 1 structured field
- MCF-2
 - See Map Coded Font Format 2 structured field

- MCF2 structured fields 73, 84
- MCF2REF parameter
 - flags and values 73
- message file
 - DD statement for, OS/390 128
 - DD statement for, z/OS 128
- messages
 - ACIF 93
 - ARSPDOCI program 171
 - PDF indexer 171
- Metacode
 - converting with Xenos transform 181
 - processing with Xenos transform 181
- MSGDD parameter
 - flags and values 73, 167
- multiple=up output
 - page definition 110

N

- naming input files
 - PDF indexer 159
- national language support (NLS) 52, 79, 137, 160, 199, 223
- NEWPAGE parameter
 - flags and values 74
- NLS 52, 79, 137, 160, 199, 223
- NODX reports and OS/390 indexer 232

O

- ODWEK
 - Xenos transform 195
- OnDemand application
 - indexer information 237
 - specifying OS/390 indexer 237
- opening reports 16, 26, 35
- order
 - bycol option 59
 - byrow option 59
 - FIELD parameter option 59
- order of inline resources 111
- OS/390
 - ACIF parameters 129
 - ACIF requirements 4, 127
 - Adobe fonts DD statement for ARSPDOCI, PDF indexer 175
 - Adobe fonts DD statement for ARSPDUMP, PDF indexer 178
 - concatenation example 131
 - DD statement for document file 127
 - FDDLOUT JCL statement for JSMAIN, Xenos transform 203
 - fddloutput parameter defined for JSMAIN, Xenos transform 205
 - fddmscript parameter defined for JSMAIN, Xenos transform 205, 210
 - fdjobparm parameter defined for JSMAIN, Xenos transform 205, 210
 - fdlicense parameter defined for JSMAIN, Xenos transform 205, 211
 - INDEX JCL statement 128
 - index object file 128
 - indexdd JCL parameter defined for ARSPDOCI, PDF indexer 174
 - INDEXDD parameter defined for ARSPDOCI, PDF indexer 175

- OS/390 (*continued*)
 - indexfile parameter defined for JSMAIN, Xenos transform 211
 - input 128
 - input file DD statement for ARSPDOCI, PDF indexer 174
 - input file DD statement for ARSPDUMP, PDF indexer 178
 - inputdd JCL parameter defined for ARSPDOCI, PDF indexer 174
 - INPUTDD parameter defined for ARSPDOCI, PDF indexer 176
 - inputfile parameter defined for JSMAIN, Xenos transform 205, 210
 - invoking ACIF 127
 - JCL example 130, 131
 - JCL for ACIF job 127
 - JCL for ARSPDOCI job 173
 - JCL for ARSPDUMP job 177
 - JCL for JSMAIN 201
 - JCL for PDF indexer job 173, 177
 - JCL for Xenos transform 201
 - JCL statement 127
 - JCL to invoke ACIF 127
 - job report file DD statement for JSMAIN, Xenos transform 203, 209
 - JSLOG JCL statement for JSMAIN, Xenos transform 203, 209
 - license file for JSMAIN, Xenos transform 205, 211
 - M2PDOUT JCL statement for JSMAIN, Xenos transform 209
 - message file, ACIF 128
 - output file DD statement for ARSPDUMP, PDF indexer 178
 - output file DD statement for JSMAIN, Xenos transform 209
 - OUTPUT JCL statement 128
 - outputdd JCL parameter defined for ARSPDOCI, PDF indexer 174
 - OUTPUTDD parameter defined for ARSPDOCI, PDF indexer 176
 - outputfile parameter defined for JSMAIN, Xenos transform 205, 211
 - parameter file DD statement for ARSPDOCI, PDF indexer 174
 - parameter file in JCL JSMAIN, Xenos transform 203, 208
 - parameters, ACIF 129
 - parmdd JCL parameter defined for ARSPDOCI, PDF indexer 173
 - RESOBJ statement 128
 - SYSIN JCL statement 128
 - SYSPRINT JCL statement 128
 - text string location file DD statement for JSMAIN, Xenos transform 203
 - USERAPPL statement 127
 - using ACIF 4, 127
 - using ARSPDOCI 173
 - using ARSPDUMP 177
 - using JSMAIN 201
 - using PDF indexer 173, 177
 - using Xenos transform 201
- OS/390 indexer
 - about 217
 - ANYEXIT parameter 227
 - Anystore Batch Capture Exit 227
 - application indexer 237
 - code page 223
 - DD:ddname parameter 237
 - DOC reports and INDEXSTYLE parameter 230

- OS/390 indexer (*continued*)
 - index exit parameter 226
 - indexer information 237
 - INDEXSTYLE parameter 229
 - INDEXEXIT parameter 226
 - INPEXIT parameter 224
 - input exit parameter 224
 - introduction 217
 - line data 217
 - national language support (NLS) 223
 - NLS 223
 - NODX reports and INDEXSTYLE parameter 232
 - OnDemand application 237
 - other parameters used by 224
 - overview 217
 - PAGE reports and INDEXSTYLE parameter 230
 - parameters 223, 224
 - PDOC reports and INDEXSTYLE parameter 231
 - specifying parameters for the ARSLOAD program 237
 - TIFF images 217
 - using 217, 237
- out-of-storage problem
 - See* Tag Logical Element structured field
- OUTEXIT parameter
 - flags and values 74
- output file
 - DCB characteristics 75
 - DD statement for ARSPDUMP, PDF indexer 178
 - DD statement for JSMAIN, Xenos transform 209
 - fddloutput parameter defined for JSMAIN, Xenos transform 205
 - format 121
 - JCL parameter defined for ARSPDOCI, PDF indexer 174
 - OUTPUTDD parameter defined for ARSPDOCI, PDF indexer 176
 - outputfile parameter defined for JSMAIN, Xenos transform 205, 211
 - writing inline resources to 111
- OUTPUT JCL statement
 - defined, OS/390 128
 - defined, z/OS 128
 - OS/390 128
 - z/OS 128
- output record exit 74, 98
- output user exit 74
- outputdd
 - JCL parameter defined for ARSPDOCI, PDF indexer 174
- OUTPUTDD parameter
 - defined for ARSPDOCI, PDF indexer 176
 - flags and values 75, 167
- outputfile parameter
 - defined for JSMAIN, Xenos transform 205, 211
- overlays 75
- OVLYLIB parameter
 - flags and values 75

P

- page definition
 - and resource file 125
 - fonts 51
 - multiple-up output 110
- page definitions 76, 78
- page indexes
 - about 3, 10, 122
 - defining 65
 - large object support 69

- PAGE reports and OS/390 indexer 230
- page segments 80
- page-level IELs 118
- PAGEDEF
 - fonts 51
- PAGEDEF parameter
 - flags and values 76
 - print file attributes 103
 - user exits 103
- pagerange index 67
- parameter file
 - comments 129
 - DD statement for ARSPDOCI, PDF indexer 174
 - DD statement for, OS/390 128
 - DD statement for, z/OS 128
 - fdjobparm parameter defined for JSMAIN, Xenos transform 205, 210
 - generic indexer 143
 - in JCL for JSMAIN, Xenos transform 203, 208
 - JCL parameter defined for ARSPDOCI, PDF indexer 173
 - PDF indexer 150
 - syntax rules, OS/390 129
 - syntax rules, z/OS 129
 - values spanning multiple records 129
- parameter values
 - spanning multiple records 129
- parameters
 - ANYEXIT parameter for OS/390 indexer 227
 - Anystore Batch Capture Exit for OS/390 indexer 227
 - ARSPDOCI program 161
 - CC 49
 - CCTYPE 49
 - CHARS 50
 - CODEPAGE: 137
 - COMMENT: 138
 - CONVERT 51
 - COORDINATES 161
 - CPGID 52
 - DCFPAGENAMES 53
 - EXTENSIONS 53
 - FDEFLIB 54
 - FIELD 55, 161, 223
 - FILEFORMAT 60, 223
 - FONTLIB 61
 - FORMDEF 61
 - generic indexer 137
 - GROUP_FIELD_NAME: 138
 - GROUP_FIELD_VALUE: 139
 - GROUP_FILENAME: 139
 - GROUP_LENGTH: 141
 - GROUP_OFFSET: 141
 - GROUPMAXPAGES 63, 223
 - GROUPNAME 64
 - IMAGEOUT 64
 - INDEX 65, 164, 223
 - index exit parameter for OS/390 indexer 226
 - INDEXDD 68, 165
 - indexing
 - example of 15, 24, 33, 46
 - INDEXOBJ 69
 - INDEXSTARTBY 70, 166
 - INDEXSTYLE parameter for OS/390 indexer 229
 - INDEXEXIT 70
 - INDEXEXIT parameter for OS/390 indexer 226
 - INPEXIT 71
 - INPEXIT parameter for OS/390 indexer 224
 - input exit parameter for OS/390 indexer 224

parameters (*continued*)

- INPUTDD 71, 167
- INSERTIMM 72
- LINECNT 72
- MCF2REF 73
- MSGDD 73, 167
- NEWPAGE 74
- OS/390 129
- OS/390 indexer 223, 224
- OUTEXIT 74
- OUTPUTDD 75, 167
- OVLYLIB 75
- PAGEDEF 76
- PARMDD 78, 168
- PDEFLIB 78
- PDF indexer 150, 161
- PRMODE 79
- PSEGLIB 80
- RESEXIT 81
- RESFILE 81
- RESOBJDD 82
- RESTYPE 83
- TRACE 85
- TRC 85
- TRIGGER 86, 168, 223
- UNIQUEBNBS 89
- USERLIB 89
- USERMASK 90
- with INDEXSTYLE=DOC for OS/390 indexer 230
- with INDEXSTYLE=NODX for OS/390 indexer 232
- with INDEXSTYLE=PAGE for OS/390 indexer 230
- with INDEXSTYLE=PDOC for OS/390 indexer 231
- Xenos transform 197
- z/ OS 129

PARM

- parameter file in JCL for JSMAIN, Xenos transform 203, 208

parmdd

- JCL parameter defined for ARSPDOCI, PDF indexer 173

PARMDD parameter

- flags and values 78, 168

PDEFLIB parameter

- flags and values 78

PDF indexer

- about 145
- Adobe fonts DD statement 175, 178
- ARSPDOCI 173
- ARSPDUMP 177
- bookmarks 159
- code page 160
- concepts 150
- constant field 163
- coordinate system 150
- default index value 163
- EBCDIC data 175
- error messages 171
- examples of ARSPDOCI 173
- examples of ARSPDUMP 177
- field mask 162
- fields 161
- file naming conventions 159
- fonts 159
- graphical indexer 153
- group indexes 164
- indexes 164
- indexing concepts 150
- introduction 145

PDF indexer (*continued*)

- JCL statement defined 173, 177
- limitations 159
- links 159
- mask option 162
- messages 171
- naming input files 159
- national language support (NLS) 160
- NLS 160
- OS/390 JCL statement 173, 177
- overview 145
- parameter file 150, 175
- parameter file defined 175
- parameter reference 161
- printing 159
- requirements 159
- restrictions 159
- system requirements 159
- transferring input files to 159
- trigger field 161
- triggers 168
- using 145
- x, y coordinate system 150
- z/OS JCL statement 173, 177

PDOC reports and OS/390 indexer 231

Portable Document Format (PDF)

- See* PDF indexer

print file attributes 102

- CC parameter 102
- cctype parameters 102
- example 102
- formdef parameters 103
- pagedef parameters 103
- parameter, cc 102
- parameters, cctype 102
- parameters, formdef 103
- parameters, pagedef 103
- parameters, prmode 103
- parameters, trc 103
- prmode parameters 103
- TRC parameters 103
- user exits 102

printing

- PDF indexer 159

PRINTOUT JCL statement

- defined 127

PRMODE parameter

- flags and values 79
- print file attributes 103
- user exits 103

PSEGLIB parameter

- flags and values 80

R

- recordrange triggers 87
- REGION size for ACIF 128
- reports
 - accessing 13, 21, 32, 44
 - example of 13, 21, 32, 44
 - format 60
 - indexing 13
 - opening 16, 26, 35
- requirements
 - PDF indexer 159
- RESEXIT parameter
 - flags and values 81

- RESOBJDD parameter
 - flags and values 82
- RESOBJDD statement
 - OS/390 128
 - z/OS 128
- resource exit
 - provided with ACIF 100
- resource file
 - DD statement for, OS/390 128
 - DD statement for, z/OS 128
 - format 115
- resource group file
 - DCB characteristics 83
- resource retrieval
 - file format 115
 - resource exit 100
- resource user exit 81
- resources
 - about 8
 - exits 81
 - file 81
 - fonts 61
 - form definitions 54, 61
 - group 81
 - inline 111
 - inline, writing to output file 111
 - overlays 75
 - page definitions 76, 78
 - page segments 80
 - RESTYPE parameter 83
 - types of 83
 - user-defined 89
 - Xenos transform 189
- restrictions
 - PDF indexer 159
- RESTYPE parameter
 - flags and values 83
- running ACIF with inline resources 111

S

- script file
 - fddmscript parameter defined for JSMAIN, Xenos transform 205, 210
- separator pages
 - application-generated 110
 - removal from output 110
- Set Coded Font Local structured field 79
- skipping header pages 70, 166
- software
 - requirements for PDF indexer 159
- SOSI 79
- storage problem
 - See* Tag Logical Element structured field
- structured fields
 - Begin Document 110, 123
 - Begin Document Index 118
 - Begin Named Group 121, 123
 - Begin Page 124
 - Begin Resource 116
 - Begin Resource Group 116
 - Composed Text Control (obsolete) 124
 - End Document 124
 - End Document Index 120
 - End Named Group 121, 124
 - End Page 124
 - End Resource 116

- structured fields (*continued*)
 - End Resource Group 116
 - group level 117
 - Index Element 109, 117, 118, 119
 - Invoke Medium Map 72, 109
 - Map Coded Font Format 1 124
 - Map Coded Font Format 2 73, 125
 - page level 117
 - Presentation Text Data Descriptor 125
 - Set Coded Font Local 79
 - Tag Logical Element 109, 113, 119, 121, 124
- SYSIN JCL statement
 - OS/390 128
 - z/OS 128
- SYSPRINT JCL statement
 - OS/390 128
 - z/OS 128
- system requirements
 - PDF indexer 159

T

- Tag Logical Element structured field
 - as part of the indexing process 119, 124
 - created in the output document file 121
 - defined 119
 - examples and rules 113
 - in named groups 106
 - out-of-storage problem, possible cause 106
 - storage problem, possible cause 106
- text string location file
 - DD statement for JSMAIN, Xenos transform 203
 - fddloutput parameter for JSMAIN, Xenos transform 205
- TIFF images
 - indexing with the OS/390 indexer 217
- TLE
 - See* Tag Logical Element structured field
- TLEs
 - example of indexing 44
 - indexing 44
- TRACE parameter
 - flags and values 85
- transaction field 18, 58
- translation reference characters (TRC) 50, 85
- TRC 50, 85
- TRC parameter
 - flags and values 85
 - print file attributes 103
 - user exits 103
- trigger field 55, 161
- TRIGGER parameter
 - options and values 86, 168
 - OS/390 indexer 223
- triggers
 - about 5, 150
 - ACIF parameter 86
 - defining 17, 26, 35
 - floating 88
 - floating and groupname 64
 - OS/390 indexer 223
 - PDF indexer parameter 168
 - recordrange 87
 - Xenos transform 207

U

- UNIQUEBNGS parameter
 - flags and values 89
- user exits
 - index 70, 97
 - input 71, 95
 - output 74
 - output record 98
 - print file attributes 102
 - provided with ACIF 95
 - resource 81
 - resource, provided with ACIF 100
- user programming exit 95
- USERAPPL
 - OS/390 statement 127
 - z/OS statement 127
- USERLIB
 - form definitions 62
 - page definitions 77
 - requesting access to 90
- USERLIB parameter
 - flags and values 89
- USERMASK parameter
 - flags and values 90
- using ACIF
 - in the OS/390 environment 127
 - in the z/OS environment 127
- using ARSPDOCI
 - in the OS/390 environment 173
 - in the z/OS environment 173
- using ARSPDUMP
 - in the OS/390 environment 177
 - in the z/OS environment 177
- using JSMAIN
 - in the OS/390 environment 201
 - in the z/OS environment 201
- using PDF indexer
 - in the OS/390 environment 173, 177
 - in the z/OS environment 173, 177
- using Xenos transform
 - in the OS/390 environment 201
 - in the z/OS environment 201

W

- Web Enablement Kit
 - Xenos transform 195
- writing inline resources to output file 111

X

- x,y coordinate system 150
- Xenos transform
 - about 181
 - AFP to HTML 185
 - AFP to PDF 185
 - AFP to XML 185
 - code page 199
 - examples 193, 201, 207
 - examples of JSMAIN 201
 - FDDLOUT JCL statement 203
 - fddloutput parameter 205
 - fddmscript 205, 210
 - fdjobparm 205, 210
 - fdlicense parameter 205, 211
 - generic index file defined 214

- Xenos transform (*continued*)
 - index file 214
 - indexfile parameter 211
 - indexing 188
 - inputfile parameter 205, 210
 - introduction 181
 - JCL statement defined 203, 208
 - job report file 203, 209
 - JSLOG JCL statement 203, 209
 - JSMAIN 201
 - license file defined 205, 211
 - loading data 191
 - M2PDOUT JCL statement 209
 - Metacode to AFP 185
 - Metacode to HTML 185
 - Metacode to Metacode 185
 - Metacode to PDF 185
 - Metacode to XML 185
 - national language support (NLS) 199
 - NLS 199
 - OS/390 JCL statement 202, 208
 - output file 209
 - outputfile parameter 205, 211
 - overview 181
 - parameter file 204, 205, 209, 210
 - parameter file defined 205
 - parameters 197
 - PCL to PDF 185
 - resources 189
 - script file 205, 210
 - script file defined 205, 211
 - text string location file 203, 205
 - using 181
 - Web Enablement Kit 195
 - WEK 195
 - z/OS JCL statement 202, 208

Z

- z/OS
 - ACIF parameters 129
 - ACIF requirements 4, 127
 - Adobe fonts DD statement for ARSPDOCI, PDF indexer 175
 - Adobe fonts DD statement for ARSPDUMP, PDF indexer 178
 - concatenation example 131
 - DD statement for document file 127
 - FDDLOUT JCL statement for JSMAIN, Xenos transform 203
 - fddloutput parameter defined for JSMAIN, Xenos transform 205
 - fddmscript parameter defined for JSMAIN, Xenos transform 205, 210
 - fdjobparm parameter defined for JSMAIN, Xenos transform 205, 210
 - fdlicense parameter defined for JSMAIN, Xenos transform 205, 211
 - INDEX JCL statement 128
 - index object file 128
 - indexdd JCL parameter defined for ARSPDOCI, PDF indexer 174
 - INDEXDD parameter defined for ARSPDOCI, PDF indexer 175
 - indexfile parameter defined for JSMAIN, Xenos transform 211
 - input 128

z/OS (continued)

- input file DD statement for ARSPDOCI, PDF indexer 174
- input file DD statement for ARSPDUMP, PDF indexer 178
- inputdd JCL parameter defined for ARSPDOCI, PDF
indexer 174
- INPUTDD parameter defined for ARSPDOCI, PDF
indexer 176
- inputfile parameter defined for JSMAIN, Xenos
transform 205, 210
- invoking ACIF 127
- JCL example 130, 131
- JCL for ACIF job 127
- JCL for ARSPDOCI job 173
- JCL for ARSPDUMP job 177
- JCL for JSMAIN 201
- JCL for PDF indexer job 173, 177
- JCL for Xenos transform 201
- JCL statement 127
- JCL to invoke ACIF 127
- job report file DD statement for JSMAIN, Xenos
transform 203, 209
- JSLOG JCL statement for JSMAIN, Xenos transform 203,
209
- license file for JSMAIN, Xenos transform 205, 211
- M2PDOUT JCL statement for JSMAIN, Xenos
transform 209
- message file, ACIF 128
- output file DD statement for ARSPDUMP, PDF
indexer 178
- output file DD statement for JSMAIN, Xenos
transform 209
- OUTPUT JCL statement 128
- outputdd JCL parameter defined for ARSPDOCI, PDF
indexer 174
- OUTPUTDD parameter defined for ARSPDOCI, PDF
indexer 176
- outputfile parameter defined for JSMAIN, Xenos
transform 205, 211
- parameter file DD statement for ARSPDOCI, PDF
indexer 174
- parameter file in JCL for JSMAIN, Xenos transform 203,
208
- parameters, ACIF 129
- parmdd JCL parameter defined for ARSPDOCI, PDF
indexer 173
- RESOBJ statement 128
- SYSIN JCL statement 128
- SYSPRINT JCL statement 128
- text string location file DD statement for JSMAIN, Xenos
transform 203
- USERAPPL statement 127
- using ACIF 4, 127
- using ARSPDOCI 173
- using ARSPDUMP 177
- using JSMAIN 201
- using PDF indexer 173, 177
- using Xenos transform 201



Program Number: 5655-H39

SC27-1375-04

